LAPIN AMK
Lapland University of Applied Sciences

VESIVOIMALAITOKSEN LÄMPÖTILA-ANTUREIDEN VISUALISOINTI WEB-SELAIMESSA

Matti Siltanen

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2022

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Matti Siltanen	Vuosi	2022
Ohjaaja	Aku Kesti		
Toimeksiantaja	Lapin AMK:n FrostBit-ohjelmistolaboratorio		
Työn nimi	Vesivoimalaitoksen lämpötila-antureiden visualisointi web-selaimessa		
Sivu- ja liitesivumäärä	46 + 3		

Tämän opinnäytetyön tavoitteena oli kehittää web-selaimessa toimiva sovellus, jossa esitetään vesivoimalaitoksen anturikaapeleiden tuottamia mittaustuloksia reaaliajassa. Vesivoimalaitoksen padon vallin läheisyydessä sijaitsee erilaisia anturikaapeleita, jotka ottavat lämpötilanäytteitä vallista. Suunnitelmana oli luoda vesivoimalaitoksen lähiympäristöstä 3D-mallinnus, jossa näkyvät vallin läheisyydessä olevat anturikaapelit ja niiden ilmoittamat lämpötilalukemat. Sovellukseen oli myös kaavailtu graafisen esityksen lisäämistä, jossa esitettäisiin aikaisemmin mitattuja lämpötila-arvoja diagrammeissa. Sovelluksen tarkoitus on tehostaa padon vallin olosuhdevalvontaa analyyttisin sekä visuaalisin keinoin.

Sovellus toteutettiin JavaScript-pohjaisella Babylon.js-nimisellä 3D-pelimoottorilla. Babylon.js-pelimoottori mahdollisti interaktiivisen 3D-maailman luomisen web-selaimen näkymään. Vesivoimalaitoksen lähiympäristöä esittävät 3D-mallit luotiin Blender-mallinnusohjelmalla. Luodut 3D-mallit ladattiin web-selaimeen esitettäväksi Babylon.js:llä. Babylon.js:stä löytyi paljon muutakin kriittistä toiminnallisuutta sovelluksen toteuttamisen kannalta. Babylon.js:stä luotiin paikallinen asennus, ja sitä ajettiin Webpackin DevServer-palvelinohjelmistolla. Sovelluksen käyttöliittymän ulkoasu toteutettiin HTML5/CSS3-merkintäkielillä. Sovelluksen ohjelmalogiikka puolestaan toteutettiin JavaScript-ohjelmointikielellä.

Lopputuloksena syntyi web-sovellus, jossa vesivoimalaitoksen anturikaapeleiden tuottamia lämpötilatietoja voitiin esittää web-selaimen ruudulla 3D-maailman näkymässä ja sekä graafisessa esityksessä. Sovelluksessa esitetään reaaliajassa päivittyvää lämpötilatietoa anturikaapeleita kuvastavissa 3D-malleissa. Lisäksi lämpötilatietoa on mahdollista hakea loppukäyttäjän antamalta päivämääräväliltä ja esittää se erityyppisissä diagrammeissa. Sovellukseen luotiin monipuolisia tapoja tiedon esittämiselle. Sovellus täytti kaikki sille asetetut kriittiset tavoitteet. Joitakin suunniteltuja lisätoimintoja jäi tekemättä aikarajoitteiden takia.

Avainsanat Babylon.js, JavaScript, web-ohjelmointi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Information and Communication Technology
Bachelor of Engineering

Author	Matti Siltanen	Year	2022
Supervisor	Aku Kesti		
Commissioned by	FrostBit Software Laboratory		
Subject of thesis	Visualization of Power Plant's Temperature Sensor Data in Web Browser		
Number of pages	46 + 3		

The objective of this thesis was to develop a web application focused primarily on the visualization of sensor data on a web browser in real-time. The sensor data is generated by a few sets of sensor cables of different types. These sensor cables are located in close proximity of a dam wall near a hydroelectric power plant. The initial plan consisted of creating a simple 3D model of the power plant's nearby environment. An end user would have the ability to examine sensor readings generated by the sensors in the 3D model. Furthermore, the plan also included implementing a graphical representation of the sensor data. The purpose of this application is to provide a way to monitor the structural integrity of the dam wall and to detect possible leakages through data visualization.

The first step in the development process was to create a fully interactive 3D world view that would be displayed on a web page. This was achieved by using a 3D game engine called Babylon.js. Babylon.js had several important features that made the development of this application possible. One of the key features included a way to import 3D models to the 3D world view that were made with a modeling software called Blender. Furthermore, Babylon.js engine was installed locally on a development server provided by Webpack library. The application's user interface was made in HTML5 and CSS3. Additionally, the program logic used in the application was programmed in JavaScript.

The end result of this project is an application, in which an end user is able to observe sensor data originating from the power plant in real-time. The application offers various ways of data visualization. For example, sensor readings are displayed in message boxes when an end user performs certain actions in the 3D world view. Sensor data is also represented in different types of graphs and charts. In conclusion, the application contains all critical functionality that was planned beforehand. Overall, the development process was successful.

Key words Babylon.js, JavaScript, web development

SISÄLLYS

1 JOHDANTO 5

2 SUUNNITELMA 6

2.1 Tehtävän kokonaiskuvaus 6

2.2 Suunnitellut ohjelmistot ja riippuvuudet 7

3 BABYLON.JS-MOOTTORI 8

3.1 Esittely 8

3.2 Paikallinen asennus 9

4 MALLINNUS 13

4.1 Yleiskuvaus 13

4.2 3D-mallinnus 14

4.3 Mallin lataaminen Babylon.js-moottorilla 16

4.4 Maaston proseduraalinen generointi 19

4.5 3D-maailman kokonaisuuden rakentaminen malleilla 21

5 LÄMPÖTILATIEDON PROSESSOINTI 24

5.1 Kuvaus 24

5.2 Lämpötilatiedon automaattinen päivitys 24

5.3 Lämpötilojen värikoodaus 25

5.4 Anturikaapelimallin korjaus 28

5.5 Tapahtumankäsittelijän kiinnitys malliin 31

6 LÄMPÖTILATIEDON GRAAFINEN ESITYS 34

6.1 Käyttöliittymän toteutus 34

6.2 Lämpötilatiedon esitys diagrammeissa 35

7 SOVELLUKSEN VIIMEISTELY 38

7.1 Yleiskuvaus 38

7.2 Viimeisten komponenttien luonti 38

8 POHDINTA 43

LÄHTEET 45

LIITTEET 46

1 JOHDANTO

Opinnäytetyön tilaajana toimi Lapin ammattikorkeakoulun FrostBit-ohjelmistolaboratorio. FrostBit on mukana Laajapintaisen painetun elektroniikan integraatio Smart City -sovelluksissa -hankkeessa. Hankkeen yhtenä osana on toimittaa vesivoimalaitokselle visuaalinen web-selaimessa toimiva sovellus, jossa esitetään vesivoimalaitoksen lähiympäristössä sijaitsevien anturikaapeleiden tuottamia lämpötilatietoja. Opinnäytetyössä käydään tämän sovelluksen toteutusprosessi läpi vaihe vaiheelta.

Vesivoimalaitoksen padon valliin on asennettu kuusi kappaletta erilaisia anturikaapeleita, jotka mittaavat padon vallin lämpötilaa reaaliajassa. Kaapeleihin on kiinnitetty useita lämpötila-antureita, joiden kappalemäärä vaihtelee kaapelikohtaisesti. Anturikaapeleiden tuottamien lämpötilatietojen analysoinnin avulla voidaan havaita ja paikantaa mahdollisia padon vesivuotoja. Hypoteettisessa tilanteessa, jossa pato alkaisi vuotamaan, vettä valuisi anturikaapeleiden päälle, minkä johdosta antureiden lämpötilalukemat alkaisivat laskemaan nopeasti ja vuoto voitaisiin paikantaa välittömästi. Vesivoimalaitoksella on tarvetta sovellukselle, jossa anturikaapeleiden tuottamia lämpötilatietoja voidaan helposti tarkkailla ja analysoida.

Sovellukselle on asetettu seuraavat vaatimukset. Vesivoimalaitoksen lähiympäristöstä tulee tehdä 3D-malli, josta ilmenee kaapeleiden fyysinen sijainti suhteessa ympäristöön. Kaikissa kaapeleissa olevien yksittäisten antureiden tuottamia senhetkisiä sekä aikaisempia lämpötilatietoja pitää pystyä tarkastelemaan. Sovelluksessa tulee olla mahdollisuus etsiä lämpötilatietoja syötetyltä aikaväliltä. Lämpötilatietojen tulee olla havainnollistettu selkeästi. Tarkoituksena on, että kaapeleiden lämpötiloja ja padon kuntoa voidaan seurata etänä reaaliajassa.

Sovelluksen toteuttamiseen liittyy web-ohjelmointia, 3D-mallinnusta ja käyttöliittymän sekä visualisoinnin suunnittelua. Toteutukseen kuuluu myös anturitiedon prosessointia sekä keräämistä valmiin rajapinnan kautta.

2 SUUNNITELMA

2.1 Tehtävän kokonaiskuvaus

Vesivoimalaitoksen lähialueen ympäristöstä suunniteltiin tehtäväksi 3D-malli, jossa näkyisivät muun muassa padon valli, maantiet, ulkorakennukset, kulkureitit ja anturikaapelit. Malli olisi interaktiivinen, eli loppukäyttäjä voisi hiirellä vapaasti pyöritellä ja vaihtaa mallin kuvakulmaa ja suurennustasoa. Klikkaamalla ruudulla olevaa anturikaapelia käyttäjä saisi tarkempaa tietoa kaapelista. Mallissa olevat anturikaapelit värikoodattaisiin tuoreimman lämpötilan perusteella.

Sovellukseen suunniteltiin myös lisättäväksi graafinen esitys, jossa voitaisiin esittää anturikaapeleiden lämpötilatietoja. Lämpötilatietoja näytettäisiin käyttäjän antamalta aikaväliltä. Graafisessa esityksessä olisi mahdollista tarkastella lämpötilatietoja loppukäyttäjän valitsemalta kaapelilta. Loppukäyttäjä voisi halutessaan nähdä kaapelin jokaisen anturin lämpötilatiedot samanaikaisesti tai vaihtoehtoisesti vain yksittäisen anturin lämpötilatiedot.

Vesivoimalaitokselta tuleva anturitieto tallentuu FrostBitin palvelimelle. FrostBit on kehittänyt oman rajapinnan, jonka kautta anturitietoa voidaan hakea palvelimelta tavallisella GET-kutsulla. Tästä syystä johtuen erillistä tietokantaa anturitiedon säilyttämiseksi ei tehdä sovelluksen osalta. Palvelimelta haettu anturitieto vastaanotetaan JSON-muodossa.

Suunnitelmissa oli myös lisätä sovellukseen automaattinen varoitusjärjestelmä ja tuki responsiiviselle selaukselle mobiililaitteilla, mikäli aikaa riittäisi. Varoitusjärjestelmä tarkkailisi anturikaapeleiden lämpötilojen keskilukuja ja varottaisi loppukäyttäjää, jos järjestelmä havaitsisi poikkeamia lämpötiloissa. Poikkeama voi johtua esimerkiksi veden valumisesta anturin päälle, jolloin anturin lämpötila lähtee nopeaan laskuun.

Sovellus suunniteltiin olemaan luonteeltaan web-pohjainen ja se esitettäisiin web-sivuina. Sovelluksen piti tukea yleisempiä web-selaimia, esimerkiksi Google Chromea, Microsoft Edgeä ja Mozilla Firefoxia. Web-ohjelmointi olisi pääsään-

töisesti frontend-painotteista. 3D-mallin näyttäminen ja lämpötilatiedon hakeminen suunniteltiin toteutettavaksi frontend-puolella. Sovelluksen tuli olla luonteeltaan helppokäyttöinen ja visuaalisesti selkeä loppukäyttäjälle.

Opinnäytetyön ohessa ei julkaista tuotettuja ohjelmakoodeja (pl. liitetiedostot) eikä myöskään toimeksiantajalta saatuja referenssimateriaaleja. Opinnäytetyössä esitetyt anturikaapeleiden lämpötilatiedot ovat kuvitteellisia. Lämpötilatiedon hakuun tarkoitettua rajapinnan dokumentaatiota ja URL-osoitteita ei sisällytetä opinnäytetyöhön.

2.2 Suunnitellut ohjelmistot ja riippuvuudet

Sovellus suunniteltiin toteutettavaksi HTML5-, CSS3- ja JavaScript-merkintä- ja ohjelmointikielillä. Sovelluksessa suunniteltiin käytettäväksi myös valmiita JavaScript-kirjastoja ja -ohjelmakehyksiä. JavaScript-pohjaiset riippuvuudet asennettaisiin npm-paketinhallintaohjelmalla. 3D-mallit luotaisiin Blender-mallinnusohjelmalla. Luodut 3D-mallit ladattaisiin ja esitettäisiin ruudulla Babylon.js-nimisen 3D-pelimoottorikirjaston avulla. Babylon.js:stä luotaisiin paikallinen asennus Webpack DevServer-palvelinohjelmistoa hyödyntäen. Anturikaapeleiden lämpötilojen graafinen esitys toteutettaisiin Plotly.js-kirjastolla. Ohjelmointiympäristönä käytettäisiin Visual Studio Code -ohjelmaa.

Babylon.js:ssä suunniteltiin käytettäväksi seuraavia riippuvuuksia:

- @babylonjs/core
- @babylonjs/materials
- @babylonjs/loaders
- @babylonjs/gui.

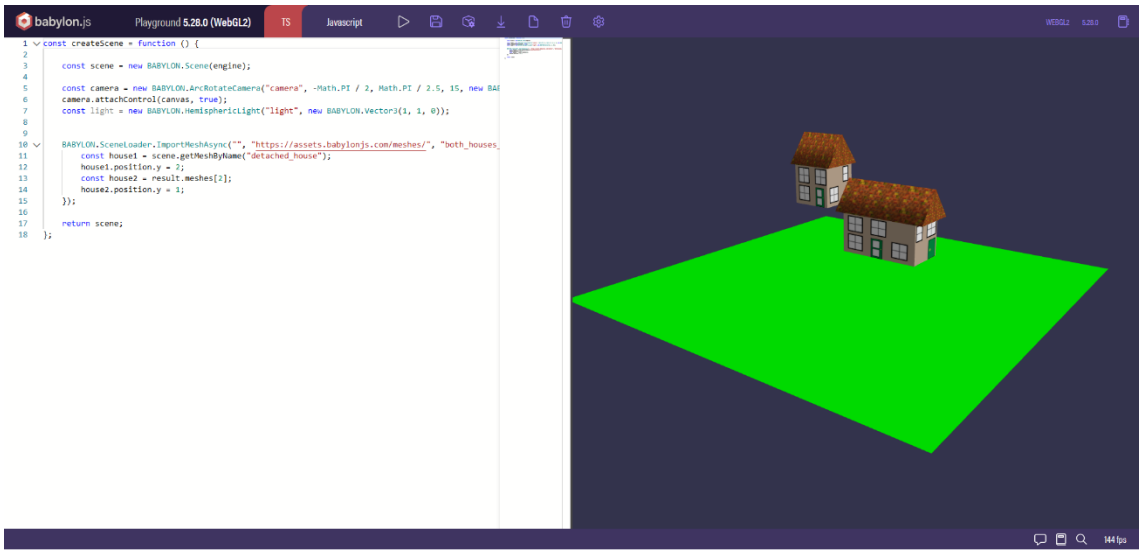
Riippuvuudet voidaan asentaa npm-ohjelmalla.

