SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
KULTTUURIALA

ATERIMEN VÄRIN VALINTASOVELLUKSEN TOTEUTUS THREE.JS:LLÄ

Interaktiivinen websovellus teollisen muotoilun tuotteen esittelyyn

TEKIJÄ: Irene Pöllänen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Kulttuuriala	
Tutkinto-ohjelma Muotoilun tutkinto-ohjelma	
Työn tekijä Irene Pöllänen	
Työn nimi Aterimen värin valintasovelluksen toteutus Three.js:llä	
Päiväys 23.5.2021	Sivumäärä/Liitteet 26/1
Toimeksiantaja/Yhteistyökumppani(t) -	

Tiivistelmä

Tässä opinnäytteessä toteutettiin interaktiivinen aterimen värin valintasovellus Three.js:llä. Three.js on JavaScript 3D-kirjasto ja API (application programming interface), jolla voidaan luoda ja esittää 3D- tietokonegrafiikkaa selaimessa käyttäen WebGL:ää. Modernit selaimet tukevat WebGL:n käyttöä, joten käyttäjän ei tarvitse asentaa mitään ylimääräisiä selainlisäosia kirjaston käyttämiseksi. Three.js hyödyntää näytönohjainta, joten sillä on mahdollista visualisoida erityisen näyttävää ja sulavaa 3D-grafiikkaa selaimessa.

Työn lähtökohtana oli aiemmilla kursseilla Rhinoceroksella mallinnettu haarukkamalli. Malli muokattiin soveltuvaksi websovellukseen Blenderillä ja mallin ympärille luotiin websovellus, missä asiakas voi valita tuotteelle minkä tahansa värin RGB-väriavaruudesta. Tässä työssä hyödynnetyt ohjelmat ovat ilmaisia ja avoimen lähdekoodin tuotteita.

Opinnäytetyön tavoitteena oli omaksua Three.js:n perusteet ja luoda jokin lopullinen sovellus, joka visualisoisi dataa, jossa olisi jokin toiminto tai joka olisi visuaalisesti mielenkiintoinen. Työtä toteutettiin kokeilun ja yrittämisen kautta. Alun haasteena oli oppia kuinka Three.js:llä luodaan webgrafiikkaa. Tekniikka ei ollut aiemmin tuttu, joten työtä aloittaessa oli haastavaa päättää tarkkaan etukäteen mitä lopullinen lopputulos tulisi olemaan.

Lopulta onnistuttiin luomaan toiminnallinen sovellus, joka julkaistiin webpalvelimelle. Täten tässä opinnäytteessä luotu sovellus on testattu myös varsinaisessa julkaisuympäristössään internetsivuilla. Sovellusta voisi kehittää vielä eteenpäin. Haarukan liikerataa voisi rajoittaa ja käyttöliittymää voisi tehdä vielä selkeämmäksi. Nyt tavoite oli luoda toiminnallisuudet prototyyppitasolla siten, että ohjelma ei kaadu ja olisi visuaalisesti uskottava.

Avainsanat

Three.js, WebGL, 3D-mallinnus, websovellus

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Culture	
Degree Programme Degree Programme in Design	
Author Irene Pöllänen	
Title of Thesis Creating a Web Application with Three.js for Choosing a Colour of Cutlery with Three.js	
Date 23 April 2021	Pages/Appendices 26/1
Client Organization /Partners	

Abstract

In this thesis an interactive web application was implemented with Three.js for choosing a colour of cutlery. Three.js is a JavaScript 3D-library and API (application programming interface) with which one can create 3D-computer graphics for browser using WebGL. Modern browsers support WebGL, so there is no need to install any extra add-ons.

The goal of this thesis was to learn the basics of Three.js and create an application which would visualize data, have a functionality or would be visually interesting. There was much trial and error during the process. The first challenge was to learn how to build web graphics with Three.js. The technology was not familiar in the beginning, so it was difficult to define the final product at the start.

The final idea for the application raised from a fork model that was modelled earlier during studies. An interactive web application was made with which a customer can choose whatever colour from the RGB colour space she or he wants for the cutlery. The original fork model was modified with Blender to be more suitable for a web application. Open-source programs were used in the execution.

As a result, a fully functional application was made and published on a web server. The result is still rough and could be fine-tuned. The motion path of the fork could be more constrained, and the user interface could still be made clearer. Nevertheless, the goal was achieved by producing a working prototype which has the functionality and is somewhat visually convincing.

Keywords
Three.js, WebGL, 3D-modeling, web application

SISÄLTÖ

KUVALUETTELO

1 JOHDANTO

Tässä opinnäytteessä tavoite on perehtyä ja oppia hyödyntämään Three.js JavaScript 3D-kirjastoa rakentamalla jonkinlainen teollisen muotoilun opintoja soveltava interaktiivinen teos. Lopputulos on aterimen värivalintasovellus, jonka toteutuksen vaiheet kuvataan tässä opinnäytetyössä.

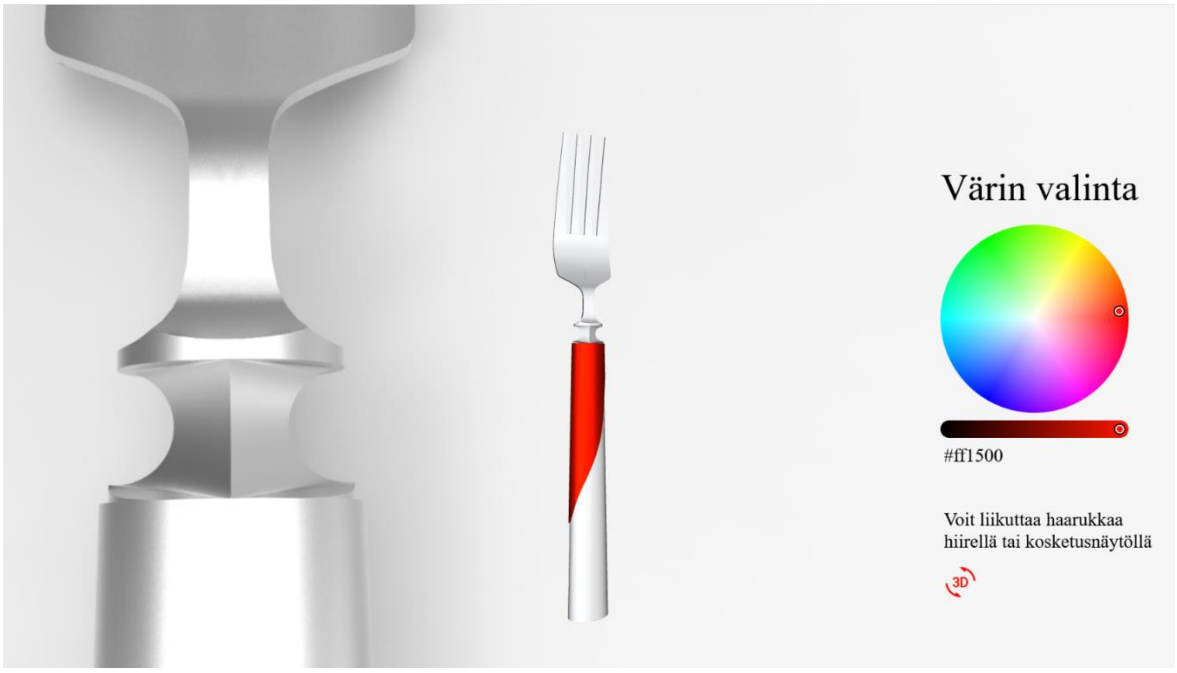
Sovelluksen aterin on Rhinoceros-ohjelmalla opintojen aikana mallinnettu haarukka. Alkuperäisen malliston ideana on, että aterimet ovat saatavilla useina eri väreinä (kuva 1). Rakennetussa sovelluksessa ideaa viedään eteenpäin siten, että asiakas voi itse valita kätevästi haarukalle minkä tahansa RGB-väriavaruuden värin. Lisäksi hän voi tarkastella mallia tarkemmin liikuttamalla sitä sovelluksessa.

KUVA 1. Haarukkamallisto. Alkuperäinen suunniteltu haarukkamallisto renderöitynä KeyShot-ohjelmalla (Pöllänen 2019)

Three.js on JavaScript 3D-kirjasto ja API (application programming interface), jolla voidaan luoda ja esittää 3D-tietokonegrafiikkaa selaimessa käyttäen WebGL:ää. Modernit selaimet tukevat WebGL:n käyttöä, joten käyttäjän ei tarvitse asentaa mitään ylimääräisiä selainlisäosia kirjaston käyttämiseksi. Three.js hyödyntää näytönohjainta, joten sillä on mahdollista visualisoida erityisen näyttävää ja sulavaa 3D-grafiikkaa selaimessa.

