PICQUER

Viihteellinen web-sovellus

Martti Halme

Opinnäytetyö
Joulukuu 2012

Mediatekniikan koulutusohjelma
Tekniikka ja liikenne

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) HALME, Martti Aleksi	Julkaisun laji Opinnäytetyö	Päivämäärä 11.12.2012
	Sivumäärä 43	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)
Työn nimi PICQUER – VIIHTEELLINEN WEB-SOVELLUS		
Koulutusohjelma Mediatekniikka		
Työn ohjaaja(t) NIEMI, Kari		
Toimeksiantaja(t) 3dboarding.de		

Tiivistelmä

Opinäytetyön aihe sai alkunsa toimeksiantajan ideasta yhdistää internetissä toimiva sosiaalinen kuvapalvelu, 3D-grafiikka ja pelielementti. Opinnäytetyön tehtävänä oli kartoittaa kaupallista potentiaalia omaavan idean toteutusmahdollisuuksia teknologisesta näkökulmasta.

Teknisessä toteutuksessa käytettiin PHP:tä, JavaScriptiä sekä ulkoisia WebGL -tekniikkaa toteuttavia kirjastoja. Tietokannanhallintajärjestelmänä toimi MySQL. Merkittävässä roolissa toteutetun prototyypin osalta oli Three.js niminen JavaScript-kirjasto, jolla 3D-grafiikan tekeminen WebGL-tekniikkaan pohjautuen oli yksinkertaisempaa ja tehokkaampaa.

Tuloksena syntyi ensikäsitys sovellusidean teknisestä toteutuksesta prototyypin muodossa. Tämän lisäksi toimeksiantajalle saatiin luotua sovelluksen jatkokehitykseen, sekä lopputuotteeseen liittyviä edellytyksiä.

Toimeksiantaja oli hyvin tyytyväinen lopputulokseen. He kokivat, että mahdollisia tulevia sijoittajia tai sidosryhmiä ajatellen, palvelee toteutettu prototyyppi paremmin kuin pelkkä Powerpoint-esitys. Picquer prototyyppi tekee ideasta vakavasti otettavamman mahdollisten tulevien yhteistyökumppaneiden silmissä.

Avainsanat (asiasanat)

WebGL, Javascript, Three.js, avoin lähdekoodi, ohjelmistokehitys, prototyyppi

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Author(s)	Type of publication	Date
HALME, Martti Aleksi	Bachelor´s Thesis	11.12.2012
	Pages 43	Language Finnish
		Permission for web publication (X)

Title
PICQUER – RESHAPING THE PLANET

Degree Programme
Media Engineering

Tutor(s)
NIEMI, Kari

Assigned by
3dboarding.de

Abstract

The idea for the bachelor's thesis originated from the company's concept of combining a social photo service with 3D graphics and elements of gaming. The aim of the thesis was to map the idea's technical possibilities and commercial potential from technological perspective.

In the technical execution PHP, JavaScript, WebGL and external libraries such as Three.js and jQuery were used. 3rd party JavaScript library Three.js played the most significant role in the creation of the prototype to enable its advanced and effective methods for making 3D-graphics in the browser. MySQL was chosen as the database management system of the prototype.

The outcome of the thesis was the early understanding of the technical prerequisites needed in a form of a created prototype. For the company, additional preconditions were also brought up regarding the further development and the future end product of the Picquer web service.