3 BABYLON.JS-MOOTTORI

3.1 Esittely

Babylon.js on JavaScript-pohjainen web-selaimessa 3D-grafiikan esittämiseen tarkoitettu pelimoottori. Babylon.js:n on kehittänyt Microsoft. Babylon.js-moottorin avulla web-selaimeen luodaan 3D-perspektiivissä esitetty maailma, jota ikään kuin katsotaan kameran kautta. Kameran kuvakulmaa voi vapaasti vaihtaa hiirellä. 3D-maailmassa voidaan esittää 2- tai 3D-grafiikkaa. Babylon.js tukee Blender-ohjelmalla luotujen 3D-mallien lataamista ja esittämistä edellä mainitussa 3D-maailmassa. Babylon.js:n avulla voidaan myös liittää tapahtumankäsittelijöitä malleihin, joiden avulla voidaan havaita esimerkiksi kaikki sellaiset tapahtumat, joissa loppukäyttäjä klikkaa hiirellä ruudulla olevaa mallia. Babylon.js mahdollistaa myös 2- ja 3D-käyttöliittymien luomisen sovelluksessa. Babylon.js:ssä on myös monia muita hyödyllisiä ominaisuuksia, ja se toimii keskeisessä roolissa sovelluksen kehityksessä. (Babylon.js 2022a.)

Vesivoimalaitoksen lähiympäristöä kuvaavat mallit ladataan ja esitetään web-selaimessa Babylon.js:n avulla. Tätä varten ensiksi on luotava kehitysympäristö, jossa moottoria voidaan käyttää. Babylon.js tarjoaa kehitysympäristön luomiseen kaksi eri tapaa. Ensimmäisenä vaihtoehtona on kehittää sovellusta Babylon.js:n omilta kotisivuilta löytyvältä Playground-editorin avulla. Playground-editori on virtuaalinen hiekkalaatikko, jossa voidaan kehittää ja kokeilla Babylon.js-pohjaista ohjelmakoodia. Editorissa on muun muassa sisäänrakennettu tekstikenttäalue, johon koodia voi kirjoittaa ja sekä renderöintialue, jossa 3D-grafiikka esitetään. Ohjelmakoodit tallentuvat Babylon.js:n omille palvelimille, jotka voidaan tarvittaessa myös kopioida paikalliselle tietokoneelle. Seuraavassa kuviossa 1 näkyy Playground-editori. (Babylon.js 2022b.)



Kuvio 1. Playground-editorin käyttöliittymä (Babylon.js 2022b)

Toisena vaihtoehtona on tehdä Babylon.js:stä paikallinen asennus, jossa kehitysympäristöä ajetaan Webpackin DevServer-palvelinohjelmiston kautta paikallisessa verkko-osoitteessa eli niin kutsutussa localhostissa. Suoritettava ohjelmakoodi tallennetaan paikalliselle koneelle. Paikalliseen asennukseen ei sisälly valmista koodieditoria. Paikallisen asennuksen etuna on se, että ohjelmakoodit voidaan jaotella tarvittaessa erillisiin tiedostoihin parantaen koodin rakennetta ja ylläpidettävyyttä. Myös paikallisten resurssien, esimerkiksi 3D-mallien lataaminen on helpompaa. Lisäksi sovelluksen kehityksen tehokkuutta voidaan parantaa käyttämällä kehittyneempiä ja monipuolisempia kehitystyökaluja. (Babylon.js 2022c.)

Tämän sovelluksen kehitysympäristö on päätetty toteuttaa Babylon.js:n paikallisella asennuksella, joka käydään läpi seuraavassa alaluvussa. Asennus on suoritettu Windows-pohjaisella käyttöjärjestelmällä.

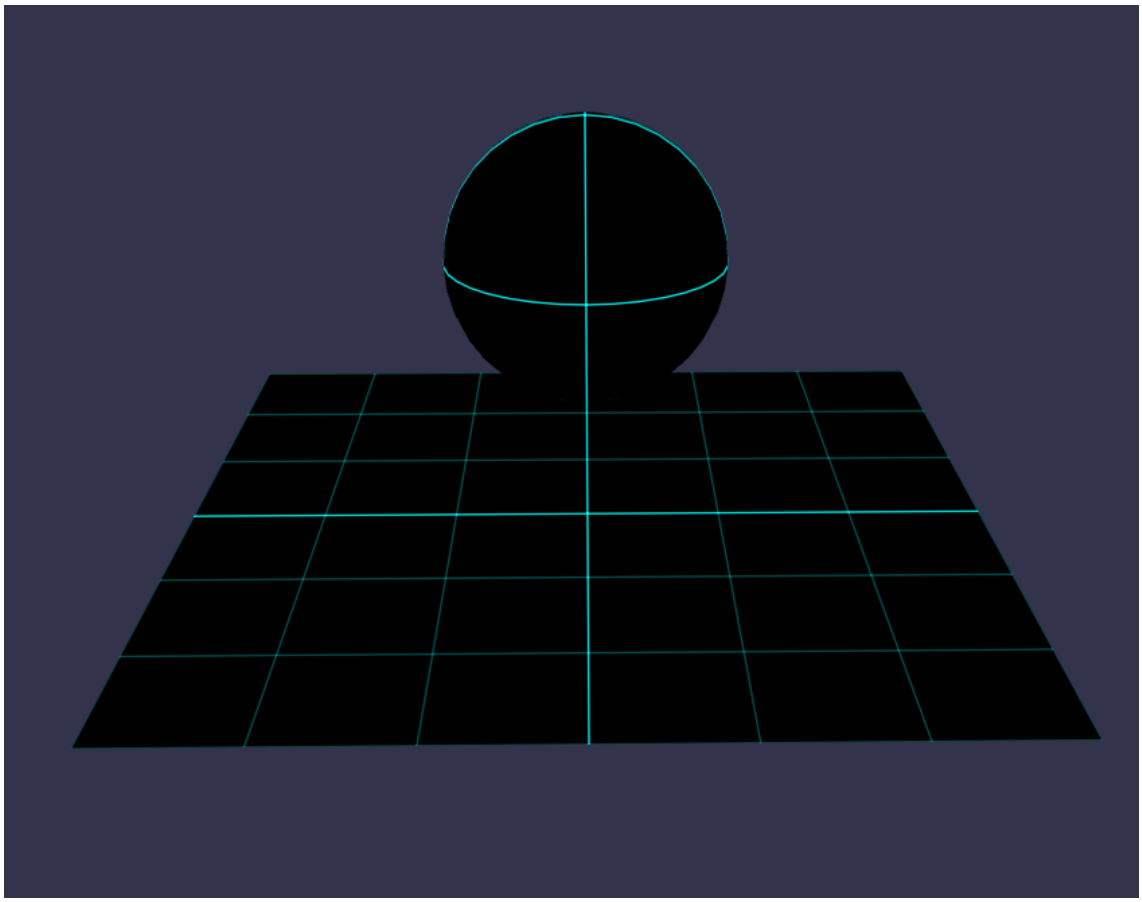
3.2 Paikallinen asennus

Babylon.js-moottorin paikallinen asentaminen on hyvin yksinkertainen prosessi. Ensimmäisenä vaiheena on asentaa Node.js-ympäristö ja npm-paketinhallintaohjelma. Npm-ohjelma tulee Node.js:n mukana. Node.js on ajonaikainen ympäristö, jonka avulla JavaScript-koodia voidaan suorittaa palvelimella. Npm-ohjelmalla voidaan asentaa ja hallinnoida Node.js:n paketteja. (Treehouse 2022.)

Toisena vaiheena on luoda projektia varten uusi kansio, johon Babylon.js ja Webpackin DevServer asennetaan. Oletetaan, että projektikansio on luotu hakemistoon ”C:\js\babylon”. Npm-ohjelma täytyy ensin alustaa projektikansiossa ennen Babylon.js:n asentamista. Tämä voidaan toteuttaa aukaisemalla komentorivi ja navigoimalla projektikansioon syöttämällä komennoksi *cd C:\js\babylon*. Npm-ohjelman alustaminen toteutetaan komennolla *npm init -y*. Webpackin DevServer voidaan asentaa komennolla *npm install webpack webpack-cli webpack-dev-server --save-dev* ja Babylon.js vastaavasti komennolla *npm install --save-dev @babylonjs/core*. (Babylon.js 2022c.)

Npm-ohjelma luo projektikansioon asennustiedostojen lisäksi node_modules-nimisen kansion, jossa sijaitsee tarvittavat riippuvuudet ja kirjastot. Asennuksen jälkeen projektikansioon pitää luoda public- ja src-nimiset kansiot erikseen. Public-kansiossa sijaitsevat kaikki web-sivuston julkiset resurssit, esimerkiksi HTML-tiedostot. Src-kansiossa sijaitsevat projektin lähdekoodit, esimerkiksi JavaScript-tiedostot, joista Webpack rakentaa web-sivun. Seuraavaksi on luotava index.html-tiedosto public-kansioon ja index.js-tiedosto src-kansioon. Tiedostoihin tuleva sisältö löytyy opinnäytetyön liitteistä 1 ja 2. Lisäksi projektin juurikansioon on luotava webpack-config.js-tiedosto, jonka sisältö löytyy liitteestä 3. (Babylon.js 2022c.)

Babylon.js:n valinnaisia riippuvuuksia saa asennettua komennolla *npm install --save-dev riippuvuuden nimi*. Asennuksessa on käytetty @babylonjs/materials-nimistä riippuvuutta. Se saadaan asennettua komennolla *npm install --save-dev @babylonjs/materials*. Kun tarvittavat tiedostot on lisätty ja riippuvuudet on asennettu, Webpackin DevServer-palvelimenohjelmiston saa päälle komennolla *npx webpack serve*. Asennuksen lopputulosta voi tarkastella web-selaimella paikallisessa verkko-osoitteessa http://localhost:8080/. Seuraavassa kuviossa 2 näkyy web-selaimessa Babylon.js:llä renderöityä grafiikkaa. (Babylon.js 2022c.)



Kuvio 2. Babylon.js-moottorilla renderöity 3D-grafiikka web-selaimessa

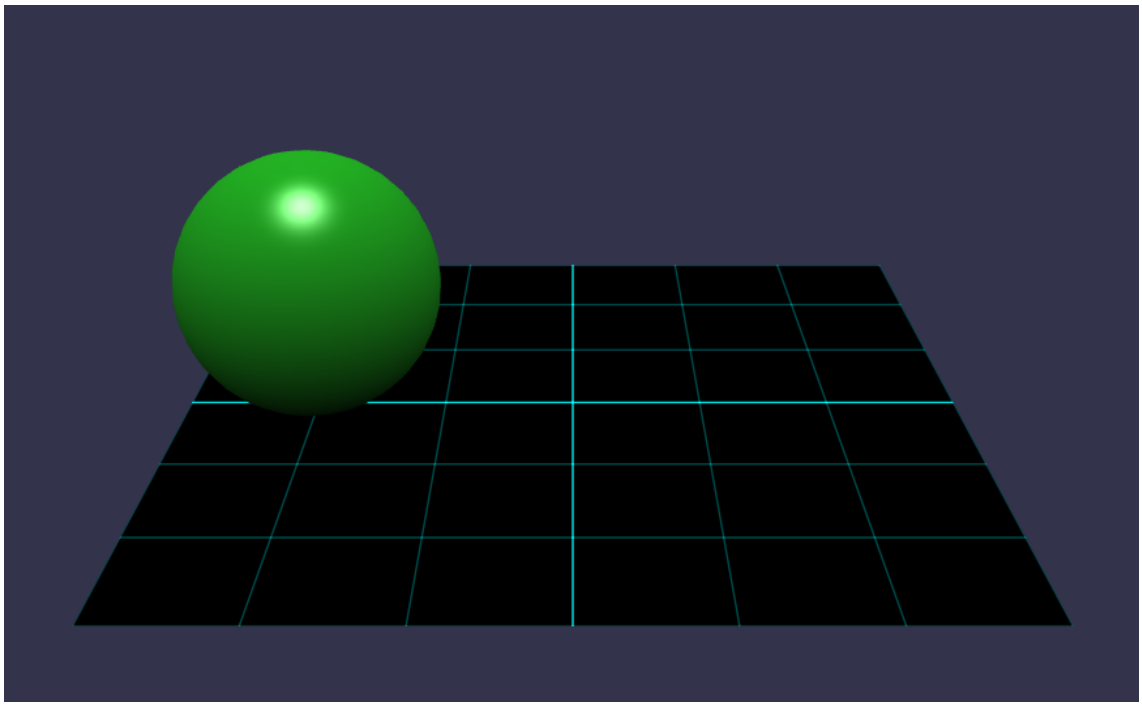
Kuviossa 2 näkyy kaksi Babylon.js:llä luotua mallia, jotka ovat pieni palsta maata ja pallo. Mallien ominaisuuksia, kuten sijaintia ja väriä on helppo muokata. Pallon sijaintia 3D-maailmassa voi muuttaa esimerkiksi korvaamalla index.js-tiedostossa rivillä 43 olevan tekstin *sphere.position.y = 2;* tekstillä *sphere.position = new Vector3(-2, 1, 0);*. Vector3-nimisen luokan rakentajametodi ottaa parametreiksi kolme reaalilukua *x*, *y* ja *z*, jotka kuvastavat mallin sijaintia 3D-maailmassa (Babylon.js 2022d). Pallon väriä voi muuttaa poistamalla samassa tiedostossa rivillä 46 olevan tekstin *sphere.material = material;* ja lisäämällä seuraavat kolme tekstiriviä:

```
var sphereMaterial = new StandardMaterial("sphere", scene);

sphereMaterial.diffuseColor = new Color3(0.2, 1, 0.2);

sphere.material = sphereMaterial;
```

Color3-nimisen luokan rakentajametodi ottaa parametreiksi kolme liukulukua väliltä 0–1, jotka kuvastavat RGB-väriformaatin värien vahvuuksia (Babylon.js 2022e). Muutosten lisäksi Babylon.js:n luokat Color3 ja StandardMaterial pitää ladata ohjelman muistiin. Luokat ladataan muistiin kirjoittamalla tiedoston alkuun tekstirivi *import { Color3, StandardMaterial } from "@babylonjs/core";*. Seuraavassa kuviossa 3 näkyy palloon tehdyt muutokset.