Tämä opinnäytetyö sisältää ohjelmointikoodia. Lukijalta oletetaan perusymmärrys webohjelmoinnista. Teksti sisältää ohjelmakoodin osia ja selityksiä näiden merkityksestä. Ohjelmakoodit on esitetty vaaleanharmaalla pohjalla `consolas`-fontilla. Opinnäytteen lopussa on liitteenä koko ohjelmakoodi, josta lukija voi tarkemmin tarkastella mihin kohtaan käsitellyt koodinosat kuuluvat lopullisessa sovelluksessa.

Opinnäytetyössä käytetyt ohjelmat (Three.js-kirjasto, Visual Studio Code ja Blender) ovat ilmaisia avoimen lähdekoodin, pois lukien Photoshop, jolla tehtiin pieniä muokkauksia sovelluksen taustakuvaan, sekä Rhinoceros, josta alkuperäinen haarukkamallin tuodaan ulos. Avoimen lähdekoodin ohjelmat ovat ladattavissa ilmaiseksi internetistä. Lopullinen ohjelma löytyy osoitteesta pikselipuuro.fi/haarukkademo (kuva 2).



KUVA 2. Kuvakaappaus tässä opinnäytetyössä tehdystä ohjelmasta (Pöllänen 2021 a)

Tämän työn tarkoitus on kerryttää kirjoittajan muotoiluosaamista ja tekniikoiden työkalupakkia Three.js-ohjelmoinnin ja 3D-mallin käsittelyn kautta. Lopullisen työn on tarkoitus tulla osaksi kirjoittajan portfoliota ja github-tiliä. Tämä raportti on suunnattu henkilöille, jotka ovat kiinnostuneita Three.js-ohjelmoinnista ja haluavat nähdä esimerkin Three.js:llä toteutetusta websovelluksesta. Three.js-ohjelmoinnista ei kirjoitushetkellä ole saatavilla suomenkielisiä tutoriaaleja tai ohjeita, joten raportti voi toimia myös suomenkielisenä opasteena Three.js:n perusteisiin.

2 MENETELMÄT JA OHJELMAT

2.1 Three.js ja WebGL

Three.js on JavaScript 3D-kirjasto ja API (application programming interface), jolla voidaan luoda ja esittää 3D-tietokonegrafiikkaa selaimessa käyttäen WebGL:ää. WebGL:n on kehittänyt Kronos Group (Khronos Group 2021). Kaikissa moderneissa selaimissa on tuki WebGL:lle, joten WebGL:n käyttämiseksi selaimessa ei tarvitse asentaa erillisiä selainlisäosia (toisin kuin käyttäessä mm. Flash:ia tai Javaa). WebGL suorittaa renderöinnin koneen näytönohjainta hyödyntäen, mikä mahdollistaa erittäin korkeatasoisen 3D-renderöinnin selaimessa - toki olettaen, että käyttäjällä on riittävän tehokas näytönohjain. WebGL-ohjelmointi suoraan JavaScript:illä on monimutkaista ja vaativaa. Three.js:n tarkoitus on tehdä WebGL:n käytöstä yksinkertaisempaa.

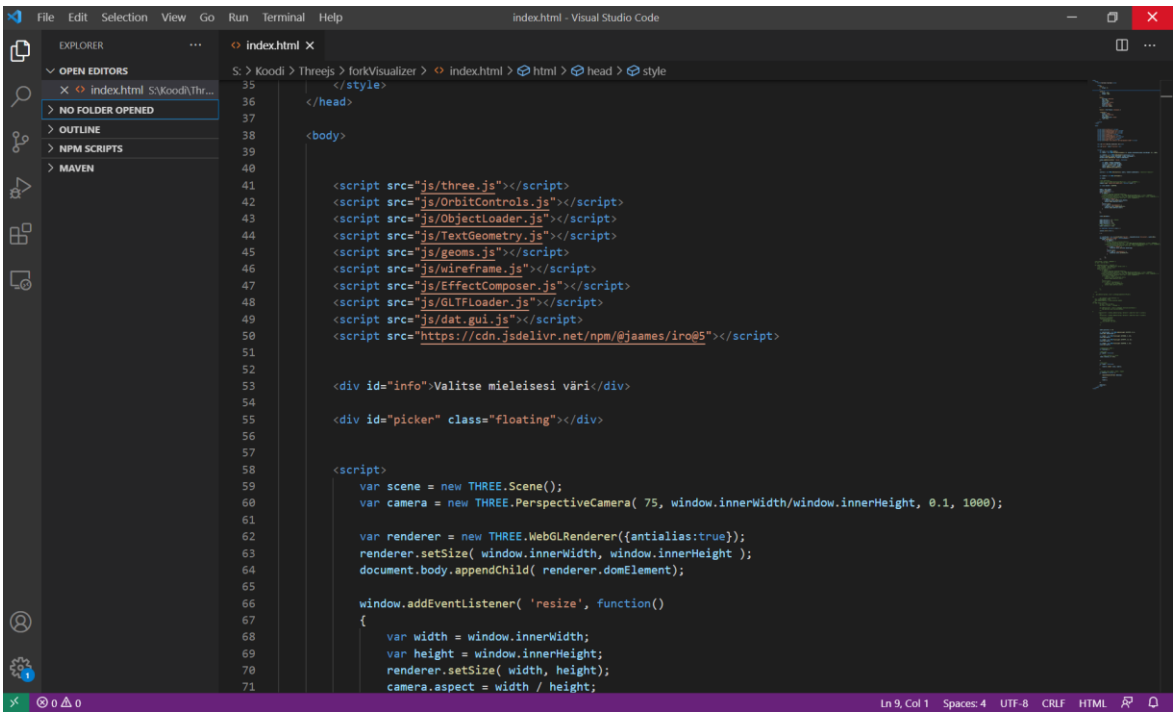
Ensimmäinen Three.js-versio julkaistiin 2010. Sen dokumentaatio löytyy Three.js:n virallisilta verkkosivuilta (Three.js julkaisuaika tuntematon a). Three.js:llä voi tehdä websovellusten lisäksi VR- (virtual reality) ja AR-ohjelmia (Augmented Reality).

Three.js:n käytöstä on lukuisia esimerkkejä. Sillä voidaan visualisoida reaaliaikaisesti lentoliikennettä (HERE Technologies 2019). Little Workshop on hyödyntänyt sitä grafiikkademossa (Little Workshop a julkaisuaika tuntematon) ja sisustussuunnittelussa huoneiston huonekalujen materiaalien valitsemisessa (Little Workshop b julkaisuaika tuntematon). Google puolestaan teki pelin, jolla pyritään opastamaan internetin käyttäjiä asialliseen ja muut huomioivaan internetkäyttäytymiseen (Google Inc. julkaisuaika tuntematon b). Blockbench on tehnyt selaimessa pyörivän 3D-mallieditorin (Blockbench julkaisuaika tuntematon).

Three.js:n käyttö edellyttää ymmärrystä JavaScript- ja HTML-ohjelmoinnista. Three.js:n opiskelun tueksi on julkaistu jonkin verran vieraskielistä kirjallisuutta, lähinnä englanniksi, kuten Learn Three.js (Dirksen, Jos 2018). Lisäksi Internetistä löytyy aiheesta useita tutoriaalivideoita kuten Garry Simonin (Simon, Garry 2019) ja Traversy Median tutoriaalit (Traversy Media 2019).

2.2 Visual Studio Code

Visual Studio Code (VS Code) on Microsoftin ilmainen koodieditori, joka soveltuu hyvin myös JavaScripti- ja Three.js-koodin käsittelyyn (KUVA 3). VS Code -editorissa on tuki virheenkorjaukselle, syntaksin korostukselle, automaattiselle koodin täydennykselle, refaktoroinille eli koodin järkeistämiselle ja Git-versionhallinnalle. Se toimii Windows, Linux ja MacOS-koneissa. VS Code on ladattavissa verkosta. (Microsoft 2021.)

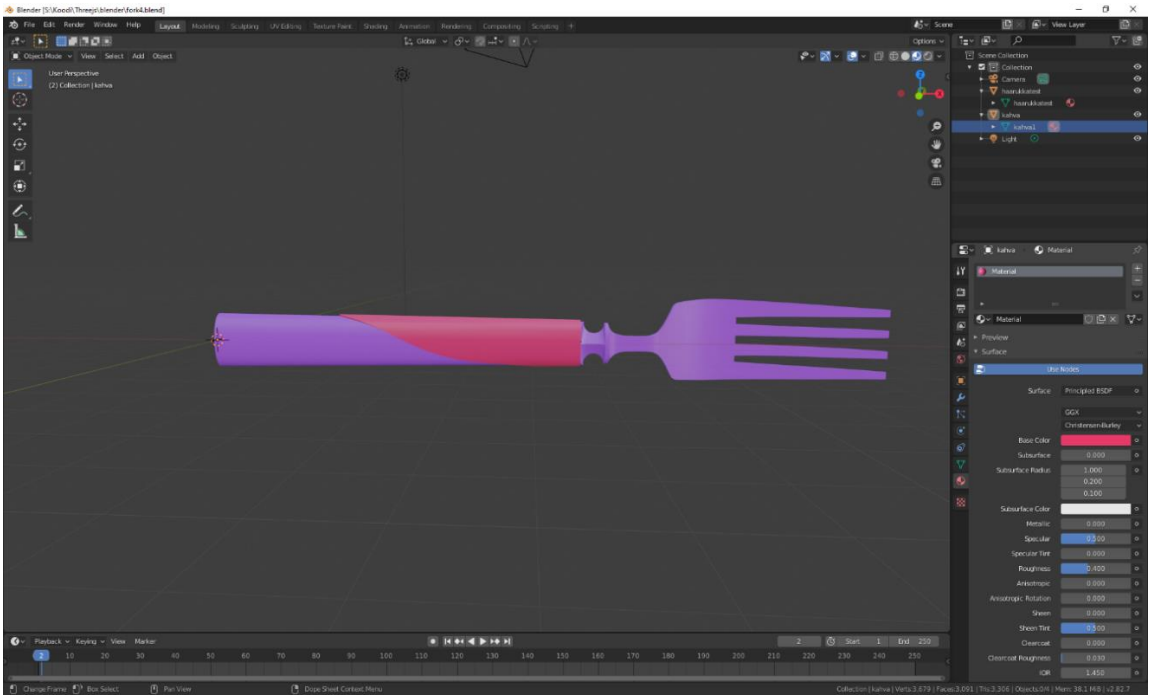


KUVA 3. Kuvakaappaus Visual Studio Codesta (Pöllänen 2021 b)