Keywords
WebGL, Javascript, Three.js, open source, software development, prototype

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

KÄSITTEET

API	*Application Programming Interface*, eli ohjelmointirajapinta.
Asiakaspuolen teknologia	*Client-side technology*. Internet -teknologia, jolla tarkoitetaan paikallisesti, käyttäjän laitteistossa, suoritettavaa sovellusta tai tekniikkaa. Asiakaspuoli voi kommunikoida palvelin puolen kanssa ja toisinpäin. Kts. myös Palvelinpuolen teknologia.
Canvas	Canvas on HTML5-elementti, jolla voidaan selaimessa tuottaa grafiikkaa usein JavaScriptin avulla.
DOM	*Document Object Model* on ohjelmointirajapinta, joka mahdollistaa HTML-dokumenttien sisällön muokkauksen.
GPU	*Graphical Processing Unit*, eli grafiikkapiiri. Prosessori, jonka laskentatehoa hyödyntämällä suoritetaan grafiikan piirtämiseen liittyviä toimintoja.
GLSL	*OpenGL Shading Language,* yksi varjostusohjelmointikielistä. GLSL:n avulla päästään käsiksi renderöintiputkeen. Kts. käsite renderöintiputki.
HTML5	Merkkauskieli, jolla esitetään sisältöä WWW:ssä. HTML5 on opinäytetyön kirjoitushetkellä HTML-kielen viimeisin ja uusin versio.
Palvelinpuolen teknologia	*Server-side technology.* Internet -teknologia, jolla tarkoitetaan palvelimella, käyttäjän laitteiston ulkopuolella suoritettavaa sovellusta tai tekniikkaa. Palvelinpuoli voi kommunikoida toisten palvelinpuolen teknologioiden ja asiakaspuolen teknologioiden kanssa. Kts myös Asiakaspuolen teknologia.
Renderöinti	3D-tietoa sisältävän kuvan piirtäminen.
Renderöintiputki	*Rendering pipeline*. Grafiikkapiireissä oikeatyyppisen datan läpikäyminen renderöitiputkessa tuottaa lopulta renderöidyn 2D kuvan, joka voidaan näyttää esimerkiksi tietokoneen näytöllä.

Shader	Varjostin. 3D-grafiikan renderöinnissä käytetty ohjelma, jonka tehtävänä on tarkemmin kuvata näytettävää kohdetta.
UI	*User Interface*, eli käyttöliittymä.
Web-sovellus	Web-sovellus on mikä tahansa ihmisen käyttämä sovellus, joka sijaitsee palvelimella. Käyttäjälle näytetään erillisen käyttöliittymän kautta internet-sivulla informaatio. Kaikki sivun tieto sijaitsee palvelimella.
Web-palvelu	Web-palvelu liittyy palvelinten väliseen tekniikkaan, jossa itse palvelu kommunikoi toisen palvelimella olevan palvelun kanssa. Tyypillisesti web-palvelun toiminta ei näy loppukäyttäjälle käyttöliittymän lävitse vaan toiminta tapahtuu taustalla. Esimerkiksi palvelinpuolen ohjelmointikieli PHP hakee ja käsittelee XML tai JSON-formaatissa olevaa informaatiota. Loppukäyttäjälle voi näkyä PHP:n palauttamaa jäsenneltyä informaatiota.

1 TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

Toimeksianto ja idea Picquer-palvelusta syntyi nykyisin Helsingissä asuvalta saksalaiselta Steffen Halmeelta (TaM). Steffen Halme työskentelee nykyisin käyttöliittymäsuunnittelun asiantuntijatehtävissä Linja Design Oy:llä. Steffen Halme on työskennellyt aikaisemmin mm. Rovio Mobile Ltd.:ssa. Opinnäytetyö on toteutettu 3dboarding -nimiselle yritykselle, jonka omistaa Steffen Halme. Kirjoitushetkellä oli vielä epävarmaa, että tullaanko projektin tiimoilta perustamaan myös oma start-up yrityksensä, jossa Steffen Halmeen lisäksi liiketoimintamallin kehittäjänä toimii hänen vaimonsa Anu Halme (YTM). Anu Halme on toiminut aiemmin mm. Business Development Managerina Riot Entertainment Ltd.:ssa sekä Sales Directorina HammerKit Oy:ssä.