Kuvio 3. Pallon muuttunut väri ja sijainti 3D-maailmassa

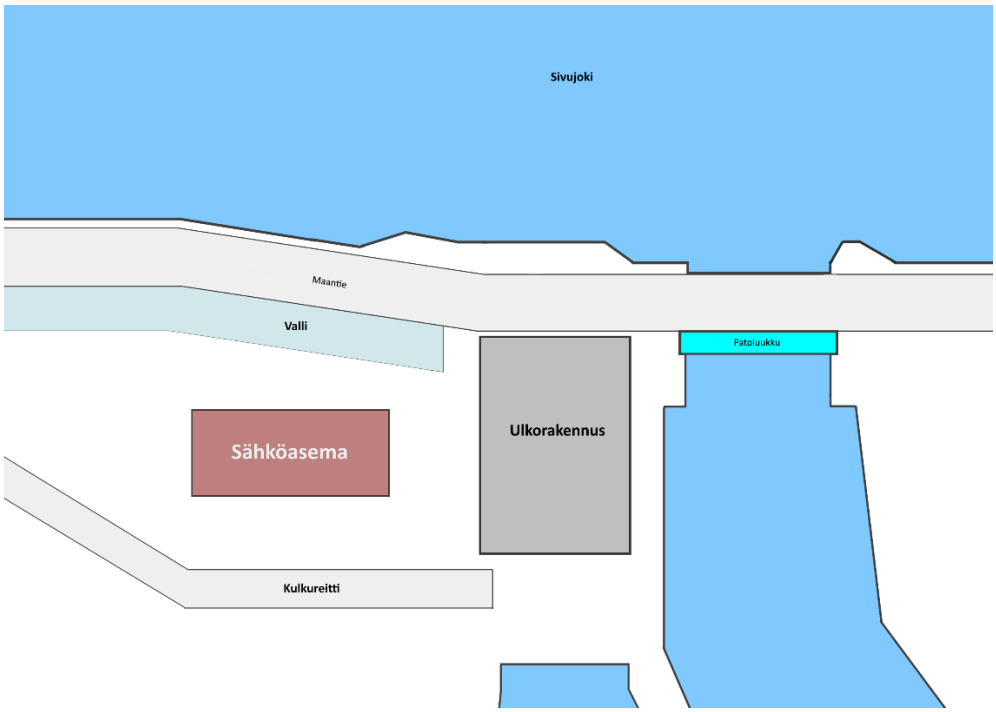
Babylon.js-moottorilla voi luoda yksinkertaisia malleja geometrisista muodoista. Vesivoimalaitoksen mallinnuksessa tarvitaan kuitenkin tarkempaa yksityiskohtaisuuden tasoa, joten mallintaminen toteutetaan Blender-ohjelmalla. Seuraavassa luvussa käydään läpi 3D-mallintamisen suunnittelu- ja toteutusvaiheet. Lisäksi luvussa selvitetään, kuinka Blenderillä luodut mallit saadaan renderöityä ruudulle Babylon.js-moottorin avulla.

4 MALLINNUS

4.1 Yleiskuvaus

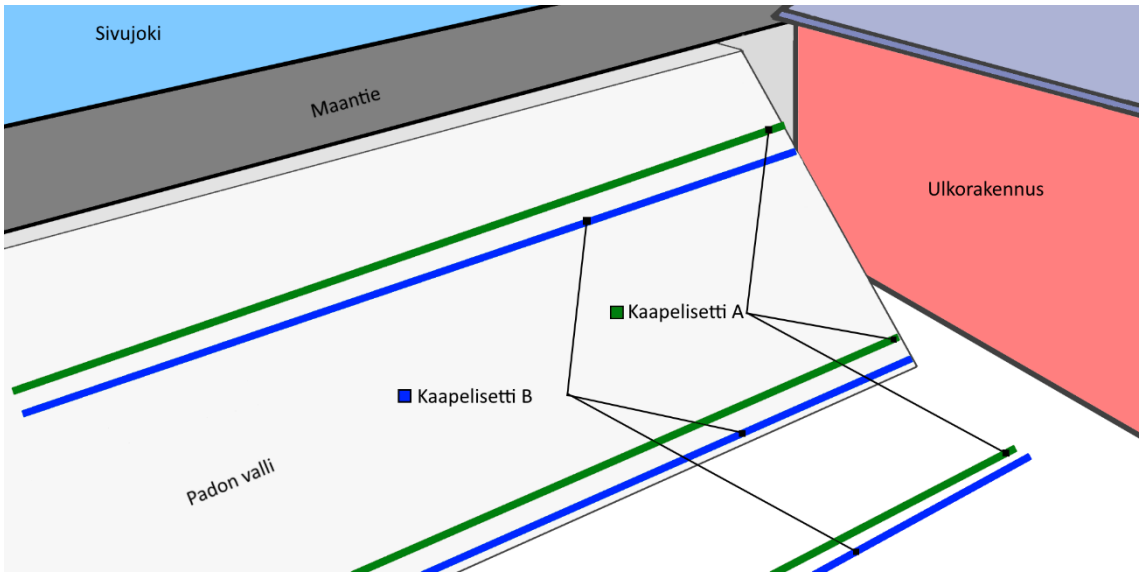
Sovelluksessa käytettävät 3D-mallit suunniteltiin luotavaksi Blender-ohjelmalla. Mallit suunniteltiin olemaan yksinkertaisia alhaisen polygonimäärän malleja. Aikomuksena oli luoda ruudulla esitettävä visuaalinen kokonaiskuva vesivoimalaitoksen lähiympäristöstä 3D-mallien avulla. Visuaalista kokonaiskuvaa varten tarkoituksena oli mallintaa muun muassa anturikaapelit, ulkorakennukset, maantiet, kulkureitit ja padon valli. Jokialue ja muu maasto luotiin proseduraalisesti Babylon.js-moottorilla, jota käsitellään yksityiskohtaisemmin tämän luvun myöhäisemmässä vaiheessa. Tärkeintä lähiympäristön visualisoinnissa on, että loppukäyttäjä pystyy helposti hahmottamaan, missä anturikaapelit sijaitsevat suhteessa vesivoimalaitoksen lähiympäristöön.

Vesivoimalaitoksen lähiympäristöstä on luotu tiedonhaun ja toimeksiantajalta saatujen referenssimateriaalien kautta seuraavassa kuviossa 4 esitetty piirustus. Kuviosta ilmenevät vesivoimalaitoksen lähialueen kokonaiskuva ja mallinnettava miljöö. Kuviossa esitetty ulkorakennus on vesivoimalaitos. Anturikaapelit sijaitsevat padon vallin läheisyydessä maantien ja sähköaseman välissä.



Kuvio 4. Vesivoimalaitoksen lähiympäristö

Seuraavassa kuviossa 5 on esitetty kaapelien sijainnit suhteessa padon valliin. Kuviossa esiintyy kaksi kaapelisettiä, joihin kumpaankin kuuluu kolme anturikaapelia. Vihreällä värjätyt kaapelit kuuluvat A-settiin ja siniset puolestaan B-settiin. Anturikaapelit ovat noin 50 metriä pitkiä. Jotkin kaapeleista ovat maanpinnan yläpuolella ja jotkin puolestaan maan sisällä. Kaapeleihin on kiinnitetty tasaisin välimatkoin lämpötila-antureita, jotka ottavat mittausnäytteitä padon vallista. Lämpötila-antureiden kappalemäärä vaihtelee kaapelikohtaisesti.



Kuvio 5. Kaapeleiden sijainti suhteessa padon valliin

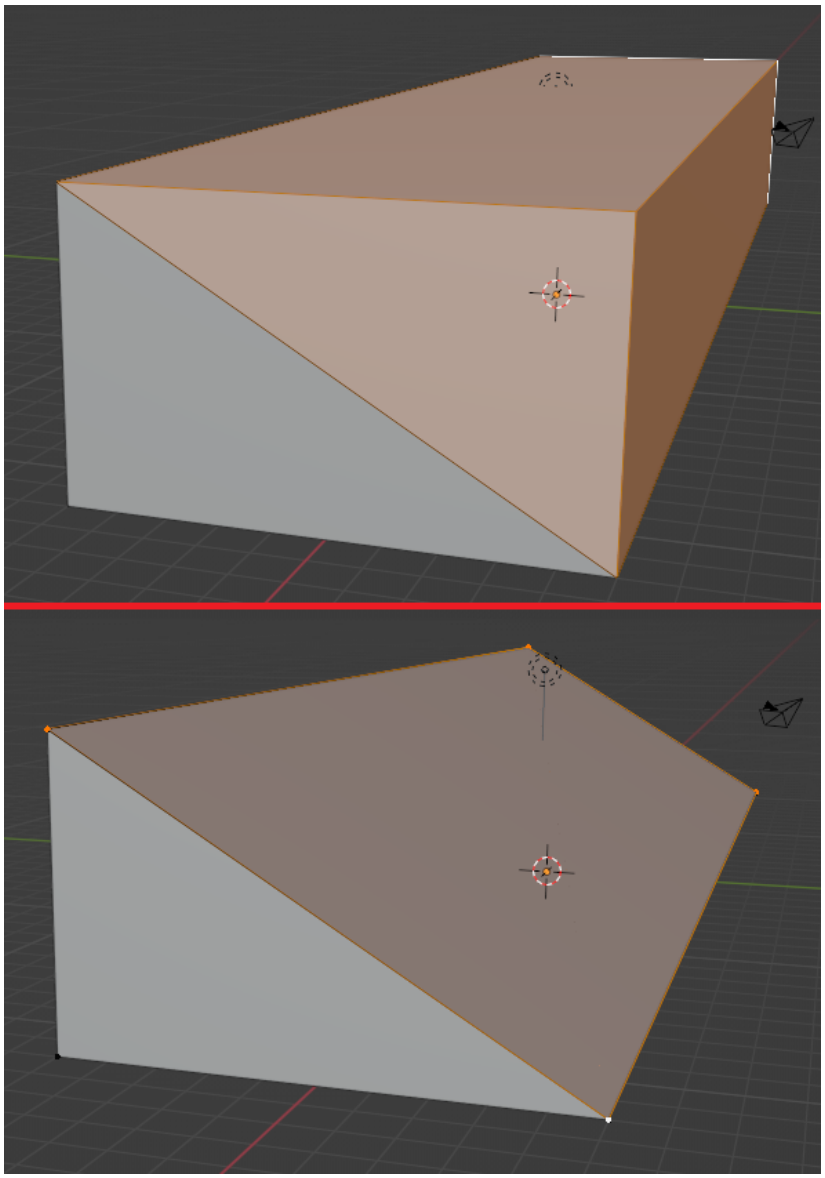
Anturikaapeleiden väri on suunniteltu vaihtumaan senhetkisen anturin tuottaman lämpötilalukeman perusteella kaapelin eri osissa. Anturikaapeleihin kiinnitetään tapahtumankäsittelijä, jonka avulla voidaan muun muassa rekisteröidä loppukäyttäjän hiiren painalluksia kaapelin kohdalla. Nämä asiat käydään läpi tarkemmin tulevissa alaluvuissa.

4.2 3D-mallinnus

3D-mallintaminen voidaan toteuttaa joko Babylon.js-moottorilla tai erillisellä mallintamiseen tarkoitetulla ohjelmalla. Babylon.js:llä voidaan luoda yksinkertaisia geometrisiä malleja, jotka eivät kuitenkaan ole tarpeeksi monipuolisia rakenteeltaan esittämään vesivoimalaitoksen malleja. Tästä syystä 3D-mallinnus toteutettiin Blender-mallinnusohjelmalla. Blender ja Babylon.js tukevat mallien käsittelyä

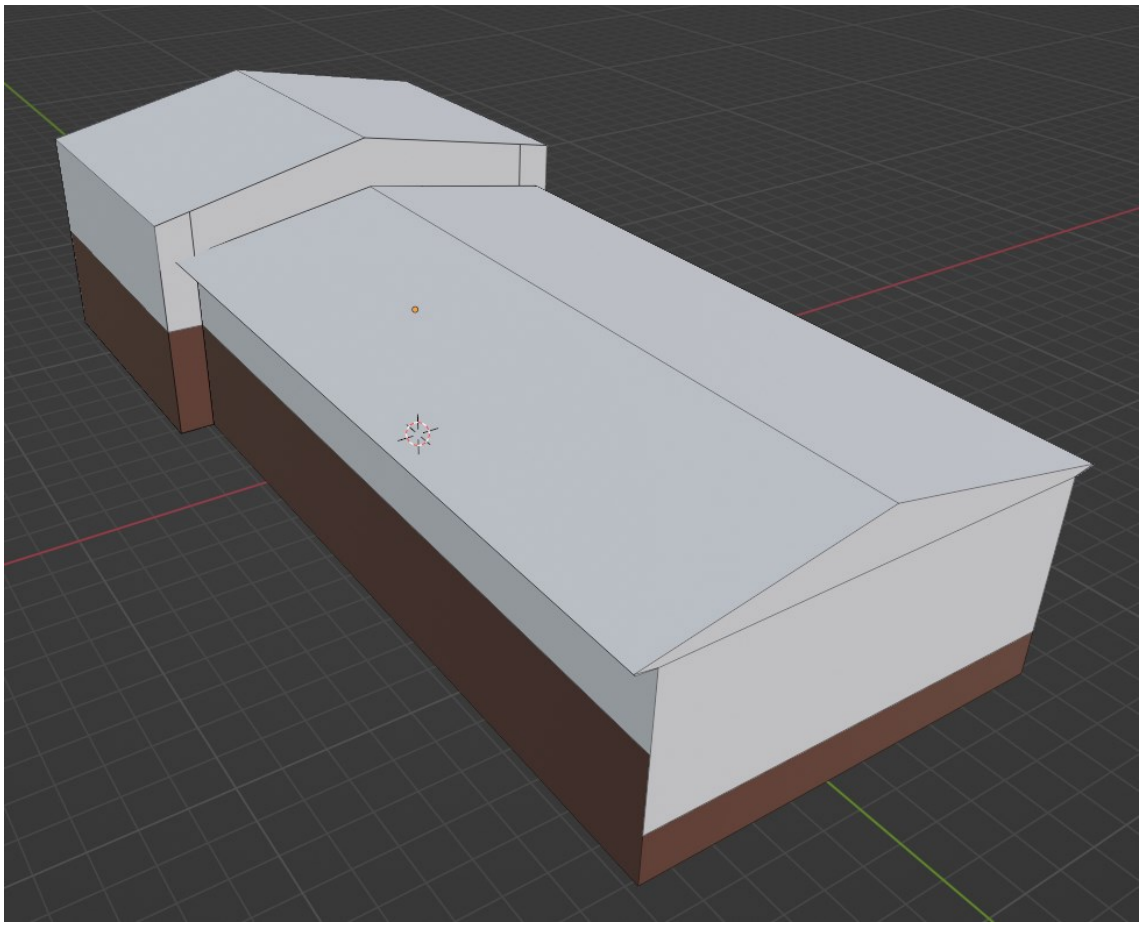
glb-formaatissa. Blenderillä voidaan tallentaa luotu malli glb-formaatissa ja Babylon.js:llä puolestaan ladata tässä formaatissa oleva malli ohjelman muistiin. (Babylon.js 2022f.)

Blenderillä mallintaminen on yksinkertaista. Blender tarjoaa tehokkaita työkaluja, joiden avulla saa veistettyä sopivan muotoisia malleja. Seuraavassa kuviossa 6 on esitetty vesivoimalaitoksen padon vallin mallintaminen. Malli on toteutettu luomalla suorakulmainen särmiö, joka on jaettu kahteen osaan läpileikkaamalla särmiö vasemmasta ylänurkasta oikeaan alanurkkaan. Kuvion yläosassa olevan särmiön oranssilla värillä korostettu osa on poistettu, jolloin lopputuloksena on syntynyt hyvin yksinkertainen malli vesivoimalaitoksen padon vallista.



Kuvio 6. Padon vallin mallinnus

Lähes jokainen vesivoimalaitoksen lähialuetta kuvastava malli on luotu samalla periaatteella eli veistämällä yhdestä tai useammasta särmiön tai kuution muotoisesta kappaleesta mallin kokonaismuoto. Blender tarjoaa myös muita työkaluja mallien muotoiluun, mitä ei kuitenkaan esitetä tässä työssä. Seuraavassa kuviossa 7 on esitetty vesivoimalaitosta kuvastava malli. Blenderin avulla voidaan luoda rakenteeltaan monimutkaisempiakin malleja vaivattomasti, kun puolestaan Babylon.js:ssä tällaisen mallin luominen olisi paljon haastavampaa.