2.3 Blender

Blender on ilmainen avoimen lähdekoodin 3D-tietokonegrafiikan mallinnus- ja animointityökalu (KUVA 4). Sitä kehittää voittoa tavoittelematon Blender foundation. Blenderin ensimmäinen versio kehitettiin jo vuonna 1994. Blenderillä on jo suhteellisen pitkä kehityshistoria ja se on kehitetty hyvin monipuoliseksi. Blender on ladattavissa Blenderin virallisilta sivuilta. (Blender Foundation julkaisuaika tuntematon.) Blender Cloud:issa on myös paljon esimerkkejä ohjelman käytöstä (Blender Cloud julkaisuaika tuntematon).

Tässä työssä Blenderiä käytetään haarukkamallin muokkaamisessa sekä mallin tiedostoformaatin muuttamiseksi verkkosivukäyttöön suositeltavaan .glp-formaattiin. Glp-formaatti on glTF:n (Graphics Language Transmission Format) binäärimuoto, jossa tekstuurit ovat tiedostossa mukana.



KUVA 4. Kuvakaappaus Blender 3D-tietokonegrafiikan mallinnus- ja animointityökalusta (Pöllänen 2021 c)

3 TYÖN TOTEUTUS

3.1 Sovelluksen rakentaminen Three.js:llä

3.1.1 Alkuvalmistelut

Alussa asetettu tavoite oli omaksua Three.js:n perusteet ja luoda jokin lopullinen sovellus, joka visualisoisi dataa, jossa olisi jokin toiminto tai joka olisi visuaalisesti mielenkiintoinen. Työtä toteutettiin kokeilun ja yrittämisen kautta. Aluksi perehdyttiin kirjallisuuteen (Ghayour 2018, Dirksen 2018) ja käytiin läpi tutoriaaleja kuten Garry Simonin (Simon, Garry 2019) ja Traversy Median tutoriaalit (Traversy Media 2019).

Three.js on JavaScript 3D-kirjasto, jonka voi ladata käyttöön Three.js:n virallisilta sivuilta (Three.js julkaisuaika tuntematon a). Toinen vaihtoehto olisi viitata sovelluksen koodissa suoraan internetissä levitettävään versioon Three.js-kirjastosta. Tässä työssä päädyttiin lataamaan Three.js-kirjaston koodit omalle koneelle kehitystä varten, sillä kirjaston lataaminen verkosta sovellusta varten aiheuttaa lievän viiveen. Lopulta myös käytetyt kirjaston osat siirrettiin palvelimelle (pikselipuuro.fi) tuotetun ohjelmakoodin kanssa.

Sovelluksen kehitysvaiheessa käytettiin Node.js-pohjaista HTTP-palvelinta. Node.js on avoimen lähdekoodin alustariippumaton ajoympäristö JavaScript-applikaatioiden ajamiseksi palvelimella. Node.js ladattiin ajoympäristön websivuilta ja asennettiin Windows-koneelle. (OpenJS Foundation julkaisuaika tuntematon.) Node.js:n mukana tulee myös npm (node package manager), joka on JavaScript-paketinhallintamanageri. HTTP-palvelin asennettiin npm:n avulla projektin juurikansiossa seuraavalla käskyllä komentokehotteessa:

```
> npm install -g http-server
```

HTTP-palvelin luotiin projektin juurikansioon. Kaikki työssä käytetyt kuvat ja 3D-mallit tallennettiin juurikansion alikansioihin. Lokaali HTTP-palvelin voi lukea tiedostoja juurikansioista ja tämän alikansioista. Luotu HTTP-palvelin käynnistettiin juurikansiossa seuraavalla komennolla:

```
> http-server
```

Lokaali webpalvelin käynnistyi porttiin 8080. Käynnissä oleva ohjelma avattiin Firefox-selaimessa asettamalla osoitekenttään *localhost:8080*. Tämän työn kehitys ja testaus tehtiin pääasiassa Mozilla Firefox-selaimessa.

Kehitysvaiheessa on suositeltavaa tehdä lokaali HTTP-palvelin, joka mahdollistaa 3D-mallien lataamisen sovellukseen. JavaScript koodilla ei tietoturvasyistä voi ladata suoraan lokaalilla koneella olevia 3D-malleja selaimeen ilman palvelinta tai muokkaamatta selaimen asetuksia. Edellä mainituista vaihtoehdoista lokaalipalvelin on suositellumpi vaihtoehto, koska selaimen asetusten muokkaaminen 3D-mallien lataamiseksi suoraan selaimeen voi altistaa tietoturvauhille. Tiedostojen lukeminen suoraan selaimesta JavaScript-koodin kautta mahdollistaisi käyttäjän henkilökohtaisten tiedostojen lukemisen koodatun web-sivun kautta, ja siksi toiminnallisuus on oletusarvoisesti estetty kaikissa selaimissa.

3.1.2 HTML runko

Koska kyseessä on web-sovellus, aluksi ohjelmalle luodaan HTML-runko:

```
<html>
    <head>
        <title>Kustomoi haarukka</title>
        <style>
            body {
                margin: 0;
            }
            canvas {
                width: 100%;
                height: 100%;
            }
            #info {
                position: absolute;
                top: 10px;
                width: 100%;
                text-align: center;
                display:block;
                color:aliceblue;
                font-size: 100px;
            }
            #colors { font-family: monospace; }
            .floating {
                float: right;
                position: absolute;
                top: 500px;
                background-color: black;
                right:20px;
```

```
            }
        </style>
    </head>
    <body>
        <script src="js/three.js"></script>
        <script src="js/OrbitControls.js"></script>
        <script src="js/ObjectLoader.js"></script>
        <script src="js/TextGeometry.js"></script>
        <script src="js/geoms.js"></script>
        <script src="js/wireframe.js"></script>
        <script src="js/EffectComposer.js"></script>
        <script src="js/GLTFLoader.js"></script>
        <script src="js/dat.gui.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/@jaames/iro@5"></script>

        </script>
            //ohjelman toiminnallisuus tulee tähän
        </script>
    </body>
</html>
```

HTML-rungossa mm. määritellään tyylimuotoilut ja ladataan tarvittavat JavaScript-kirjastot. <head>lohkossa tehdään tyylimääritykset ja määritetään <body>lohkossa <script>elementeissä ladattavat Three.js-kirjaston JavaScript-tiedostot. Lisäksi rakennetaan ohjelman koko toiminnallisuus yhden <script>elementin sisään <body>lohkossa. Ohjelman koodi on tehty tässä opinnäytteessä yhteen tiedostoon, jotta kokonaisuus olisi lukijalle helpompi hahmottaa.