Kuvio 7. Vesivoimalaitoksen malli Blender-ohjelman esikatselussa

Kaikki vesivoimalaitoksen lähialuetta kuvastavat mallit ovat 3D-mallinuksen osalta staattisia tarkoittaen, että mallit eivät ole animoituja. Tämän takia mallintaminen on yksinkertaista ja nopeaa.

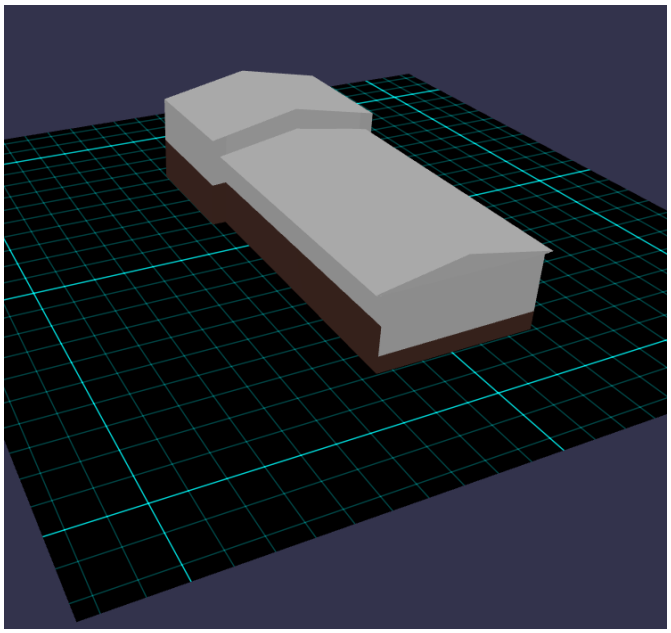
4.3 Mallin lataaminen Babylon.js-moottorilla

Malleja voidaan ladata Babylon.js:n SceneLoader-luokan avulla. SceneLoader-luokka tarjoaa monta eri tapaa ladata erillisiä malleja. Babylon.js tukee mallien

lataamista gtlf-, glb-, obj- ja stl-formaateissa. Tässä työssä käytetyt mallit tallennettiin glb-formaatissa. Babylon.js tarvitsee edellä mainituissa formaateissa olevien mallien käsittelyyn tarkoitetun lisäosan asentamisen ennen mallien lataamista. Lisäosa voidaan asentaa npm-ohjelmalla. Lisäosan nimi on @babylonjs/loaders. Lisäosa ladataan ohjelman muistiin kirjoittamalla index.js-tiedoston alkuun tekstirivi *import '@babylonjs/loaders';*. (Babylon.js 2022g.)

Varsinainen mallin lataaminen voidaan toteuttaa esimerkiksi SceneLoader-luokan ImportMeshAsync-nimisellä funktiolla. Kyseinen funktio ottaa argumenteiksi mallin nimen, tiedostopolun, tiedostonimen ja scene-objektin. Mallin nimi on se nimi, joka on annettu mallille Blender-ohjelmassa. Mallin nimi voi olla myös lista kaikista tiedostossa olevista mallien nimistä. Mallin nimeksi kelpaa myös tyhjä merkkijono, jolloin kaikki tiedostossa esiintyvät mallit ladataan. Tiedostopolku on projektin public-kansiossa sijaitseva polku, jossa mallit sijaitsevat. Tiedostonimi koostuu tiedostolle annetusta nimestä ja käytetystä formaatista, esimerkiksi valli.glb. Scene-objektilla tarkoitetaan Babylon.js-moottorin 3D-maailmaa, jossa kaikki mallit esiintyvät. Scene-argumentti on valinnainen. (Babylon.js 2022g.)

Seuraavassa koodiesimerkissä on havainnollistettu, kuinka ImportMeshAsync-funktiota voidaan käyttää: ”*SceneLoader.ImportMeshAsync(”asema”, ”./meshes/”, ulkorakennus.glb, scene);*”. Seuraavassa kuviossa 8 näkyy tällä funktiolla ladattu vesivoimalaitoksen malli web-selaimessa. Kyseinen funktio on asynkroninen funktio tarkoittaen, että muun ohjelmakoodin suorittaminen ei pysähdy mallien lataamisen ajaksi. Mallia ei pysty myöskään renderöimään ruudulla tai sen ominaisuuksia muokkaamaan, ennen kuin funktio on varmistanut, että malli on onnistuneesti ladattu web-selaimeen. Funktion palautusarvo on Promise-objekti. Promise-objektilla viitataan asynkronisen funktion tilaan ja ladattavan resurssin arvoon. Asynkronisen funktio tila voi olla onnistunut tai epäonnistunut, mikä kertoo, onnistuiko resurssin lataaminen vai ei. Onnistuneen lataamisen jälkeen resurssin arvoon (tässä tapauksessa malliin) pääsee kiinni kutsumalla then-nimistä funktiota pisteoperaattorin avulla. (Babylon.js 2022g; MDN Web Docs 2022a.)



Kuvio 8. Babylon.js-moottorilla ladattu malli web-selaimessa

Seuraavassa koodiesimerkissä on havainnollistettu mallin lataaminen Promise-objektina:

```
const promise = SceneLoader.ImportMeshAsync("asema", "./data/meshes/", "ulkorakennus.glb", scene);

promise.then((result) => {

    const meshBody = result.meshes[0];

    meshBody.position = new Vector3(0, 2, 0);

});
```

Koodiesimerkistä voi huomata, että then-funktio ottaa argumentiksi muuttujan nimen, johon ladattu malli on tallennettu. Babylon.js:ssä ladatun mallin juuri on säilötty meshes-nimiseen listaan, joka löytyy yleensä listan ensimmäisestä indeksistä (0). Ohjelma kutsuu then-funktiota automaattisesti, kun malli on latautunut. Koodiesimerkissä web-selaimeen ladataan vesivoimalaitoksen malli ja se siirretään sen jälkeen 3D-koordinaatteihin x = 0, y = 2 ja z = 0, kun malli on latautunut onnistuneesti. (Babylon.js 2022h.)

4.4 Maaston proseduraalinen generointi

Vesivoimalaitoksen lähiympäristössä esiintyy korkeuseroja jokialueiden ja maaston välillä sekä muita epäsymmetrisyyksiä. Tällaisten erojen havainnollistaminen kaksiulotteisella horisontaalitasoa kuvastavalla kappaleella on hyvin vaikeaa. Tätä ongelmaa varten Babylon.js-moottori tarjoaa ratkaisuksi kolmiulotteisen maaston proseduraalisen generoimisen korkeuskartalla. Korkeuskartta on kuvatiedosto, joka on tallennettu harmaasävykuvana. Harmaasävykuvassa jokaisen pikselin harmaan sävyn voimakkuus esitetään kokonaislukuna välillä 0–255. Lukuarvolla 0 pikseli on täysin musta ja puolestaan lukuarvolla 255 pikseli on täysin valkoinen. (Babylon.js 2022i.)

Korkeuskartta seuraa samaa periaatetta ja muuttaa pikseleiden harmaan sävyn voimakkuuden korkeudeksi. Lukuarvo 0 (musta väri) kuvastaa korkeuden minimiarvoa ja lukuarvo 255 (valkoinen väri) kuvastaa puolestaan korkeuden maksimiarvoa. Seuraavassa kuviossa 9 on esitetty projektissa käytetty korkeuskartta. Korkeuskartta on piirretty vesivoimalaitoksen lähiympäristöstä. Mustalla näkyvät kohdat kuvastavat jokialuetta ja harmaalla näkyvät kohdat puolestaan maata. Valkoisella näkyvät kohdat esittävät padon harjaa, jonka päällä ajotie kulkee. Valkoiset alueet jakava musta syvänne on se kohta, jossa patoluukku sijaitsee ajotien suuntaisesti. (Babylon.js 2022j.)

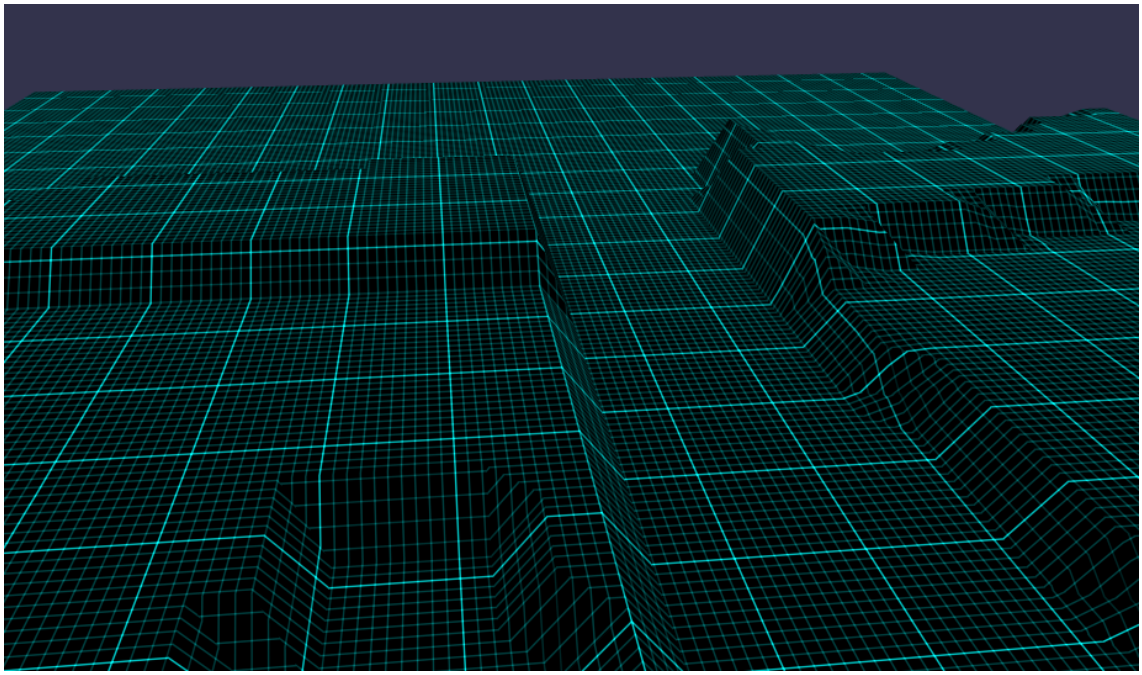
Kuvio 9. Korkeuskartta vesivoimalaitoksen ympäristöstä

Maasto luodaan korkeuskartasta Babylon.js:n MeshBuilder-luokan CreateGroundFromHeightMap-funktiolla. Funktio ottaa argumenteiksi mallin nimen, tiedostopolun ja options-nimisen JavaScript-objektin, joka sisältää valinnaisia parametrejä, kuten kartan pituuden ja leveyden sekä korkeuden minimi- ja maksimiarvot. Ennen MeshBuilder-luokan käyttämistä se tulee ladata ohjelman muistiin koodirivillä *import { MeshBuilder } from "@babylonjs/core";*. Maasto voidaan generoida seuraavalla koodiesimerkillä:

```
const options = { width: 192, height: 192, subdivisions:50, minHeight: 0,
maxHeight: 10 }

const ground = MeshBuilder.CreateGroundFromHeightMap("kartta",
"./data/kartta.png", options, scene);
```

Options-objektissa oleva subdivisions-parametri kertoo mallin polygonimäärän. Mitä isompi lukema on, sitä korkealaatuisempi luotu malli on. Korkea polygonimäärä vaatii enemmän laskentatehoa. Seuraavassa kuviossa 10 voidaan nähdä proseduraalisesti generoitu vesivoimalaitoksen ympäristö. (Babylon.js 2022j.)



Kuvio 10. Babylon.js-moottorilla proseduraalisesti generoitu maasto

Kun maasto on saatu generoitua, seuraavana tehtävänä mallintamisessa on enää sijoittaa kaikki Blenderillä luodut mallit paikoilleen ja lisätä maastolle sitä kuvasta tekstuuri. Nämä asiat ovat helppo toteuttaa Babylon.js-moottorilla.

4.5 3D-maailman kokonaisuuden rakentaminen malleilla

Seuraavaksi maastoa kuvaavalle mallille luotiin tekstuuri, joka ladattiin erillisestä tiedostosta. Helpoin tapa luoda tekstuuritiedosto on kopioida korkeuskartan kuvatiedosto ja muuttaa kopioitu kuva värikuvaksi. Kopioidussa kuvassa ääriviivat ovat jo valmiina paikoillaan, niin tehtäväksi jää enää kuvan värjääminen eli miettiä mitkä värit kuvaavat maastoa parhaiten. Esimerkiksi jokialueet voidaan värjätä siniseksi, maasto vihreäksi ja maantie harmaaksi. Seuraavassa kuviossa 11 näkyy tällä menettelyllä tehty tekstuuritiedosto.

Kuvio 11. Maastoa kuvaava tekstuuritiedosto

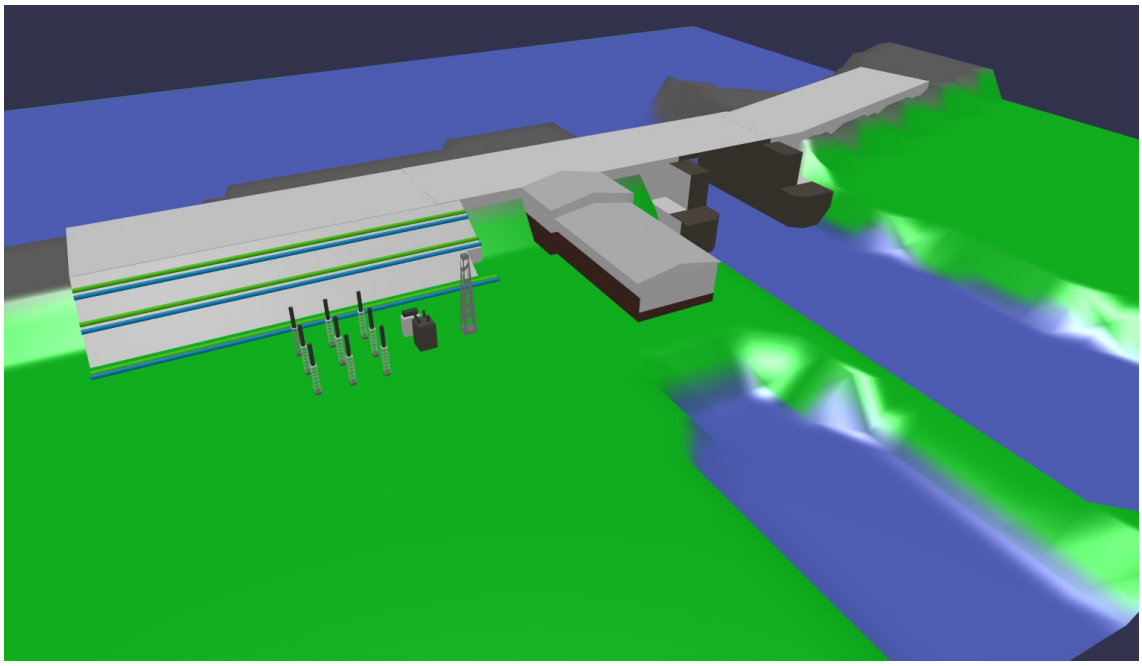
Ennen tekstuurin lisäämistä malliin Babylon.js:n *Texture*-luokka pitää ladata ohjelman muistiin koodirivillä *import { Texture } from "@babylonjs/core";*. Tekstuuri voidaan lisätä malliin koodiesimerkillä:

```
const material = new StandardMaterial("ground", scene);

material.diffuseTexture = new Texture("./data/tekstuuri.png", scene);

ground.material = material;
```

Seuraavaksi vaiheeksi jää enää mallien lisääminen Babylon.js-moottorilla tehtyyn 3D-maailmaan. Mallien metatiedoista rakennetaan JavaScript-objekti, johon tallennetaan muun muassa mallien nimet, tiedostopolut ja sijainnit 3D-koordinaatistossa. Mallit ladataan yksinkertaisen toistorakenteen avulla käyttäen aikaisemmin mainittua ImportMeshAsync-funktiota. Tällä menettelyllä mallit saadaan vaivattomasti ladattua ruudulle ja mallien ominaisuuksien muokkaaminen jälkikäteen on nopeaa. Seuraavassa kuviossa 12 näkyy web-selaimessa esitetty Babylon.js-moottorin 3D-maailma, johon Blenderillä luodut mallit ladattiin. Kuvasta voidaan myös havaita maaston uusi tekstuuri.