3.1.3 Scene, Camera, Renderer

Three.js-sovellus perustuu scene-objektiin, joka toimii säiliönä kaikille renderöitäville objekteille ja valoille. Scene-objectin luonti tapahtuu koodissa seuraavalla käskyllä:

```
var scene = new THREE.Scene();
```

Tämän jälkeen luodaan camera-objekti, joka määrittää mitä, näemme kun scene renderöidään. Tässä tapauksessa luodaan perspektiivikamera (PerspectiveCamera):

```
var camera = new THREE.PerspectiveCamera(75, window.innerWidth/window.in-
nerHeight, 0.1, 1000);
```

Tämän jälkeen luodaan renderer-objekti, joka laskee mitä näytöllä tulisi näkyä siinä suunnassa mihin camera-objekti on suunnattu, sekä määritetään ikkunan koko ja lisätään renderer-objekti dokumentin runkoon:

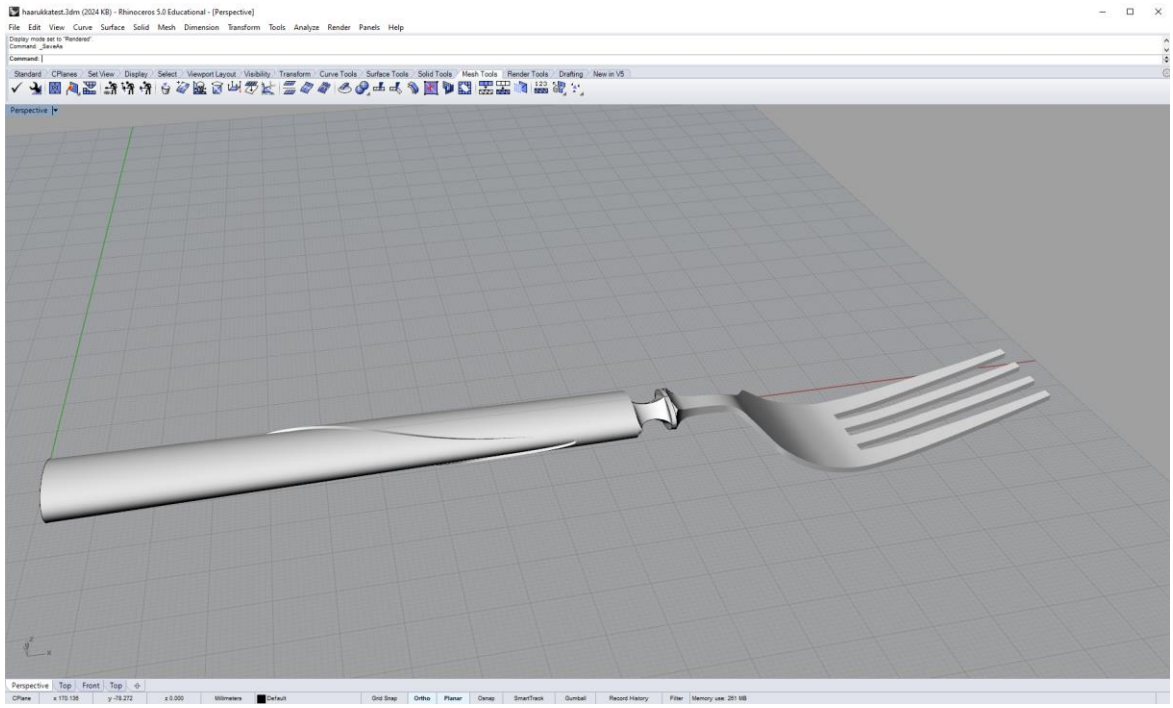
```
var renderer = new THREE.WebGLRenderer({antialias:true});
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement);
```

3.2 Haarukkamallin muokkaus

3.2.1 Mallin käsittely Blenderissä

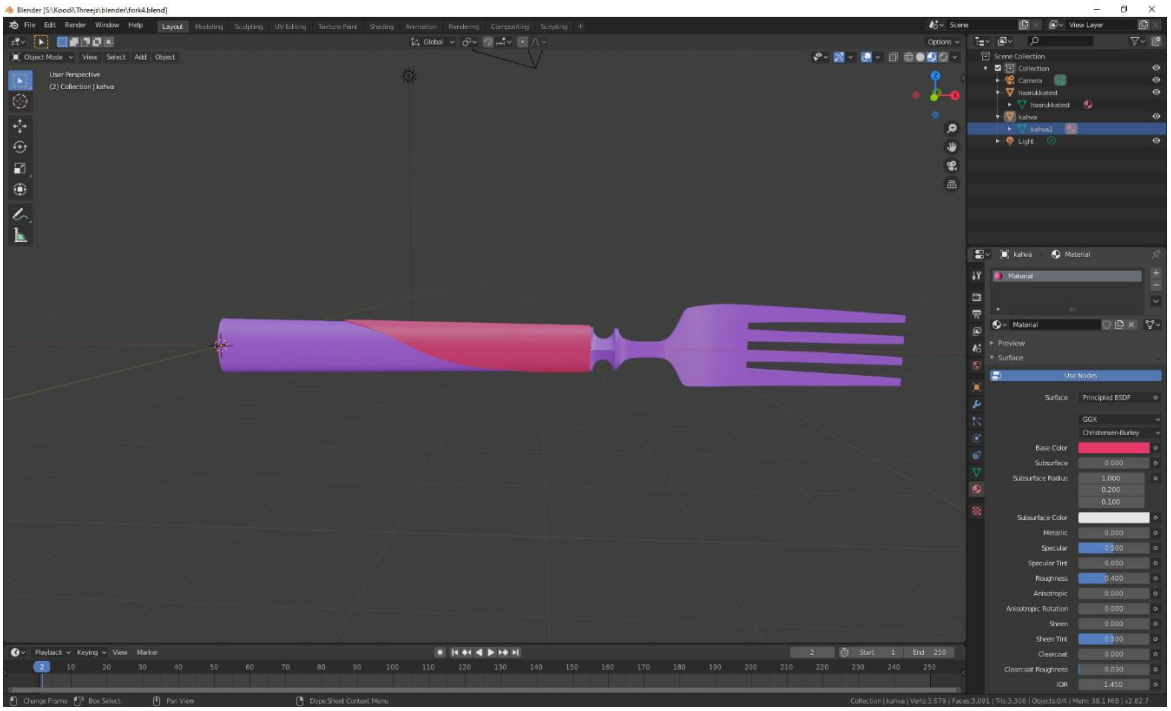
Ohjelman haarukkamalli on luotu Rhinoceros-ohjelmalla kurssityönä (KUVA 5). Malli on tallennettuna Rhinoceroksen omaan .3dm-formaattiin. Websovellusta varten malli on kuitenkin suositeltavaa muuttaa .glp-formaattiin. Glp-formaatti on glTF:n (Graphics Language Transmission Format) binäärimuoto, jossa tekstuurit ovat tiedostossa mukana. Glp:tä käyttäessä tekstuureja ei tarvitse referoida malliin mukaan erillisinä kuvina. (Library of Congress 2019)

Malli voidaan muuttaa .glp-formaattiin Blenderin avulla. Ensin Rhinoceroksessa tallennetaan malli .obj-formaatissa. Sitten malli viedään obj.-formaatissa Blenderiin, josta se viedään ulos .glp-formaatissa. Siirrettäessä mallia Rhinoceroksesta Blenderiin tulee mallin tarkkuuteen kiinnittää huomiota, jotta rakenne säilyy esteettisenä. Monimutkainen kappaleen rakenne voi kärsiä epäsopivasta tarkkuudesta.



KUVA 5. Kuvakaappaus haarukkamallista Rhinoceros-ohjelmassa (Pöllänen 2021 d)

Lisäksi alkuperäisestä mallista erotetaan kahvan väritettävä osa erilliseksi verkoksi (mesh), jotta se voidaan määrittää eri väriseksi kuin muu osa haarukkaa (KUVA 6). Tämän onnistuu käyttämällä Blenderin separate-toimintoa (Blender Foundation 2021).



KUVA 6. Haarukkamalli Blenderissä jaettuna kahteen erilliseen osaan. Eri osia havainnollistetaan kuvassa värein: runko on violetti ja kahvan koristeosa on vaaleanpunainen. (Pöllänen 2021 e)

3.2.2 Mallin vienti sovellukseen

Muokattu haarukkamalli viedään ohjelmaan .glp-formaatissa. Samalla mallille määritetään materiaalien ominaisuudet sopivilla parametreillä. Haarukan materiaaleista pyritään saamaan sopivan metallisen näköiset. Malli ladataan ohjelmaan ja materiaalit määritetään seuraavalla koodilla:

```
var loadery = new THREE.GLTFLoader();
var model;
loadery.load( 'models/fork_model.glb', function ( gltf ) {
model = gltf.scene;
console.log(model);
model.traverse((o) => {
if (o.isMesh) {
    if (o.name == "kahva") {
        o.material = new THREE.MeshPhysicalMaterial({
            metalness: 0.6,
            roughness: 0.5,
            reflectivity: 0.0,
            envMap: null
        });
        o.material.color.setHex(color_handle);
    }
    if ( o.name == "haarukkatest") {

        o.metalness = 1.0;
        o.roughness= 0.0;
        o.material.color.set(0xcaccce );
        o.material.reflectivity = 1.0;
        o.material.envMap = null;
```

```
                o.material.clearcoat = 0.0;

            }
        }
        });
        scene.add(model);
        model.position.z = 7; //radians
        model.position.x = 0;
        model.position.y = -8;
        model.rotation.z = 1.57;
        model.rotation.x = 6.25;

        }, undefined, function ( error ) {

        console.error( error );

        } );
```

Three.js sisältää useita erityyppisiä materiaaleja, jotka eroavat toisistaan ominaisuuksiltaan. Näitä on esimerkiksi MeshBasicMaterial, MeshPhongMaterial ja MeshPhysicalMaterial. Käytettäessä MeshBasicMaterial:ia tämä ei ota huomioon scene-objektin valoja eikä mallin pintaan muodostu heijastuksia. MeshPhongMaterial saa puolestaan mallin näyttämään muoviselta. MeshPhysicalMaterial heijastaa valoja ja sisältää metalness- ja roughness-ominaisuuset, joita muokkaamalla voidaan kappaleen pintaan saada metallin näköisyyttä. Tässä työssä haarukan materiaali määritetään MeshPhysicalMaterial:illa. (Three.js julkaisuaika tuntematon b.)