Kuvio 12. Vesivoimalaitoksen lähiympäristön mallinnus

Kuviossa 12 on esitetty mallien avulla muun muassa ajotie, padon valli, sähköasema, vesivoimalaitos ja patoluukku. Anturikaapeleiden 3D-mallit voidaan nähdä olevan vasten padon vallia. Ne näyttävät sinisiltä ja vihreiltä lieriöiden muotoisilta kappaleilta. Vesivoimalaitoksen lähiympäristöstä on siis saatu aikaiseksi web-selaimessa toimiva interaktiivinen 3D-mallinnus, josta näkyy anturikaapeleiden sijainti suhteessa lähiympäristöön. 3D-mallinnuksen vaihe on saatu valmiiksi, pois lukien mallien ja tekstuurimateriaalien parantelu ja viimeistely, jotka voidaan jättää myöhemmälle vaiheelle.

Projektin seuraavana vaiheena on hakea lämpötilatietoa rajapinnan kautta ja visualisoida se 3D-mallinnuksessa sekä kuvailla lämpötilatiedon haun ohjelmallista toteutusta. Tarkoituksena on värjätä anturikaapelimallit tuoreimman lämpötilalukeman perusteella ja luoda sovellukseen käyttöliittymä. Käyttöliittymässä esitetään esimerkiksi sivupaneeli, jossa loppukäyttäjä pääsee näkemään tarkempaa lämpötilatietoa anturikaapelista. Sivupaneeli aukeaa, kun loppukäyttäjä klikkaa hiirellä ruudulla olevaa anturikaapelia. Edellä mainitut asiat käydään läpi seuraavassa luvussa.

5 LÄMPÖTILATIEDON PROSESSOINTI

5.1 Kuvaus

Vesivoimalaitoksen anturikaapeleiden lämpötilatieto välittyy FrostBitin palvelimelle. Anturikaapeleissa on kiinni useita lämpötila-antureita, jotka tuottavat lämpötilatietoa. Lämpötilatieto koostuu anturin lämpötilalukemasta ja päivämääräleimasta. Lämpötilatieto on haettavissa tavallisella GET-kutsulla rajapinnan kautta, mitä voi hakea kaapeli- tai anturikohtaisesti tunnistetiedon perusteella. GET-kutsuun voi lisätä parametreiksi päivämäärävälin, jonka ajalta anturitietoa haetaan. Rajapintaa ei esitellä tätä yksityiskohtaisemmin tässä työssä.

Tässä luvussa kerrotaan, kuinka lämpötilatietoa haetaan palvelimelta, kuinka sitä käsitellään ohjelmallisesti ja kuinka sitä esitetään sovelluksessa. Luvussa esitetään myös toteutuksen aikana ilmenneitä haasteita ja niiden ratkaisuja.

5.2 Lämpötilatiedon automaattinen päivitys

Lämpötilatietoa on aikomuksena esittää sivupaneelissa, joka aukeaa, kun käyttäjä klikkaa hiirellä 3D-maailmassa esiintyvää anturikaapelia. Lämpötilatietoa esitetään myös graafisessa esityksessä, jonka toteutus käydään läpi seuraavassa luvussa. Lisäksi 3D-maailmassa olevien anturikaapeleiden väri on suunniteltu päivittymään reaaliajassa. Näiden tarpeiden perusteella lämpötilatiedon kerääminen on päätetty toteuttaa siten, että ohjelma kerää ja säilöö pienen pätkän verran lämpötilatietoa muistissa sekä päivittää sitä automaattisesti.

Lämpötilatiedon keräys on toteutettu siten, että ohjelman käynnistyessä ohjelma hakee lämpötilatietoa rajapinnan kautta jokaisesta anturikaapelista viimeisen 24 tunnin ajalta. Ohjelma säilöö saadun lämpötilatiedon web-selaimen muistiin. Lämpötilatieto vie yhteensä noin 4–5 megatavun edestä tilaa muistissa. Alustavan tiedonhaun jälkeen ohjelma virkistää muistissa olevaa lämpötilatietoa minuutin välein. Tiedon virkistys on toteutettu siten, että ohjelma hakee rajapinnan kautta lämpötilaa viimeiseiltä kymmeneltä minuutilta ja tarkastaa, onko saadun lämpötilatiedon joukossa uutta tietoa saatavilla. Ohjelma tallentaa uuden lämpötilatiedon web-selaimen muistiin vanhan jatkoksi. Ohjelma myös tarkistaa, onko

muistissa yli 24 tuntia vanhaa tietoa, ja jos on, niin ohjelma poistaa vanhentuneen tiedon muistista. Web-selaimessa on siis suoraan saatavilla koko ajan päivitettyä tuoreinta lämpötilatietoa viimeiseiltä 24 tunnilta.

Lämpötilatiedon automaattisella päivityksellä on muutamia hyödyllisiä vaikutuksia. Esimerkiksi käyttäjän klikatessa anturikaapelimallia lämpötilatieto voidaan hakea välittömästi selaimen muistista ja tulostaa ruudulle. Tällä tavoin mahdolliset syöttöviiveet tiedon esittämisessä eliminoituvat kokonaan, kun lämpötilatietoa ei tarvitse hakea erikseen palvelimelta. Tämä vähentää myös rajapintaa ylläpitävään palvelimeen kohdistuvaa verkkoliikenteen määrää. Tämä asia korostuu varsinkin hypoteettisessa tilanteessa, jossa loppukäyttäjä klikkaisi useampaa anturikaapelimallia vuoron perään nopealla tahdilla. Ilman automaattista tiedon päivitystä, jokaisesta hiiren painalluksesta pitäisi tehdä erillinen kutsu palvelimelle, joka kuluttaisi enemmän palvelimen resursseja ja viiveaika nousisi huomattavasti. Toisin sanoen palvelimelta ei ole järkevää syytä hakea samaa tietoa, joka on saatavilla selaimen muistista.

Joka kerta kun uutta lämpötilatietoa on tallennettu selaimen muistiin, ohjelma suorittaa sille määritellyt metodit, jotka tekevät erilaisia toimintoja perustuen uusiin lämpötilatietoihin. Esimerkiksi yksi tällainen metodi on vastuussa 3D-maailmassa olevien anturikaapelimallien värien päivityksestä. Metodi päivittää anturimallien värit vastaamaan tuoreimpia lämpötila-arvoja.

5.3 Lämpötilojen värikoodaus

Anturikaapeleiden lämpötilojen värikoodaus tehtiin luomalla värigradientti, josta laskettiin sopiva väriarvo anturin ilmoittamalle lämpötilalukemalle. Värikoodauksen avulla pyritään havainnollistamaan visuaalisesti anturikaapelissa olevien antureiden tuottamia lämpötilalukemia. Värikoodauksen avulla pitää myös pystyä havaitsemaan pienemmätkin lämpötilalukemien erot. Loppukäyttäjän pitäisi pystyä näkemään esimerkiksi viiden °C:een ero antureissa värien perusteella. Tarkoituksena oli luoda porrastettu värigradientti, jossa esimerkiksi lämpimät lämpötilat näkyisivät punertavilla väreillä ja viileät lämpötilat näkyisivät sinertävillä väreillä.

Värikoodauksen logiikka toteutetaan ohjelmallisesti funktiolla, joka ottaa argumenteiksi antureiden tuottamia lämpötilalukemia ja niitä kuvastavan värigradientin, josta ohjelma laskee kullekin lämpötilalukemalle oikean väriarvon. Väriarvoja käsitellään RGB-väriformaatissa, jossa punaisen, vihreän ja sinisen väriarvot esiintyvät kokonaislukuina välillä 0–255. Väriarvot porrastetaan tietyin välimatkoin lämpötilan perusteella, esimerkiksi lämpötilat 20–25 °C näytetään punasävyisillä väreillä ja lämpötilat 15–20 °C näytetään oranssin sävyisillä väreillä. Porrastetuista lämpötiloista luodaan värigradientti, josta lasketaan väri annetulle lämpötilalle. Seuraavassa kuviossa 13 on esitetty väripaletti suunnitellusta värikoodauksesta eri lämpötiloille.

Lämpötila	RGB-väriarvot	Väri
-20 °C	rgb(255,30,247)	
-15 °C	rgb(163,73,164)	
-10 °C	rgb(63,72,232)	
-5 °C	rgb(0,162,232)	
0 °C	rgb(17,169,156)	
5 °C	rgb(34,177,76)	
10 °C	rgb(181,230,29)	
15 °C	rgb(255,242,0)	
20 °C	rgb(255,127,39)	
25 °C	rgb(237,28,36)	
30 °C	rgb(136,0,21)	

Kuvio 13. Alustava värikoodauksessa käytettävä väripaletti

Värigradientti toteutetaan luomalla funktio, joka ottaa argumenteiksi anturin lämpötilan T, gradienttiportaikon minimi- ja maksimilämpötilat t_0 ja t_1, sekä lämpötiloja kuvaavat minimi- ja maksimiväriarvot c_0 ja c_1. Funktio luo värijanan väriarvoista c_0 ja c_1, josta anturin lämpötilan T väri lasketaan suhteessa lämpötiloihin t_0 ja t_1. Suhteen r saa laskettua kaavalla 1:

$$r = \frac{|T - t0|}{\Delta t} \tag{1}$$

missä

r	on	lämpötilojen T, t_0 ja t_1 välinen suhde
T	on	anturin mittaama lämpötila
$t0$	on	värin minimilämpötila
Δt	on	lämpötilojen t_0 ja t_1 erotus.

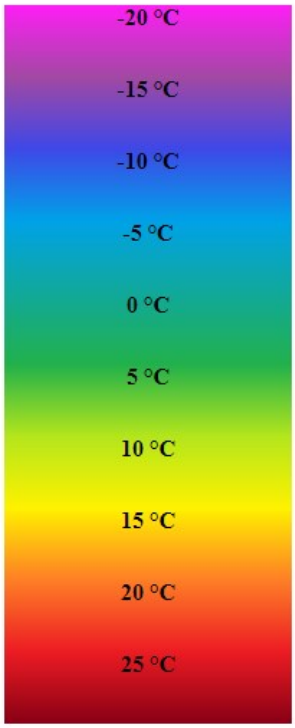
Lämpötilojen suhdetta r käytetään värijanan kertoimena, josta saadaan laskettua oikea väri anturin lämpötilalle T kaavalla 2:

$$C = |c_0 - \Delta c * r| \tag{2}$$

missä

C	on	RGB-värien palautusarvo (0–255)
c_0	on	RGB-värien minimiväriarvo
Δc	on	RGB-värien c_0 ja c_1 erotus
r	on	lämpötilojen T, t_0 ja t_1 välinen suhde.

Esimerkiksi minimi- ja maksimilämpötiloilla t_0 = -5 °C ja t_1 = 0 °C, sekä anturin lämpötilalla T = -2.5 °C suhde r on 0.5 eli anturin lämpötilan T väri sijaitsee tällöin keskellä lämpötiloista c_0 ja c_1 muodostunutta värijanaa. Jos värijanan minimi- ja maksimiväriarvot c_0 ja c_1 ovat (0,0,100) ja (200,200,0), niin C = (100,100,50). RGB-värin arvo C pyöristetään lähimpään kokonaislukuun. Seuraavassa Kuviossa 14 on esitetty yhtenäinen värigradientti useammalle minimi- ja maksimiväriarvoille.



Kuvio 14. Värigradientti, josta anturin väri otetaan lämpötilan perusteella

Tilanteessa, jossa anturin lämpötila osuu suoraan värigradientissa olevan lämpötilan askeleen kohdalle ($T = t_0$ tai $T = t_1$), funktio palauttaa suoraan lämpötilaa t_0 tai t_1 vastaavan väriarvon, sillä mitään laskentaa ei tarvitse tehdä. Tilanteessa, jossa anturin lämpötila ylittää suurimman mahdollisen värigradientissa olevan lämpötilan ($T > t_1$), funktio palauttaa vastaavan lämpötilan t_1 väriarvon. Funktio toimii samalla periaatteella, kun lämpötila on pienempi kuin pienin alhaisin värigradientin lämpötila ($T < t_0$).

5.4 Anturikaapelimallin korjaus

Projektin edetessä tuli huomattua, että anturikaapeleita varten tehdyissä 3D-malleissa esiintyy ongelmia värien muuttamisen osalta. Aikomuksena on muuttaa anturikaapelimallin väriä jokaisen kaapelissa olevan anturin kohdalla lämpötilan perusteella. Ongelmana on, että useamman värin yhtäaikainen näyttäminen yksittäisessä anturikaapelin mallissa ei onnistu Babylon.js-moottorilla, sillä mallissa saa esitettyä vain yhden värin kerrallaan.

Tämä ongelma ratkaistaan luomalla jokaista anturikaapelissa olevaa anturia kohden lieriön muotoinen malli, jotka asetellaan vierekkäin yhdensuuntaisesti siten, että asetelma näyttää yhdeltä kokonaiselta kaapelimallilta. Lieriömallien yhteispituuden tulee olla sama kuin vanhan kaapelimallin pituuden. Yksittäisen lieriömallin pituus saadaan jakamalla antureiden lukumäärä vanhan kaapelimallin pituudella, esimerkiksi jos vanhan kaapelin pituus on 50 metriä ja antureiden lukumäärä on 20, silloin yksittäisen lieriömallin pituus on 50 m / 20 = 2,5 m.

Lieriömallit liikutetaan x-akselilla vanhan kaapelin paikalle 3D-maailmassa. Lieriömallit sijoitetaan paikoilleen vasemmalta oikealle luontijärjestyksessä. Babylon.js-moottori keskittää jokaisen luodun mallin 3D-sijainnin annetuissa koordinaateissa, esimerkiksi jos lieriön pituus on kymmenen metriä, mikä sijaitsee koordinaateissa x = 0, y = 0 ja z = 0, sen vasemmanpuolimmainen pääty on x-akselilla -5. Tämä seikka pitää huomioida laskiessa lieriöiden oikeaa sijaintia x-akselilla. Vanhan kaapelimallin vasemman puolimmaisen päädyn x-akselin arvo voidaan laskea kaavalla 3:

$$cx = x - \frac{cl}{2} \tag{3}$$

missä

cx	on	kaapelin x-akselin vasen pääty
x	on	kaapelin x-akselin sijainti
cl	on	kaapelin pituus.