Miellyttävän lopputuloksen saavuttamiseksi voi joutua tekemään lukuisia materiaali- ja parametrikokeiluja. Lisäksi materiaalin käyttäytymiseen vaikuttaa myös valaistus, joten materiaaleja säätäessä tulee myös säätää valaistusta.

3.2.3 Valaistus

Valaistus on oleellinen osa Three.js-sovellusta, etenkin käyttäessä materiaaleja, jotka heijastavat valoja. Haarukkamallissa oli metallinen materiaali. Sovellukseen lisätään useita pistevaloja (PointLight) mallin ympärille, jotta mallin metallisuus korostuisi heijastusten kautta. Ohjelmaan lisätään neljä PointLight:ia seuraavalla koodilla:

```
        const frontLightLeft = new THREE.PointLight( 0xffffff, 0.5, 0,2 );
        frontLightLeft.position.set( 45, 45, 180 );
        scene.add( frontLightLeft );

        const frontRightLight = new THREE.PointLight( 0xffffff, 0.5, 0,2 );
        frontRightLight.position.set( -45, -45, 180 );
        scene.add( frontRightLight );

        const frontCenterLight = new THREE.PointLight(0xffffff, 0.25, 0,2 );
        frontCenterLight.position.set( -10, -0, 400 );
        scene.add( frontCenterLight );
```

```
const backLight = new THREE.PointLight( 0xffffff, 0.7, 100,2 );
backLight.position.set( -5, 5, -18 );
scene.add( backLight );
```

Three.js sisältää useita erilaisia mahdollisia valonlähteitä, jotka eroavat toisistaan ominaisuuksiltaan. Näitä on esimerkiksi AmbientLight, DirectionalLight ja PointLight. Käytettäessä AmbientLight:ia tämän väri lisätään kaikkialle. Sillä ei ole tiettyä suuntaa ja se ei muodosta varjoja. SpotLight on puolestaan kaukainen valonlähde, joka valaisee koko alueen samalla valonmäärällä. PointLight on valonlähde, joka himmenee etääntyessä kohteestaan, toisinkuin DirectionalLight, missä valonmäärä on vakio. Useampi PointLight aseteltuna kappaleen ympärille, luo kappaleeseen useita heijastumia, mikä korostaa esimerkiksi metallin luonnetta.

3.2.4 Kahvan värin muuttaminen käyttäjän toimesta

Ohjelmassa hyödynnetään James Danielin tekemää väripoiminta-widgettiä (kuva 7). Tämä widgetti on JavaScript-pohjainen, eikä se sisälly Three.js-kirjastoon. Widgetin käytön ohjeet löytyvät James Danielin ylläpitämiltä verkkosivuilta. Widgetin käyttö on lisensoitu MPL 2.0:lla (Mozilla Public License) ja se on täysin ilmainen sekä omaan, että kaupalliseen käyttöön. (Daniel, James julkaisuaika tuntematon a.)

KUVA 7. Värinpoiminta-widgetti (Daniel julkaisuaika tuntematon b)

Seuraavalla koodilla määritetään widgetti ja toteutetaan toiminnallisuus, missä haarukan väri muuttuu, kun käyttäjä siirtää kursoria väripoiminta-widgetin päällä:

```
var color_handle = 0xff1500;
var colorPicker = new iro.ColorPicker('#picker',
{layoutDirection:"vertical", width:300,color: "#ff1500"});
colorPicker.on('color:change', function(color) {
    model.traverse((o) => {
        if (o.isMesh) {
            if (o.name == "kahva") {
                o.material.color.set(color.hexString);
                console.log(color.hexString)
```

```
                        }
                        if ( o.name == "haarukkatest") {
                            o.material.color.set(0xcaccce );
                        }
                    }
                    });
                document.getElementById("hex-text").innerHTML = color.hexString;
            });
```

Widgetin taustaväri tulee määrittää taustan väriseksi, kun widgetin halutaan sulautuvan mahdollisimman eleettömästi taustaan. Taustakuvan värin hex-arvo asetetaan widgetin väriksi. Oletusarvoisesti widgetin ympärillä on musta laatikko.

3.3 Muita ohjelman osia

3.3.1 Taustakuva

Three.js:n tausta on oletusarvoisesti musta, joten taustalle tulee lisätä kuva tai jokin toinen väri, jos sovellukseen ei haluta mustaa taustaa. Ohjelma taustakuvaksi lisätään vaalean harmaa muokattu renderöintikuva haarukasta, jossa korostuu mallin varren yksityiskohta (kuva 8, Pöllänen 2021). Alkuperänen kuva on renderöity KeyShot-ohjelmalla Mallinsuunnittelu-kurssilla. Alkuperäinen kuva ei kuitenkaan ollut tarpeeksi leveä, joten kuvaa tuli kasvattaa. Photoshop-ohjelmalla valitaan eyedropper tool:illa taustan väri ja levitetään tämän väristä aluetta oikealle puolelle lisää. Lopullinen kuva lisätään ohjelmaan seuraavalla koodilla:

```
const loader = new THREE.TextureLoader();
const bgTexture = loader.load('kuvat/tausta.png');
scene.background = bgTexture;
```

KUVA 8. Ohjelman taustakuva (Pöllänen 2021 f)

3.3.2 Kappaleen pyöriminen ja liike

Three.js:ssä voidaan liikuttaa kappaleita. Automaattinen liike voidaan määrittää update-funktion sisällä. Haarukka asetetaan pyörimään sovelluksessa seuraavalla koodilla:

```
var update = function()
{
    if (rotate) {
        model.rotation.y += 0.01;
    }
};
```

Pyörimisen hallitsemiseksi koodiin lisätään nappula, jolla pyörimisen voi pysäyttää ja aloittaa uudelleen. Nappulaan lisätään toimintaa kuvaava ikoni. Käyttöliittymissä käytetään nykyisin tyypillisesti ikoneita tekstin sijaan minimalistisessa tyylissä. Tällöin sovellus ei rajoitu tiettyyn kieleen, eikä siitä tarvitse tehdä eri versioita eri kielillä. Tässä sovelluksessa käytetään seuraavaa ikonia napissa, millä pyörimisen voi lopettaa: 

(Google Inc. julkaisuaika tuntematon a).

Ikoniin lisätään vielä värin muutos, kun nappia painetaan. Ikoni on punainen kun haarukka pyörii, ja vihreä kun haarukka on paikallaan. Nappitoiminto ja värin vaihtuminen toteutetaan seuraavalla koodilla:

```
document.getElementById("toggleRotateButton").addEventListener(   "click",
             function(event) {
                 console.log(event.target.innerHTML)

             if (rotate) {
```

```
                document.getElementById("rotateImg").classList.remove('red');
                document.getElementById("rotateImg").classList.add('green');
            }
            else {
                document.getElementById("rotateImg").classList.remove('green');
                document.getElementById("rotateImg").classList.add('red');
                 }
                   rotate = !rotate;
               },
               false
           );
           var rotate = true;
```

Lisäksi ohjelmassa voidaan määrittää OrbitControls-kontrollit, joilla käyttäjä voi liikuttaa esimerkiksi scene:n kameraa:

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

Nyt käyttäjä voi tarkastella haarukkaa eri kulmista sekä läheltä ja kaukaa hiiren tai kosketusnäytön avulla. Toiminnallisuus lisää ohjelman dynaamisuutta.

3.3.3 Ohjelman käynnistys ja visualisointisilmukka

Ohjelman varsinainen käynnistys ja visualisointisilmukka tapahtuu seuraavalla koodilla:

```
           var render = function()
           {

               const canvas = renderer.domElement;
               const canvasAspect = canvas.clientWidth / canvas.clientHeight;
               const imageAspect = bgTexture.image ? bgTexture.im-
age.width / bgTexture.image.height : 1;
               const aspect = imageAspect / canvasAspect;

               bgTexture.offset.x = aspect > 1 ? (1 - 1 / aspect) / 2 : 0;
               bgTexture.repeat.x = aspect > 1 ? 1 / aspect : 1;

               bgTexture.offset.y = aspect > 1 ? 0 : (1 - aspect) / 2;
               bgTexture.repeat.y = aspect > 1 ? 1 : aspect;

               renderer.render( scene, camera);

           }

           var VisualizationLoop = function ()
           {
               requestAnimationFrame( VisualizationLoop);
               update();
               render();
           };
```

```
            VisualizationLoop();
```

Ohjelma kutsuu toistuvasti funktiota VisualizationLoop, jossa scene päivitetään kutsumalla update-funktiota update ja renderöidään kutsumalla funktiota render. Funktiossa render myös säädetään taustakuvan mittasuhteet vastaamaan selainikkunan kokoa. Näin taustakuva pysyy mielekkään muotoisena myös selainikkunaa pienennettäessä tai suurennettaessa.

4 POHDINTA

Tässä opinnäytetyössä tein aterimen värin valintasovelluksen Three.js:llä. Three.js on JavaScript 3d-kirjasto ja API (application programming interface), jolla voidaan luoda ja esittää 3D- tietokonegrafiikkaa selaimessa käyttäen WebGL:ää. Hyödynsin työssäni aiemmilla kursseilla Rhinoceroksella mallintamani haarukkamalli ja ideaa siitä, että suunnittelemassani mallistossa sitä olisi tarjolla useissa eri väreissä. Nyt vein ajatusta askeleen pidemmälle, siten että asiakas voi valita tuotteelle minkä tahansa RGB-väriavaruuden värin websovelluksessa. Muokkasin mallin soveltuvaksi websovellukseen Blenderillä ja loin mallin ympärille interaktiivisen websovelluksen. Tässä opinnäytetyössä tekemäni ohjelma löytyy osoitteesta pikselipuuro.fi/haarukkademo (KUVA 2).

Motivaatio tehdylle ohjelmalle oli oma kiinnostukseni Three.js ohjelmointiin ja 3D-mallien käsittelyyn. Lopullisen työn on tarkoitus tulla osaksi omaa portfoliotani, mutta voisin myös tarjota ohjelmaa Three.js:n ylläpitämään sivustoon erilaisista projekteista (Three.js julkaisuaika tuntematon a).

Aihe oli ajoittain melko haastava. Alussa lähdin yrittämään webgrafiikan tekoa suoraan WebGL:llä, ennen kuin löysin Three.js:n – kirjaston, joka tekee WebGL-sovellusten toteuttamisesta suoraviivaisempaa ja etenkin aloittelijoille helpompaa. Minulla ei ollut aiempaa kokemusta WebGL tai Three.js-ohjelmoinnista – lähinnä intoa kokeilla. Oma aiempi ohjelmointikokemukseni töiden puolesta perustuu lähinnä tietokantaohjelmointiin ja data-analytiikkaan. Siten projektin alussa aikaa kului paljon 3D-ohjelmoinnin perusteiden ja Three.js:n syntaksin omaksumisessa.

Aluksi tein paljon Three.js tutoriaaleja, joita löysin mm. youtubesta ja erilaisilta verkkosivuilta. Lisäksi hankin kaksi kirjaa työni tueksi: Real-Time 3D Graphics with WebGL 2 (Chayour, Farhad 2018) ja Learn Three.js (Dirksen, Jos 2018). Ennen lopullista haarukan värivalintasovellusta tein harjoitelman, jossa harjoittelin 3D-objektien muodostusta ja liikuttamista Three.js-ohjelmoinnilla. Tämä harjoitelma löytyy osoitteesta pikselipuuro.fi/bittiboksi.

Minua on aina kiinnostanut 3D-ohjelmointi, jossa muotoja ja liikkeitä hallitaan tilassa puhtaasti ohjelmakoodilla. Teollisen muotoilun opinnot ovat opettaneet minulle paljon 3D-objekteista, tekstuureista ja valaistuksesta. Three.js mahdollistaa suunnittelemieni mallien viemisen interaktiiviseen websovellukseen ja esimerkiksi näiden mallien muokkaamisen sovelluksen käyttäjän toimesta.

Yksi suurimmista haasteistani oli saada 3D-malli vietyä Three.js ohjelmaan eheänä. Aluksi en saanut edes mallia näkymään. Jonkin aikaa verkkohakuja tehtyäni minulle selvisi, että jos Three.js-sovellukseen vie 3D-mallin, tulee tämä tehdä palvelimen välityksellä. Lopulta sain lokaalin palvelimen toimimaan ja sain mallin vietyä .obj-muodossa, mutta se näyttäytyi täysin mustana, enkä saanut siihen liitettyä mitään materiaaleja. Sitten taas tovin verkosta tietoa etsittyäni minulle selvisi että malli kannattaa viedä .glp-formaatissa, ja että tämän muutoksen voisi tehdä Blenderissä. Rhinoceros ei kuitenkaan tue .glp-formaattia, joten sen kautta formaatin muutos ei onnistunut. Asensin Blenderin jotta voisin tehdä sillä tarvittavat muutokset. Minulla ei ole paljoa aiempaa kokemusta Blenderin käytöstä, joten aluksi minun tuli tutustua tarkemmin sen toiminnallisuuksiin. Lisäksi minulla kului jonkin aikaa löytää ne toiminnallisuudet, joilla sain jaettua mallin runkoon ja kahvaan. Tämä jako tuli tehdä, jotta oli mahdollista vaihtaa vain kahvan väriä Three.js-ohjelmassa. Tässä kohtaa opin paljon

3D-mallin tiedoston (puu)rakenteesta ja kuinka pystyn tarkastemaan mallin rakennetta ja ominaisuuksia Firefox-selaimen Developer Tools:in kautta.

Toinen haastava vaihe oli saada Three.js:ssä haarukka metallin väriseksi ja valaistus visioni mukaiseksi. Alun kokeiluissa haarukka näytti mattapintaiselta muovilta. Säätämällä materiaalin metallisuutta ja karkeutta onnistuin useiden yritysten kautta löytämään yhdistelmän, joka oli mahdollisimman realistinen. Tätä fotorealistisempi metallisuus vaatisi monimutkaisempaa mallin käsittelyä Blenderissä. Metallisuus ilmenee katsojalle heijastuksina, jotka edellyttävät riittävän määrän sopivasti sijoiteltuja valoja. Päädyin sijoittamaan haarukan ympärille neljä pistemäistä valonlähdettä, kaksi eteen ja kaksi taakse. Valojen sijoittaminen paikalleen vaati 3D-avaruuden hahmottamista, sillä ne tuli asetella parametrein avaruuden pisteisiin. Säätämällä valojen voiman riittävän suureksi muttei liian häikäiseväksi, pystyin luomaan visiotani vastaavan asetelman.

Sopivan värivalintawidgetin löytyminen, ohjelmaan yhdistäminen ja asettelu vei oman aikansa. Päädyin Daniel Jamesin tekemään widgettiin (Daniel, James julkaisuaika tuntematon a), koska se oli yksinkertaisen oloinen ja sain sen toimimaan koodissa haluamallani tavalla. Lisäsin käyttöliittymään widgetin alle myös näkyviin valitun RGB-avaruuden hexvärikoodin.

Halusin tehdä sovelluksesta visuaalisesti mahdollisimman yksinkertaisin ja valoisan. Three.js sovelluksen tausta on oletusarvoisesti musta. Päädyin laittamaan taustaksi taustakuvan, joka on vaalean harmaa ja sisältää vasemmassa laidassa lähikuvan haarukan kaulasta – mallin koristeellisimmasta osasta. Oikeaan reunaan asetin värinvalintawidgetin sekä napin, jota painamalla käyttäjä pystyy säätelemään haarukan pyörimistä: päälle ja pois. Napissa ei ole mitään tekstiä. Se on toteutettu toimintaa kuvaavalla ikonilla, jotta käyttöliittymä pysyisi mahdollisimman yksinkertaisena.

Sovelluksia rakennetaan nykyisin selaimeen JavaScriptillä todella paljon. Three.js mahdollistaa grafiikan liittämisen noihin sovelluksiin. Three.js:llä voi luoda visuaalisesti mielenkiintoista interaktiivista animaatiota tai webtaidetta. Kiinnostus digitaaliseen taiteeseen on maailmalla kasvussa ja etenkin kryptotaide on pinnalla (Thompson, Clive 2021). Tehdessäni tätä opinnäytetyötä olen seurannut kuinka suuria summia sijoittajat ovat laittaneet lohkoketjupohjaiseen NTF (non-fungible token) digitaiteeseen. Digitaalinen taide siihen linkitetyllä lohkoketjupohjaisella omistuksella on mielestäni mielenkiintoinen konsepti. Three.js:llä on mahdollista muun muassa luoda interaktiivista digitaidetta kosketusnäytöllisiin näyttöihin.

Käytin tässä työssä ilmaisia ohjelmia: Three.js:ää, Visual Studio Code:a ja Blenderiä. Etenkin kaupalliset animaatio-ohjelmat ovat tyypillisesti hyvin kalliita, ja aloittelevalla muotoilijalla harvemmin on mahdollista ostaa esimerkiksi Autodesk 3D MAX 3D-mallinnus- ja renderointiohjelmistoa (Autodesk, Inc. 2021). Blender Foundationin missiona on tarjota mahdollisuus 3D-mallien käsittelyn ja muokkauksen ilmaiseksi avoimen lähdekoodin versiolla. Avoimen lähdekoodin periaatteiden mukaisesti lähdekoodiin on mahdollista tutustua ja muokata lähdekoodia sekä käyttää ohjelmaa mihin tahansa tarkoitukseen. Avoimen lähdekoodin kannattajat kokevat, että kuten tieteessä, tieto kuuluu kaikille. Lisäksi koetaan, että ohjelman bugit löytyvät helpommin, kun lähdekoodi on vapaasti luettavissa.