Jokaisen luodun anturin x-akselin aloitusarvona on vanhan kaapelimallin vasemmanpuoleisen päädyn x-akselin arvo *cx*. Aloitusarvoon lisätään anturin pituus jokaista luotua anturia kohden. Lopuksi jokainen anturi siirretään oikealle puolet oman pituutensa määrästä, minkä tarkoituksena on kumota Babylon.js-moottorin automaattisesti tekemä mallin 3D-sijainnin keskitys. Anturin lieriömallin x-akselin oikea sijainti voidaan siten laskea kaavalla 4:

$$sx = cx + (i - 1) * sl + \frac{sl}{2} \tag{4}$$

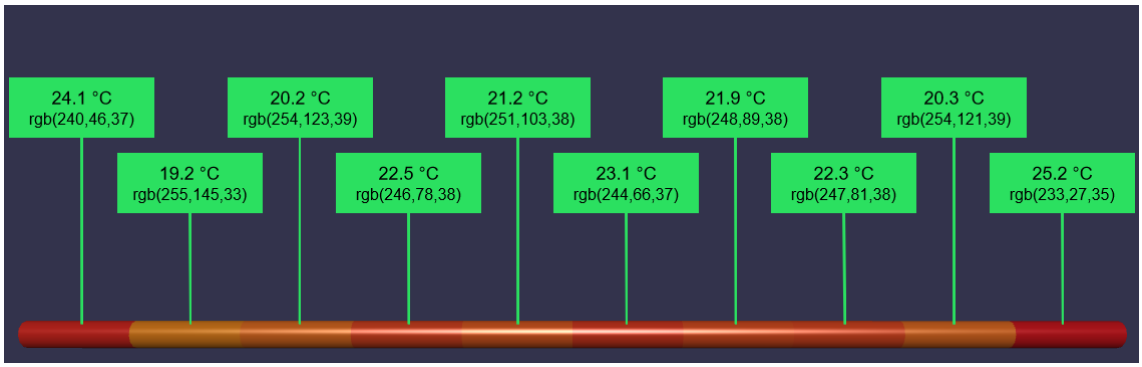
missä

sx	on	anturin x-akselin sijainti
cx	on	kaapelin x-akselin vasen pääty
i	on	anturin järjestysluku luotavien antureiden joukossa
sl	on	anturin pituus.

Alla olevassa kuviossa 15 on esitelty anturimallien luominen Babylon.js-moottorilla. Mallit luotiin MeshBuilder-luokan CreateCylinder-funktiolla. Kuvion yläosassa näkyy viisi luotua anturimallia. Sijoittamalla anturimallit vierekkäin edellisen kaavan avulla anturimallit näyttävät yhdeltä kokonaiselta kaapelimallilta, niin kuin kuvion alaosasta voidaan nähdä. Lopputuloksena on, että jokaiseen anturimalliin voidaan kiinnittää materiaali, jonka väriä voidaan muuttaa vastaamaan anturin ilmoittamaa lämpötilaa.

Kuvio 15. Babylon.js-moottorilla luodut anturimallit

Luotujen anturikaapelimallien toiminnallisuutta on testailtu luomalla jokaiselle anturimallille materiaali, jonka väriä on muutettu edellisessä aliluvussa esitetyn värigradientin perusteella. Seuraavassa kuviossa 16 on esitetty uudet anturimallit, joiden värit on laskettu eri lämpötiloista. Lämpötilat ovat satunnaisesti generoituja desimaalilukuja väliltä 19,0–26,0. Värit vaihtuvat tällä alueella oranssista punaiseen riippuen lämpötilasta.



Kuvio 16. Eri lämpötiloja kuvastavat värit anturimalleissa

Tarkoituksena on, että loppukäyttäjä voi nopeasti saada yleiskuvan kaapelissa olevien antureiden ilmoittamista lämpötiloista ja sekä havaita mahdollisia poikkeamia antureiden kohdalla, jos sellaisia esiintyy. Poikkeamien havainnollistaminen on tärkeää, sillä sen avulla voidaan huomata nopeasti, jos vettä alkaa valumaan anturikaapeleiden päälle.

5.5 Tapahtumankäsittelijän kiinnitys malliin

Jokaiseen anturimalliin kiinnitetään tapahtumankäsittelijä, jonka avulla voidaan havaita erilaisia loppukäyttäjän tekemiä toimintoja, esimerkiksi kun hiiren osoitin osuu anturimallin päälle. Tapahtumankäsittelijä voidaan kiinnittää 3D-malliin Babylon.js-moottorin ActionManager-luokalla, jossa on määritelty tapahtumat ja niiden toiminnot. ActionManager-luokkaan voidaan kiinnittää erikseen ExecuteCodeAction-luokka, joka kutsuu funktion automaattisesti, kun loppukäyttäjä tekee 3D-mallia koskevan määritellyn toiminnon. Funktioon välittyy tieto tapahtumatyypistä, toiminnosta ja 3D-mallista, jota tapahtuma koskee. Funktion avulla voidaan esimerkiksi piirtää ruudulle teksti-ikkuna, jossa lukee anturimallin lämpötilatietoa hiiren osoittimen osuessa mallin päälle. (Babylon.js 2022k; 2022l.)

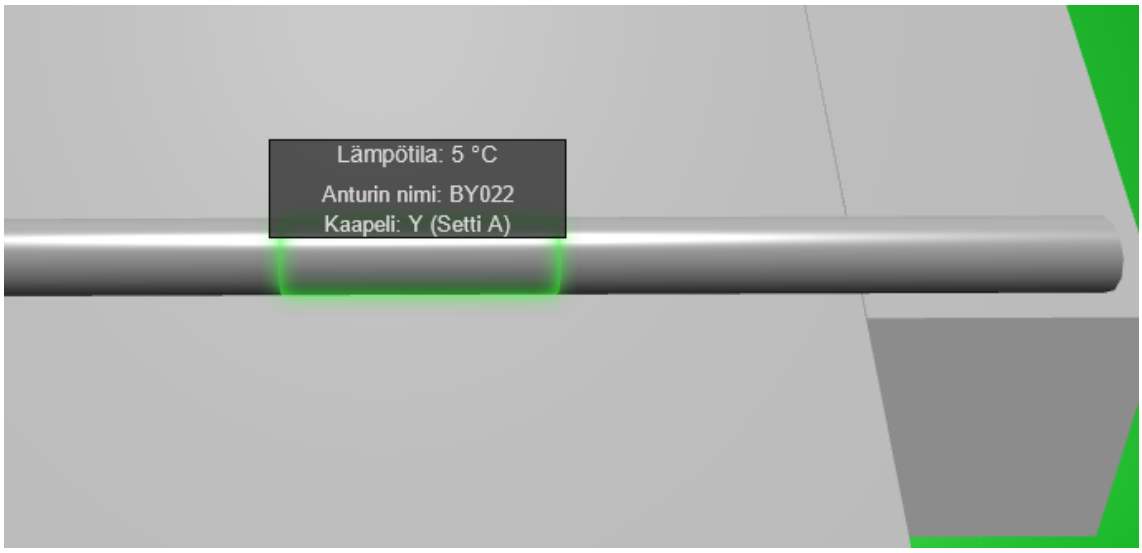
Anturimalleihin on suunniteltu neljä seuraavaa tapahtumaa toteutettavaksi:

- Hiiren osuessa anturimalliin ruudulle piirtyy teksti-ikkuna, jossa lukee anturin ja kaapelin tunnus sekä tuorein lämpötilalukema. Lisäksi anturimallin ulkoreunat värjätään korostusvärillä.

- Hiiren poistuessa anturimallin päältä teksti-ikkuna piilotetaan ja korostusväri poistetaan.

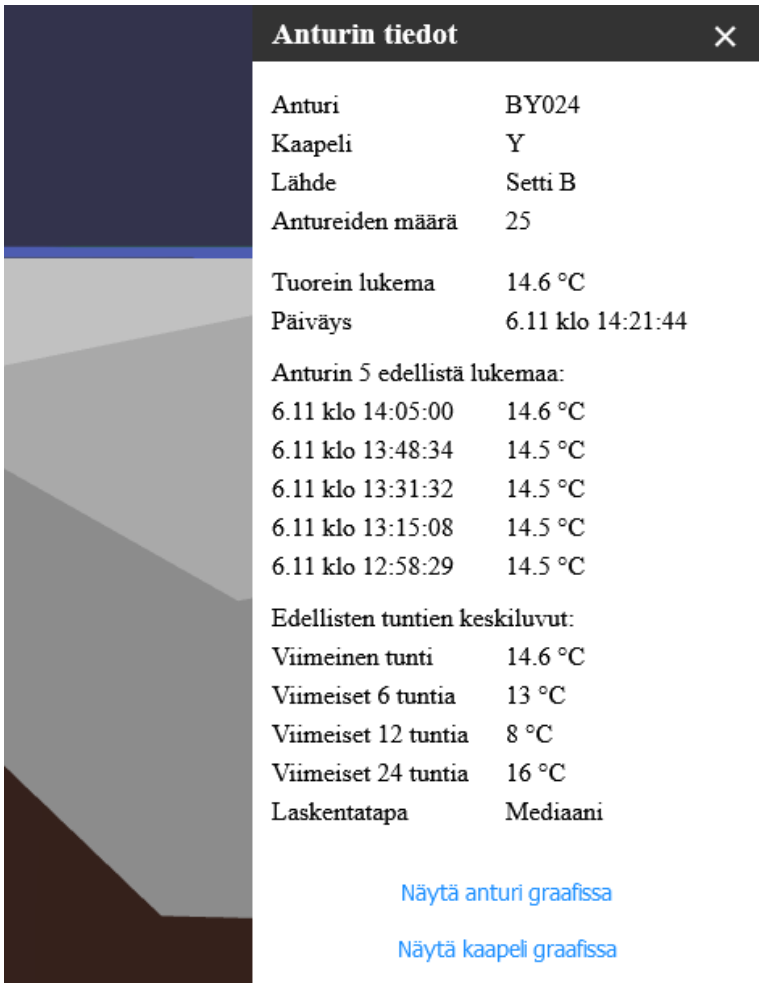
- Käyttäjän klikatessa anturimallia sivupaneeli aukeaa, missä näkyy anturin tunnus, kaapelin tunnus ja lämpötilatiedot.
- Hiiren osuessa kaapelilistassa olevan kaapelin tunnuksen päälle jokaisen anturin yläpuolelle piirtyy tekstilaatikko, jossa näkyvät kunkin anturin tunnus ja tuorein lämpötilalukema.

Seuraavassa kuviossa 17 hiiren osoitin osuu yhden kaapelissa olevan anturimallin päälle, jolloin anturimallin yläpuolelle piirtyy teksti-ikkuna, jossa esiintyy tietoa anturista. Mallin reunat on värjätty korostusvärillä. Hiiren osoittimen kosketus malliin voidaan havaita ActionManager-luokan OnPointerOverTrigger-ominaisuudella. Vastaavasti hiiren osoittimen poistuminen mallin päältä voidaan havaita OnPointerOutTrigger-ominaisuudella. (Babylon.js 2022k.)



Kuvio 17. Teksti-ikkuna piirtyy ruudulle, kun hiiren osoittimen vie anturimallin päälle

Hiiren painalluksesta syntyvät tapahtumat voidaan havaita ActionManager-luokan OnPickTrigger-ominaisuudella (Babylon.js 2022k). Klikattaessa anturimallia ruudun oikeaan laitaan ilmestyy sivupaneeli, jossa on esitetty tietoa klikatusta anturista seuraavan kuvion 18 mukaisesti. Sivupaneelissa esiintyy lämpötilatiedon alapuolella kaksi nappia, joita painamalla anturista tai kaapelista esitetään lisää tietoa graafisessa esityksessä. Graafisen esityksen osuus on käyty läpi yksityiskohtaisesti seuraavassa luvussa.



Kuvio 18. Klikatun anturimallin tietoja sivupaneelissa

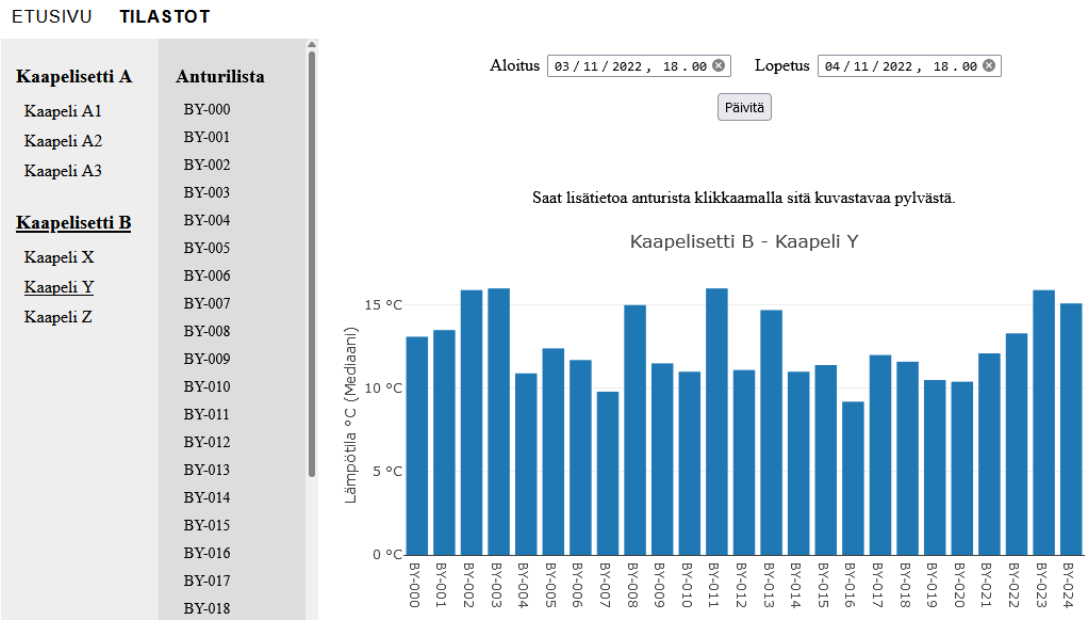
Sovelluksen teknisesti haastavin vaihe on yhdistää palvelimelta saatu lämpötilatieto luotuihin 3D-malleihin. Tapahtumankäsittelijän avulla tämä vaihe on saatu toteutettua onnistuneesti. Palvelimelta saatua lämpötilatietoa voidaan visualisoida mallien avulla, ja sitä voidaan esittää monipuolisesti sovelluksen käyttöliittymän eri ikkunoissa loppukäyttäjän tekemien toimintojen kautta.

6 LÄMPÖTILATIEDON GRAAFINEN ESITYS

6.1 Käyttöliittymän toteutus

3D-mallinnuksen lisäksi vesivoimalaitoksen anturikaapeleiden tuottamat lämpötilatiedot havainnollistetaan graafisessa esityksessä erityyppisinä diagrammiesityksinä. Loppukäyttäjän valitsemasta anturikaapelista luodaan pylväsdiagrammi, jossa esiintyvät jokaisen kaapelissa olevan anturin tunnus ja lämpötilalukeman keskiluku valitulta aikaväliltä. Pylväsdiagrammi mahdollistaa nopean yleiskuvan luomisen anturikaapeleiden lämpötilalukemista. Graafisessa esityksessä on myös mahdollisuus selata yksittäisen anturin lämpötilatietoja annetulta aikaväliltä viivadiagrammissa esitettynä. Diagrammit luodaan plotly.js-nimisellä kirjastolla.