Valitsin itselleni tämän aiheen, jossa koin pystyväni yhdistämään sitä mitä opin teollisen muotoilun opinnoissa aiempaan ohjelmointitaustaani. Luulen että myös jatkossa tulen hyödyntämään muotoilun opintojani poikkitieteellisesti työelämässä ja siksi tällainen soveltava aihe tuntui luontevalta valinnalta. Olen opiskellut aiemmin tietojenkäsittelytiedettä ja ohjelmoinut myös työssäni. Opiskelen Savonialla teollista muotoilua monimuotovuosikurssilla, missä monilla vuosikurssilaisillani oli jo muidenkin alojen opintoja ja työelämän kokomusta takana - kuten itsellänikin. Moni vuosikurssilainen ammensi näkemystä omaan opinnäytteeseen myös aiemmasta taustastaan, ja itsekin nyt valitsin saman polun. Näin opintojeni viimeisessä työssä halusin yhdistää ohjelmoinnin ja muotoilun.

Muotoilijan työnkuva voi olla erittäin moninainen. Muotoilija voi taidoillaan luoda esim. uusia tuotteita, palveluja tai elämyksiä. Mielestäni nykypäivän muotoilijan on tärkeää uskaltaa hyödyntää pelottomasti erilaisia digitaalisia työkaluja, kuten mallinnus- ja kuvankäsittelyohjelmia. Jatkuva uuden oppiminen myös itsenäisesti on mielestäni olennainen osa nykypäivän työelämää ja tämä pätee myös muotoilijan uralla. Tässä työssä osoitin kykyä omaksua ja hyödyntää uusia työkaluja itsenäisesti.

Opin paljon tätä työtä tehdessä. Opin Three.js:n perusteet, lisää webohjelmoinnista sekä 3D-mallin käsittelystä. Opin erityisesti mitä tulee ottaa huomioon, kun 3D-malli viedään websovellukseen, ja miten sen osia voi käsitellä koodissa, kuten tässä kahvan värin muuttaminen käyttäjän valinnan mukaan.

Työtä voisi vielä kehittää eteenpäin. Haarukan liikuttaminen sovelluksessa on vielä aika villiä. Liikerataa voisi vielä rajoittaa, jotta käyttäjä ei esim. heitä haarukkaa pois koko näkymästä. Lisäksi olen testannut sovellusta lähinnä oman työasemani näytöltä Firefox-selaimella. Katsoin myös, miten sovellus käyttäytyy puhelimen näytöllä, mutta siinä sovellus ei asettunut aivan yhtä hyvin. Prototyyppiä eteenpäin kehitettäessä pitäisi websovellusta testata ennen julkaisua useilla eri näytöillä sekä eri selaimilla.

Jos sovellusta jatkokehitettäisiin tavoitteena luoda sovellus, jonka kautta käyttäjä voisi tilata kyseisen valitsemansa värisen haarukan, niin ohjelmaan tulisi lisätä tilauslomake ja tietokanta. Tilauslomakkeeseen käyttäjä voisi syöttää yhteystietonsa ja haluamansa määrän aterimia. Lomakkeen tiedot tallentuisivat tämän jälkeen tietokantaan, josta aterimien valmistaja voisi hakea tilaustiedot. Nyt värivalinta pysyy vain ohjelman käyttäjän tietokoneen välimuistissa ja katoaa kun selain suljetaan.

Värivalintasovellus voisi toimia yhtenä toiminnallisena osana nettikauppaa. Ohjelmaa voisi myös muokata ja laittaa siihen jonkin muun tuotteen, jonka väriä voisi muuttaa. Lopputulos on kuitenkin itsenäisesti toimiva prototyyppi, jonka pohjalta konseptia voi jo demonstroida ja kysyä mielipiteitä, joiden pohjalta sovellusta voisi kehittää eteenpäin.

LÄHTEET

Autodesk, Inc. 2021. 3ds Max -ohjelmisto | Katso hinnat ja osta virallinen 3ds Max 2022. Verkkojulkaisu. https://www.autodesk.fi/products/3ds-max/overview?term=1-YEAR. Viitattu 16.5.2021.

Blender Cloud julkaisuaika tuntematon. Blender Cloud. Verkkojulkaisu. https://cloud.blender.org/welcome/. Viitattu 17.5.2021.

Blender Foundation julkaisuaika tuntematon. Blender.org – Home of the Blender project – Free and Open 3D Creation Software. Verkkojulkaisu. https://www.blender.org/. Viitattu 20.4.2021.

Blender Foundation 2021. Separate – Blender Manual. Verkkojulkaisu. https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/separate.html. Viitattu 20.4.2021.

Blockbench julkaisuaika tuntematon. Blockbench a boxy 3D model editor. Verkkojulkaisu. https://blockbench.net/. Viitattu 9.5.2021.

Daniel, James julkaisuaika tuntematon a. Iro.js. Verkkojulkaisu. Iro.js. https://iro.js.org/. Viitattu 12.5.2021.

Daniel, James julkaisuaika tuntematon b. Värinpoiminta-widgetti. Kuvakaappaus osasta Iro.js verkkosivua. Verkkosivut omistaa Daniel James.

Dirksen, Jos 2018. Learn Three.js, Third Edition. Birmingham: Packt Publishing.

Ghayour, Farhad 2018. Real-Time 3D Graphics with WebGL 2. Birmingham: Packt Publishing.

Google Inc. julkaisuaika tuntematon a. Icons – Google Fonts. Verkkojulkaisu. https://fonts.google.com/icons. Viitattu 15.5.2021.

Google Inc. julkaisuaika tuntematon b. Interland: Be Internet Awesome. Verkkojulkaisu. https://beinternetawesome.withgoogle.com/en_us/interland. Viitattu 10.5.2021.

HERE Technologies 2019. Mapping Live Flights: Working with HERE Studio and Three js. Youtube-video. https://www.youtube.com/watch?v=E78Pw2d-kpM. Viitattu 16.5.2021.

Khronos Group 2021. WebGL Overview – The Khronos Group Inc. Verkkojulkaisu. https://www.khronos.org/webgl/. Viitattu 15.7.2021.

Library of Congress 2019. glTF (GL Transmission Format) home. Verkkojulkaisu. Sustainability of Digital Formats: Planning for Library of Congress Collections. https://www.loc.gov/preservation/digital/formats/fdd/fdd000498.shtml. Viitattu 17.5.2021.

Little Workshop julkaisuaika tuntematon a. Virtual Showroom. Verkkojulkaisu. https://showroom.littleworkshop.fr/. Viitattu 10.5.2021.

Little Workshop julkaisuaika tuntematon b. Track. Verkkojulkaisu. Demo. https://demos.littleworkshop.fr/track. Viitattu 10.5.2021.

Microsoft 2021. Visual Studio Code – Code Editing. Refined. Verkkojulkaisu. Visual Studio Code. Viitattu 1.4.2021.

OpenJS Foundation julkaisuaika tuntematon. Node.js. Verkkosivu. Node.js:n lataussivu. https://nodejs.org/en/. Viitattu 23.5.2021.

Pöllänen, Irene 2019. Haarukkamallisto. Renderöity KeyShot-ohjelmalla. Kuopio: Irene Pölläsen kokoelmat.

Pöllänen, Irene 2021 a. Kuvakaappaus tässä opinnäytetyössä tehdystä ohjelmasta. Kuvakaappaus. Kuopio: Irene Pölläsen kokoelmat.

Pöllänen, Irene 2021 b. Kuvakaappaus Visual Studio Codesta. Kuvakaappaus. Visual Studio Code on Microsoftin tuote.

Pöllänen, Irene 2021 c. Kuvakaappaus Blender 3D tietokonegrafiikan mallinnus- ja animointityökalusta. Kuvakaappaus. Blender:iä kehittää Blender Foundation.

Pöllänen, Irene 2021 d. Kuvakaappaus haarukkamallista Rhinoceros-ohjelmassa. Kuvakaappaus. Rhinoceros-ohjelmiston omistaa Robert McNeel & Associates.

Pöllänen, Irene 2021 e. Haarukkamalli Blenderissä jaettuna kahteen erilliseen osaan. Kuvakaappaus. Blender:iä kehittää Blender Foundation.

Pöllänen, Irene 2021 f. Ohjelman taustakuva. Kuva. Kuopio: Irene Pölläsen kokoelmat.

Simon, Garry 2019. Three.js Crash Course for Absolute Beginners - 3D in the Browser. Youtube-video. https://www.youtube.com/watch?v=6oFvqLfRnsU. Viitattu 20.4.2021.

Thompson, Clive 2021. The NTF Gold Rush: How Cryptoartists Kick-Started a Boom. Verkkojulkaisu. New Your Times. https://www.nytimes.com/2021/05/12/magazine/nft-art-crypto.html. Viitattu 17.5.2021.

Three.js julkaisuaika tuntematon a. Dokumentaatio ja verkkosivut. Verkkojulkaisu. https://threejs.org/. Viitattu 12.3.2021.

Three.js julkaisuaika tuntematon b. Three.js Material. Verkkojulkaisu. https://threejs.org/docs/#api/en/materials/Material. Viitattu 16.5.2021.

Traversy Media 2019. Getting Started With Three.js. Youtube-video. https://www.youtube.com/watch?v=8jP4xpga6yY. Viitattu 13.4.2021.

LIITE 1: OHJELMAKOODI

```
<!doctype html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
        <title>Haarukkademo</title>

        <style>
            body {
                margin: 0;
            }

            canvas {
                width: 100%;
                height: 100%;
            }

            #info-header {
                width: 100%;
                text-align: bottom;
                color:black;
                font-size: 60px;
                bottom: 0;
                word-wrap: break-word;
                position: absolute;
            }

            .info-text {
                color:black;
                font-size: 30px;
                top: 0;
                margin: 5px;
                padding-right: 10px;
                margin-right: 10px;
            }

            #hex-text {
                color:black;
                font-size: 30px;
                top: 0;
                position: absolute;
                margin: 5px;
            }

            #colors { font-family: monospace; }

            .sidenav {
                height: 100%;
                position: fixed;
                z-index: 1;
                top: 0;
```

```
        background-color: transparent;
        overflow-x: hidden;
        padding-top: 20px;
        width:20%;
        right:0
    }

    .floating {
        float: right;
        position: relative;
        background-color: transparent;
    }

    .content {
        height: 100%;
        display: flex;
        flex-direction: column;
        -webkit-box-align: center;
        -moz-box-align: center;
        -ms-flex-align: center;
        -webkit-align-items: center;
        align-items: center;
        justify-content: center;
    }

    .box {
        width: 100%;
        height: 100px;
        margin: 5px;
        padding: 15px;
        position:relative;
    }
    .spacer {
        width: 100%;
        height: 2%;
        position:relative;
    }
    .colorpicker-box {
        width: 100%;
        height: 320px;
        margin: 5px;
        padding: 15px;
        position:relative;
    }

    .hex-box {
        width: 100%;
        height: 25px;
        margin: 5px;
        padding: 15px;
        position:relative;
    }
```

```
            .button {
                background-color: transparent;
                border: none;
                color: white;
                text-align: center;
                text-decoration: none;
                display: inline-block;
                font-size: 18px;
                width: 110px;
                text-align: left;
            }
            .red {
                filter: invert(20%) sepia(97%) saturate(4013%) hue-rotate(353deg) brightness(93%) con-
trast(127%);
            }
            .green {
                filter: invert(20%) sepia(97%) saturate(4013%) hue-rotate(100deg) brightness(93%) con-
trast(127%);
            }
        </style>
        <link href="https://fonts.googleapis.com/icon?family=Material+Icons"rel="stylesheet">

    </head>

    <body>


        <script src="js/three.js"></script>
        <script src="js/OrbitControls.js"></script>
        <script src="js/ObjectLoader.js"></script>
        <script src="js/TextGeometry.js"></script>
        <script src="js/geoms.js"></script>
        <script src="js/wireframe.js"></script>
        <script src="js/EffectComposer.js"></script>
        <script src="js/GLTFLoader.js"></script>
        <script src="js/dat.gui.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/@jaames/iro@5"></script>

        <div id="sidenav" class="sidenav">
            <div id="container" class="content">
                <div class="box"><div id="info-header">Värin valinta</div></div>
                <div id="picker" class="colorpicker-box"></div>
                <div class="hex-box"><div id="hex-text">#ff1500</div></div>
                <div class="spacer"></div>
                <div class="box">
                    <div>
                        <p class="info-text">Voit liikuttaa haarukkaa hiirellä tai kosketusnäytöllä
                        </p>
                    </div>
                    <br/>
                   <div>
```

```
                    <button class="button left" id="toggleRotateButton" type="button">
                        <span class="material-icons md-48 red left" id=rotateImg>
                            <img src="img/3d_rotation_black_48dp.svg" aria-hidden="true">
                        </span>
                    </div>


                </div>
            </div>
        </div>



        <script>
            var scene = new THREE.Scene();
            var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.inner-
Height,  0.1, 1000);
            var renderer = new THREE.WebGLRenderer({antialias:true});
            renderer.setSize( window.innerWidth, window.innerHeight );
            document.body.appendChild( renderer.domElement);

            var color_handle = 0xff1500;
            var colorPicker = new iro.ColorPicker('#picker', {layoutDirection:"verti-
cal", width:300,color: "#ff1500"});
            colorPicker.on('color:change', function(color) {
                model.traverse((o) => {
                    if (o.isMesh) {
                        if (o.name == "kahva") {
                            o.material.color.set(color.hexString);
                            console.log(color.hexString)
                        }
                        if ( o.name == "haarukkatest") {
                            o.material.color.set(0xcaccce );
                        }
                    }
                    });
                document.getElementById("hex-text").innerHTML = color.hexString;
            });

            document.getElementById("toggleRotateButton").addEventListener("click",
                function(event) {
                    console.log(event.target.innerHTML)

                    if (rotate) {
                        document.getElementById("rotateImg").classList.remove('red');
                        document.getElementById("rotateImg").classList.add('green');
                    }
                    else {
                        document.getElementById("rotateImg").classList.remove('green');
                        document.getElementById("rotateImg").classList.add('red');
                    }
                    rotate = !rotate;
```

```
    },
    false
);
var rotate = true;
window.addEventListener( 'resize', function()
{
    var width = window.innerWidth;
    var height = window.innerHeight;
    renderer.setSize( width, height);
    camera.aspect = width / height;
    camera.updateProjectionMatrix();
    colorPicker.resize(width*0.16);
    document.getElementById('picker').style.height =  Math.round(width*0.17)+'px';
});

//tausta
const loader = new THREE.TextureLoader();
const bgTexture = loader.load('kuvat/tausta.png');
scene.background = bgTexture;
//

controls = new THREE.OrbitControls( camera, renderer.domElement); //controllit


var loadery = new THREE.GLTFLoader();

var model;

//uusi materiaali
loadery.load( 'models/fork_model.glb', function ( gltf ) {

model = gltf.scene;
console.log(model);
model.traverse((o) => {
if (o.isMesh) {

    if (o.name == "kahva") {
        o.material = new THREE.MeshPhysicalMaterial({
            metalness: 0.6,
            roughness: 0.5,
            reflectivity: 0.0,
            envMap: null
        });
        o.material.color.setHex(color_handle);
    }
    if ( o.name == "haarukkatest") {

        o.metalness = 1.0;
        o.roughness= 0.0;
        o.material.color.set(0xcaccce );
        o.material.reflectivity = 1.0;
        o.material.envMap = null;
```

```
        o.material.clearcoat = 0.0;

    }

}
});

scene.add(model);

model.position.z = 7; //radians
model.position.x = 0;
model.position.y = -8;
model.rotation.z = 1.57;
model.rotation.x = 6.25;

}, undefined, function ( error ) {

console.error( error );

} );

camera.position.z = 20;

var ambientLight = new THREE.AmbientLight( 0xFFFFFF, 0.25);

const sphereSize = 1;

const frontLightLeft = new THREE.PointLight( 0xffffff, 0.5, 0,2 );
frontLightLeft.position.set( 45, 45, 180 );
scene.add( frontLightLeft );

const frontRightLight = new THREE.PointLight( 0xffffff, 0.5, 0,2 );
frontRightLight.position.set( -45, -45, 180 );
scene.add( frontRightLight );

const frontCenterLight = new THREE.PointLight( 0xffffff, 0.25, 0,2 );
frontCenterLight.position.set( -10, -0, 400 );
scene.add( frontCenterLight );

const backLight = new THREE.PointLight( 0xffffff, 0.7, 100,2 );
backLight.position.set( -5, 5, -18 );
scene.add( backLight );


var timeloop= 0;

//Renderöintilogiikka
var update = function()
{
    if (rotate) {
        model.rotation.y += 0.01;
    }
```

```
            };

            //Piirretään scene
            var render = function()
            {

                const canvas = renderer.domElement;
                const canvasAspect = canvas.clientWidth / canvas.clientHeight;
                const imageAspect = bgTexture.image ? bgTexture.image.width / bgTexture.image.height
                : 1;
                const aspect = imageAspect / canvasAspect;

                bgTexture.offset.x = aspect > 1 ? (1 - 1 / aspect) / 2 : 0;
                bgTexture.repeat.x = aspect > 1 ? 1 / aspect : 1;

                bgTexture.offset.y = aspect > 1 ? 0 : (1 - aspect) / 2;
                bgTexture.repeat.y = aspect > 1 ? 1 : aspect;

                renderer.render( scene, camera);

            }

            //Toistetaan visualisointilooppia ( update, render, repeat)
            var VisualizationLoop = function ()
            {
                requestAnimationFrame( VisualizationLoop);
                update();
                render();
            };

            VisualizationLoop();
        </script>
    </body>
</html>
```