Graafisen esityksen käyttöliittymä toteutetaan luomalla uusi HTML-tiedosto, jossa on määritelty sivun käyttöliittymä seuraavan kuvion 19 tapaisesti. Sivun yläosaan tulee navigointipalkki, josta voi vaihtaa sivun näkymää 3D-maailman ja graafisen esityksen välillä. Vasempaan sivuun piirtyy kaksi sivumenua. Ensimmäisessä sivumenussa loppukäyttäjä voi valita haluamansa anturikaapelin, josta näytetään tietoa pylväsdiagrammissa. Toisessa sivumenussa loppukäyttäjä voi valita anturin, josta esitetään yksityiskohtaisempaa tietoa anturista viivadiagrammissa. Sivun yläosassa keskellä sijaitsee kaksi päivämääräkenttää, joiden ilmoittamalta aikaväliltä lämpötilatietoa haetaan. Sivumenujen ja graafisen esityksen sisältö luodaan dynaamisesti perustuen palvelimelta saatuun tietoon.

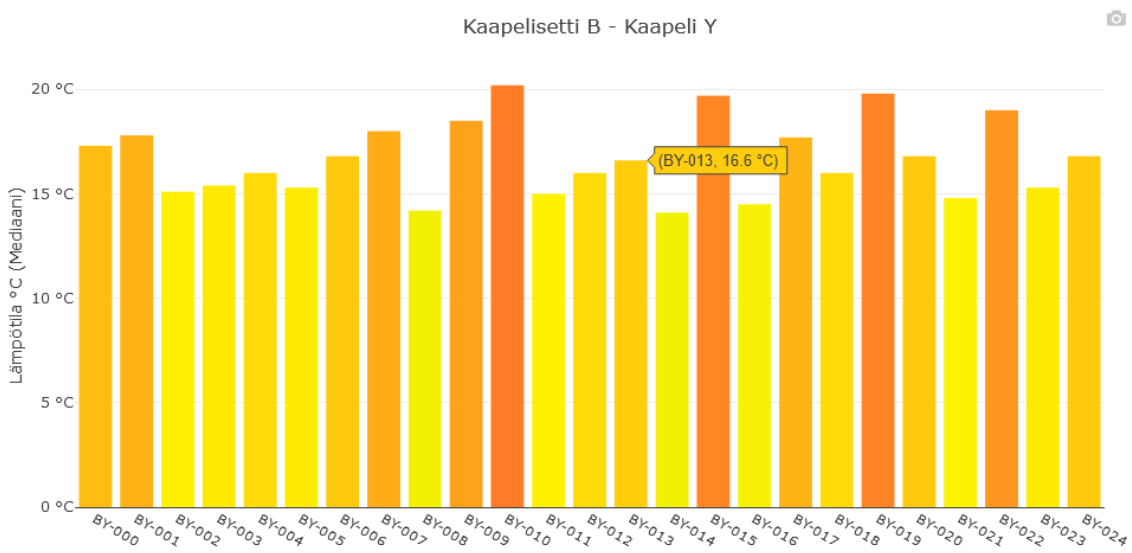


Kuvio 19. Käyttöliittymän ulkoasu graafiselle esitykselle

Loppukäyttäjän painaessa päivämääräkenttien vieressä olevaa hakunappia sovellus päättää kuinka tiedonhaku suoritetaan. Jos päivämäärien ilmoittama aikaväli on alle 24 tuntia nykyhetkestä, sovellus hakee lämpötilatiedon web-selaimen muistista. Jos aikaväli ylittää 24 tuntia nykyhetkestä, sovellus hakee lämpötilatiedon palvelimelta. Lämpötilatietoa on saatavilla selaimen muistista viimeiseltä 24 tunnilta, niin kuin luvussa 5.2 on kuvattu. Ruudulle tulostuu myös virheviestejä, jos sovellus havaitsee annetuissa päivämäärissä epäkohtia.

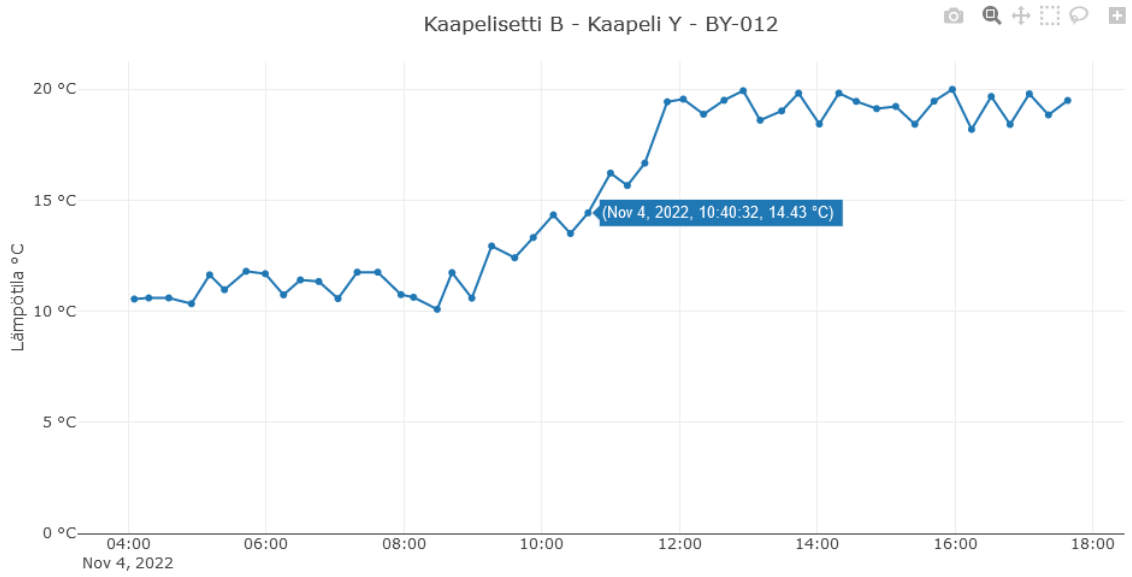
6.2 Lämpötilatiedon esitys diagrammeissa

Anturikaapeleiden lämpötilat kuvataan pylväsdiagrammissa seuraavan kuvion 20 tapaisesti. Diagrammissa näkyvät jokaisen kaapeliin kiinnitetyn anturin tunnus ja lämpötilalukeman keskiluku. Antureiden tunnukset on kuvattu x-akselilla ja lämpötilalukemien keskiluvut on kuvattu y-akselilla. Lämpötilan keskiluku lasketaan niistä lämpötilalukemista, jotka ovat löytyneet annetulta aikaväliltä. Keskiluku muodostuu joko lämpötilalukemien keskiarvosta, mediaanista tai moodista. Keskiluvun laskentatapa on loppukäyttäjän päätettävissä. Jokainen pylväs värikoodataan keskiluvun perusteella. Yksittäisen anturin lämpötilatietoja pääsee tarkastelemaan valitsemalla anturilistasta haluttu anturi tai vaihtoehtoisesti klikkaamalla anturia kuvastavaa pylvästä.



Kuvio 20. Kaapelin lämpötilatiedon esitys pylväsdiagrammissa

Yksittäisen anturin lämpötilatiedot kuvataan viivadiagrammissa seuraavan kuvion 21 tapaisesti. Anturin lämpötilatiedon mittauspäivämäärät on kuvattu x-akselilla ja y-akselilla on kuvattu puolestaan lämpötilalukemat.



Kuvio 21. Yksittäisen anturin lämpötilatiedon esitys viivadiagrammissa

Graafisessa esityksessä on mahdollista selata molemmissa diagrammeissa palvelimella säilöttyä lämpötilatietoa menneiltä kuukausilta siihen päivämäärään asti, kun tiedonkeruu on aloitettu. Lämpötilatietoa voidaan esittää kaapelikohtai-

sesti pidemmältäkin aikaväliltä, sillä antureiden mittaamat useat eri lämpötila-arvot voidaan kiteyttää yhdeksi numeroksi laskemalla lämpötilojen keskiluku, jolloin ne mahtuvat pylväsdiagrammiin esitettäväksi.

7 SOVELLUKSEN VIIMEISTELY

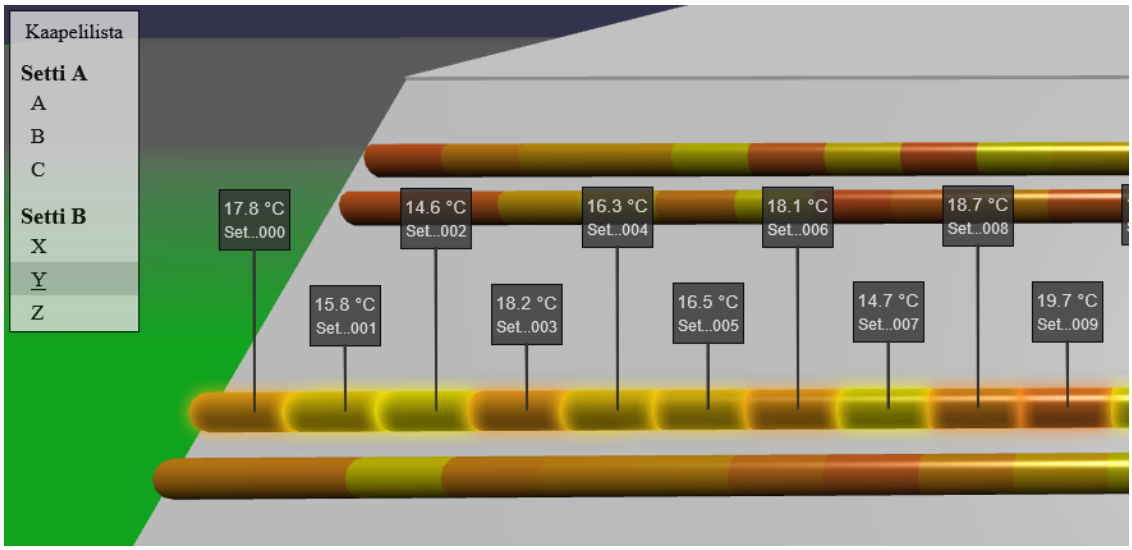
7.1 Yleiskuvaus

Sovelluksessa voidaan esittää palvelimelta saatua lämpötilatietoa 3D-anturimallien ja graafisen esityksen avulla. Anturimalleihin on kiinnitetty Babylon.js:llä toteutettuja tapahtumankäsittelijöitä, joiden avulla voidaan esittää tarkempaa lämpötilatietoa yksittäisestä anturista hiiren tapahtumien kautta.

Sovelluksen toiminnallisuuden kannalta viimeisenä osana on enää luoda kolme käyttöliittymään liittyvää HTML-pohjaista komponenttia, joiden tarkoitus on tehostaa sekä lämpötilatiedon visualisointia että analysointia. Kaavaillut komponentit ovat kaapelilista, animaatiopaneeli ja asetukset-ikkuna. Kaapelilista ja animaatiopaneeli ovat suunniteltu esitettäväksi 3D-maailman näkymässä. Asetukset-ikkunan näyttämistä varten navigointivalikkoon lisätään painonappi, josta ikkuna saadaan esille.

7.2 Viimeisten komponenttien luonti

Kaapelilista on luettelo kaapelilähteiden tunnuksista ja niihin kuuluvista kaapeleiden tunnuksista. Kaapelilista luodaan dynaamisesti palvelimelta haetun tiedon perusteella. Kaapelilistaan liitetään JavaScript-pohjainen tapahtumankäsittelijä, jonka avulla selvitetään, minkä kaapelin tunnuksen päällä hiiren osoitin on. Hiiren osoittimen osuessa kaapelin tunnuksen päälle 3D-maailmassa olevien kaapelin tunnusta vastaavien anturimallien päälle piirtyy teksti-ikkuna, jossa näkyvät antureiden tunnukset ja niiden mittaamat tuoreimmat lämpötilalukemat seuraavan kuvion 22 mukaisesti. Kaapelilista mahdollistaa nopean yleiskuvan saamisen jokaisesta kaapelia kuvastavasta anturimallijoukosta ja niiden senhetkisistä lämpötiloista.

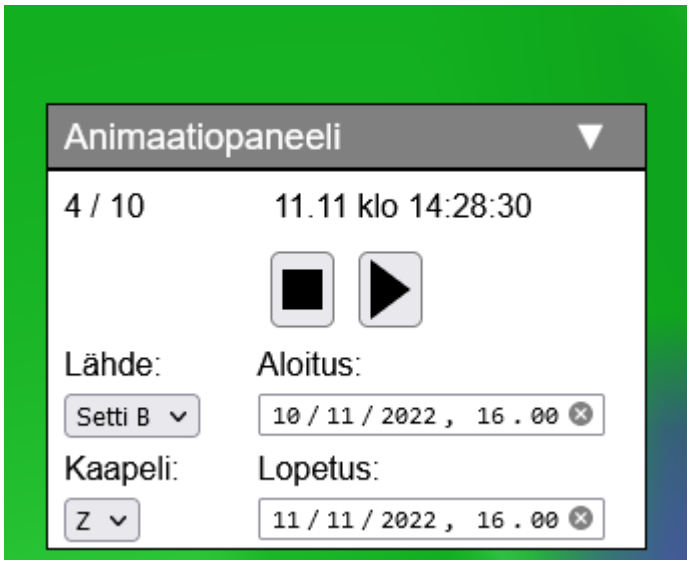


Kuvio 22. Kaapelilistan toiminnallisuuden demonstrointi

Kaapelilista piirtyy vasempaan yläkulmaan 3D-maailman näkymässä. Kaapelilistan ulkoasu on luotu HTML-merkintäkielellä. Anturimallien päälle piirtyvät teksti-ikkunat luotiin Babylon.js:n omalla 2D-käyttöliittymäkirjastolla.

3D-maailman näkymään lisätään myös animaatiopaneeli, jolla voidaan hallita anturimallien lämpötilojen animaatioesitystä. Animaatioesityksessä esitetään nopealla tahdilla vaihtuvia lämpötilalukemia jokaisen anturimallin kohdalla, jotka ovat löytyneet loppukäyttäjän syöttämältä aikaväliltä. Animaatioesityksessä hyödynnetään kaapelilistaa varten tehtyä toiminnallisuutta, jossa lämpötilalukemia esitetään anturimallien yläpuolelle piirtyvissä teksti-ikkunoissa aikaisemman kuvion 22 mukaisesti. Animaation avulla palvelimella säilöttyä lämpötilatietoa voi helposti visualisoida sovelluksen 3D-maailman näkymässä.

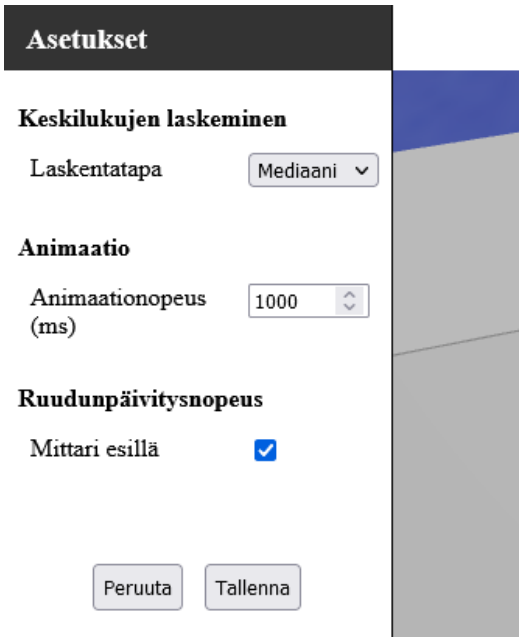
Seuraavassa kuviossa 23 näkyy animaatiopaneelin käyttöliittymän ulkoasu. Animaatiopaneeli sisältää syöttökentät aloitus- ja lopetuspäivämäärille ja sekä kaapelilähteen ja kaapelin tunnuksille. Syöttökenttien yläpuolella on painonapit animaation toistamiselle ja pysäyttämiselle. Painonappien yläpuolella näkyy tietoa animaation aikana senhetkisestä animaation tilasta. Vasemmalla puolella sijaitseva luku ”4/10” ilmaisee, että loppukäyttäjän syöttämältä aikaväliltä on löytynyt kymmenen kappaletta animoitavia lämpötilalukemia, joista 4. lukemaa vastaavat löytyneet lämpötilat on juuri animoitu ruudulle. Oikealla puolella sijaitseva päivämäärä kertoo, milloin ruudulla animoitu lämpötila on mitattu.



Kuvio 23. Animaatiopaneelin käyttöliittymän ulkoasu

Animaation alkaessa sovellus hakee valitun kaapelin lämpötilatiedon annetulta päivämääräväliltä joko web-selaimen muistista tai palvelimelta. Jos päivämääräväli on alle 24 tuntia nykyhetkestä, lämpötilatieto haetaan muistista. Muussa tapauksessa tieto haetaan palvelimelta. Animaatio alustetaan piirtämällä teksti-ikkunat loppukäyttäjän valitseman kaapelin anturimallien päälle, joihin antureiden tunnukset ja niiden mittaamat lämpötilalukemat päivitetään. Animaation aikana jokaisen anturimallin teksti-ikkunaan päivittyy yhden sekunnin välein lämpötilalukema, joka on löytynyt haetusta lämpötilatiedosta. Myös anturimallin väri päivittyy kuvaamaan senhetkistä lämpötilalukemaa. Animaatio kestää niin kauan aikaa, kunnes viimeiset lämpötilalukemat on esitetty. Animoitava lämpötilatieto toistetaan päivämääräjärjestyksessä vanhimmasta uusimpaan.

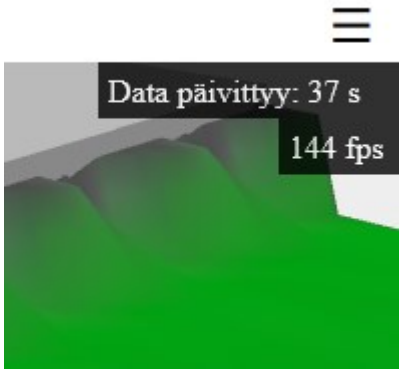
Viimeisenä komponenttina sovellukseen lisätään sivupaneeli sovelluksen asetuksille, joita loppukäyttäjä pystyy muokkaamaan. Navigointipalkkiin lisätään painonappi, josta asetuksien sivupaneeli aukeaa. Seuraavassa kuviossa 24 on esitetty asetuksien sivupaneelin ulkoasu. Asetuksista voi vaihtaa muun muassa lämpötilojen animaatioesityksessä käytettävän animaationopeuden ja lämpötilojen keskiluvun laskentatavan. Keskilukua hyödynnettiin graafisen esityksen pykälädiagrammissa, jossa näytettiin antureiden lämpötilatietoja ja sekä 3D-maailman näkymän sivupaneelissa, jossa näytettiin klikatun anturimallin tietoja. Sovelluksen tuetut keskiluvun laskentatavat ovat keskiarvo, mediaani ja moodi.



Kuvio 24. Asetukset-sivupaneelin ulkoasu

Sovelluksen asetukset tallennetaan avain-arvo-parina käyttäen Web Storage-nimistä rajapintaa. Rajapinta tarjoaa localStorage-ominaisuuden, jonka avulla loppukäyttäjän tekemät muutokset säilyvät sovelluksen muistissa istuntojen välillä (MDN Web Docs 2022b).

Sovelluksen 3D-maailman näkymään on päätetty myös lisätä kaksi mittaria seuraavan kuvion 25 mukaisesti. Mittarit piirtyvät oikeaan yläkulmaan navigointipalkin alapuolelle. Ylimmäinen mittari ilmoittaa, milloin lämpötilatieto päivittyy seuraavan kerran. Lämpötilatietoa päivitetään automaattisesti minuutin välein niin kuin 5. luvun alaluvussa 5.2 on kuvattu. Alimmainen mittari puolestaan kertoo ruudunpäivitysnopeuden, jonka näkyvyyttä voi vaihtaa asetuksista.



Kuvio 25. 3D-näkymän mittarit navigaatiopalkin alapuolella

Tältä osin sovelluksen käyttöliittymän ja ohjelmalogiikan toiminnallisuus on valmis. Viimeisimpinä tehtävinä asioina on enää optimoida 3D-maailman esitystä Babylon.js-moottorilla, testailla ja varmentaa sovelluksen toiminnallisuus Google Chrome-, Mozilla Firefox- ja Microsoft Edge -nimisillä web-selaimilla, sekä parannella käyttöliittymän ulkoasua, 3D-malleja ja tekstuureja. Lisäksi sovellus toimitetaan FrostBitille. Näitä asioita ei kuitenkaan käydä läpi tässä opinnäytetyössä.

8 POHDINTA

Kaikki sovelluksen kriittiset tehtävät saatiin tehdyksi. Sovellukseen välittyy lämpötilatietoa vesivoimalaitokselta FrostBitin palvelimen rajapinnan kautta. Sovelluksella voidaan seurata padon vallin olosuhdetilaa reaaliajassa lämpötilalukemien ohjelmallisen visualisoinnin kautta.

Vesivoimalaitoksen lähiympäristöstä on tehty 3D-mallinnus, jossa näkyy anturikaapeleiden sijainnit. Ruudulla esitetty 3D-mallinnuksen näkymä on interaktiivinen, eli loppukäyttäjä voi hiiren avulla vaihtaa näkymän kuvakulmaa vapaasti ja anturimalleja klikkaamalla ruudulle piirtyy ikkunoita, joissa esiintyy anturin lämpötilatietoja. Palvelimelta haettu anturikaapeleiden lämpötilatieto voidaan esittää monipuolisesti sovelluksessa. Jokaisen anturikaapeleissa olevien antureiden mittaamia lämpötila-arvoja voidaan tarkkailla reaaliajassa 3D-mallinnuksen näkymässä. Sovelluksessa on myös mahdollista etsiä aikaisempaa lämpötilatietoa annetulta päivämääräväliltä ja sitä voidaan esittää visuaalisena animaationa 3D-näkymässä anturimallien avulla tai vaihtoehtoisesti graafisen esityksen pylväs- ja viivadiagrammeissa. Graafisessa esityksessä on mahdollisuus tarkastella lämpötila-antureiden mittaamia tuloksia keskilukuina esitettyinä annetulta päivämääräväliltä. Keskiluvun laskentatapaa voi vaihtaa sovelluksen asetuksista.

Sovelluksen tehtävien osalta tekemättä jäivät keskilukujen poikkeamien havainnointi ja niihin liittyvän varoitusjärjestelmän toteutus. Aikomuksena oli myös luoda käyttöliittymästä responsiivinen versio mobiililaitteille. Sovelluksen kriittisten tehtävien toteuttaminen vei kuitenkin yllättävän paljon aikaa, joten edellä mainittuja asioita ei ehditty yksinkertaisesti toteuttamaan. Käyttöliittymän muuttaminen responsiivikseksi ei olisi kuitenkaan kovin vaikeaa, kun se voitaisiin toteuttaa tekemällä muutoksia sovelluksen CSS-tyylitiedostoon. Varoitusjärjestelmän toiminnallisuuden lisääminen sivustolle on haastavampi ongelma. Se voitaisiin kuitenkin toteuttaa ohjelmallisesti esimerkiksi seuraamalla automaattisesti päivittyviä lämpötila-arvoja ja vertailemalla tuoreimpia arvoja muutamaan edelliseen arvoon. Näiden lukujen joukosta voitaisiin etsiä nopeasti laskevia luku sarjoja jokaisen anturin kohdalla, mikä voisi mahdollisesti johtua veden valumisesta anturin päälle.

Opinnäytetyötä tehdessä tuli opittua paljon uutta asiaa. Minulla ei ollut aikaisempaa kokemusta 3D-mallintamisesta. Blenderin omilta kotisivuilta löytyi kuitenkin reilusti tutoriaalivideoita, joissa käsiteltiin mallintamisen perusteita ja ohjelman käyttöä. Tutoriaalien ja mallintamisen harjoittelun parissa vierähti useampi tunti. Mallintaminen oli kuitenkin helppoa loppujen lopuksi. Samaa voi sanoa Babylon.js-pelimoottorista, josta minulla ei myöskään löytynyt aikaisempaa kokemusta. Babylon.js:n kotisivuilta löytyi kaiken kattava dokumentaatio pelimoottorin ominaisuuksista ja tutoriaaliesimerkkejä, joita pystyi tekemään Playground-editorissa. Näiden materiaalien avulla pelimoottorin perusteiden oppiminen oli helppoa ja nopeaa. Arvioin, että onnistuin tehtävässä kohtuullisen hyvin, vaikkei minulla ollut aikaisempaa asiantuntemusta näistä osa-alueista projektin alkuvaiheilla.

LÄHTEET

Babylon.js 2022a. Babylon.js game engine. Viitattu 17.10.2022 https://www.babylonjs.com/games/.

Babylon.js 2022b. The Playground. Viitattu 18.10.2022 https://doc.babylonjs.com/toolsAndResources/thePlayground.

Babylon.js 2022c. Babylon.js ES6 support with Tree Shaking. Viitattu 18.10.2022 https://doc.babylonjs.com/setup/frameworkPackages/es6Support.

Babylon.js 2022d. Class Vector3. Viitattu 20.10.2022 https://doc.babylonjs.com/typedoc/classes/BABYLON.Vector3.

Babylon.js 2022e. Class Color3. Viitattu 20.10.2022 https://doc.babylonjs.com/typedoc/classes/BABYLON.Color3.

Babylon.js 2022f. Loading Any File Type. Viitattu 21.10.2022 https://doc.babylonjs.com/features/featuresDeepDive/importers/loadingFileTypes.

Babylon.js 2022g. Class SceneLoader. Viitattu 22.10.2022 https://doc.babylonjs.com/typedoc/classes/BABYLON.SceneLoader.

Babylon.js 2022h. Getting Started - Chapter 1 - Working with Models. Viitattu 23.10.2022 https://doc.babylonjs.com/features/introductionToFeatures/chap1/first_import.

Babylon.js 2022i. Creating Ground From a Height Map. Viitattu 23.10.2022 https://doc.babylonjs.com/features/features-DeepDive/mesh/creation/set/ground_hmap.

Babylon.js 2022j. More On Height Maps. Viitattu 23.10.2022 https://doc.babylonjs.com/features/featuresDeepDive/mesh/creation/set/height_map.

Babylon.js 2022k. Class ActionManager. Viitattu 30.10.2022 https://doc.babylonjs.com/typedoc/classes/BABYLON.ActionManager.

Babylon.js 2022l. Class ExecuteCodeAction. Viitattu 30.10.2022 https://doc.babylonjs.com/typedoc/classes/BABYLON.ExecuteCodeAction.

MDN Web Docs 2022a. Promise. Viitattu 23.10.2022 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise.

MDN Web Docs 2022b. Web Storage API. Viitattu 18.11.2022 https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API.

Treehouse 2022. Installing Node.js® and NPM on Windows. Viitattu 18.10.2022 https://treehouse.github.io/installation-guides/windows/node-windows.html.

LIITTEET

Liite 1. Babylon.js, index.html-tiedoston sisältö

Liite 2. Babylon.js, index.js-tiedoston sisältö

Liite 3. Webpack-konfiguraatiotiedosto, webpack.config.js-tiedoston sisältö

Liite 1. Babylon.js, *index.html*-tiedoston sisältö.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Local Development</title>

    <script
src="https://code.jquery.com/pep/0.4.2/pep.min.js"></script>

    <style>
      html,
      body {
        width: 100%;
        height: 100%;
        padding: 0;
        margin: 0;
        overflow: hidden;
      }

      #renderCanvas {
        width: 100%;
        height: 100%;
        display: block;
        font-size: 0;
      }
    </style>
  </head>

  <body>
    <canvas id="renderCanvas" touch-action="none"></canvas>

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

Liitteessä esiintyy *index.html*-tiedoston sisältö, joka mainittiin opinnäytetyön 3. luvun 2. alaluvussa kohdassa Paikallinen asennus. Sisältö on kopioitu Babylon.js:n kotisivujen osoitteesta https://doc.babylonjs.com/setup/frameworkPackages/es6Support.

Liite 2. Babylon.js, *index.js*-tiedoston sisältö.

```
import { FreeCamera } from "@babylonjs/core/Cameras/freeCamera";
import { Engine } from "@babylonjs/core/Engines/engine";
import { HemisphericLight } from "@babylonjs/core/Lights/hemispher-
icLight";
import { Vector3 } from "@babylonjs/core/Maths/math.vector";
import { CreateGround } from "@babylonjs/core/Meshes/Build-
ers/groundBuilder";
import { CreateSphere } from "@babylonjs/core/Meshes/Build-
ers/sphereBuilder";
import { Scene } from "@babylonjs/core/scene";

import { GridMaterial } from "@babylonjs/materials/grid/gridMaterial";

// Get the canvas element from the DOM.
const canvas = document.getElementById("renderCanvas");

// Associate a Babylon Engine to it.
const engine = new Engine(canvas);

// Create our first scene.
var scene = new Scene(engine);

// This creates and positions a free camera (non-mesh)
var camera = new FreeCamera("camera1", new Vector3(0, 5, -10), scene);

// This targets the camera to scene origin
camera.setTarget(Vector3.Zero());

// This attaches the camera to the canvas
camera.attachControl(canvas, true);

// This creates a light, aiming 0,1,0 - to the sky (non-mesh)
var light = new HemisphericLight("light1", new Vector3(0, 1, 0),
scene);

// Default intensity is 1. Let's dim the light a small amount
light.intensity = 0.7;

// Create a grid material
var material = new GridMaterial("grid", scene);

// Our built-in 'sphere' shape.
var sphere = CreateSphere("sphere1", { segments: 16, diameter: 2 },
scene);

// Move the sphere upward 1/2 its height
sphere.position.y = 2;

// Affect a material
sphere.material = material;

// Our built-in 'ground' shape.
var ground = CreateGround("ground1", { width: 6, height: 6,
subdivisions: 2 }, scene);

// Affect a material
ground.material = material;

// Render every frame
```

```
engine.runRenderLoop(() => {
  scene.render();
});
```

Liitteessä esiintyy *index.js*-tiedoston sisältö, joka mainittiin opinnäytetyön 3. luvun 2. alaluvussa kohdassa Paikallinen asennus. Sisältö on kopioitu Babylon.js:n kotisivujen osoitteesta https://doc.babylonjs.com/setup/frameworkPackages/es6Support.

Liite 3. Webpack-konfiguraatiotiedosto, *webpack.config.js*-tiedoston sisältö.

```
const path = require('path');

module.exports = {
    mode: 'development',
    devtool: "inline-source-map",
    entry: './src/index.js'
};
```

Liitteessä esiintyy *webpack.config.js*-tiedoston sisältö, joka mainittiin opinnäytetyön 3. luvun 2. alaluvussa kohdassa Paikallinen asennus.