1.2 Sovelluksen idea ja liiketoimintamalli

Idean nimi web-sovellukselle on *PICQUER- Reshaping the planet* (myöhemmin *Picquer)*. Picquer on viihteellinen web-palvelu, jossa käyttäjien omat, paikkatiedolliset valokuvat sijoittuvat korttipakkoina 3D-karttapallolle. Käyttäjät valloittavat valokuvillaan maailman uudelleen kuten entisajan tutkimusmatkailijat.

Verkkopalvelujen liiketoiminta- ja ansaintamalli on yksi alan kuumimpia kysymyksiä, jota monella taholla mietitään. Loppukäyttäjien valmius maksaa digitaalisista palveluista suoraan on yleensä aika alhainen, muutamia poikkeuksia lukuunottamatta (esim. verkkopankki, konserttilippujen toimitus, lailliset musiikkipalvelut kuten iTunes). Yleisimmät liiketoimintamallit ovat joko mainosrahoitteisuus, freemium-mallit, joissa muutamat premium-käyttäjät maksavat, tai sitten kulut ja mahdollinen tuotto on leivottu jonnekin muualle kuten jäsen- tai lupamaksuihin.

3dboardingin arvion mukaan Picquer-palvelulle sopivimmat liiketoimintamallit ovat joko

1) freemium-malli

2) markkinointisovellus-malli, jossa ensin hankitaan mahdollisimman paljon käyttäjiä ja myöhemmin pyritään myymään palvelu jollekin toiselle yritykselle markkinointityökaluksi.

3) mainosrahoitteisuus tai

4) kännykkäsovelluksen myynti sovelluskaupoista.

Freemium-mallissa peruskäyttö on ilmaista, mutta laajemmasta käytöstä kuten vaikkapa omasta, ei-julkisesta valokuva-albumista peritään maksu. Tällaisella suljetulla kuvatilillä voi dokumentoida vaikkapa oman maailmanmatkan.

Markkinointisovellus-mallissa ostajakohderyhmänä ovat toiset verkkopalvelut, joiden profiiliin Picquer sopii. Tällaisia olisivat esim. matkapalvelut kuten eBookers, Skyscanner tai hotels.com tai valokuvaukseen liittyvät palvelut kuten Kodakin tai Ifolorin kuvakaupat. Kolmas sopiva kohderyhmä ovat karttapalvelut kuten Nokia Maps.

Teknisesti tämä vaihtoehto saattaa aiheuttaa myöhempiä järjestelmäintegraatioita, joihin olisi hyvä varautua jo tietokantojen määrittelyvaiheessa valitsemalla mahdollisimman joustava tietokantarakenne.

3dboarding näkee mainosrahoituksen mielekkääksi vaihtoehdoksi nimenomaan paikallistasolla, koska gps-kordinaatit mahdollistavat tarkan kohdennukset. Sopiva mainonnan muoto voisi olla vaikkapa paikallisen rantaravintolan sponsoroitu kuvakortti Picquer-pakassa.

Älypuhelin on todennäköisesti se laite, jolla suurin osa Picquer-palveluun syötettävistä kuvista otetaan. Niin sanottu natiivi, ladattava ja maksullinen kännykkäsovellus sopisi näin ollen hyvin palvelun luonteeseen. 3dboarding selvittää mahdollisuuksia rakentaa natiivi sovellus, joka käyttäisi samaa dataa web-sovelluksen kanssa. Tällä hetkellä suurimmat sovelluskaupat ovat Applen AppStore, Google Play Android-sovelluksille ja Nokia Store.

Myös näiden yhdistelmä on mahdollinen. Lopulta toteutettava liiketoimintamalli kirkastuu talven 2012-13 rahoittajien ja yrityskummien kanssa käytävissä keskusteluissa ja eri start-up-yrittäjyystapahtumissa. (Halme 2012a.)

1.3 Tausta ja tavoitteet

Opinnäytetyön taustana oli Steffen Halmeen idea web-palvelusta, jossa suosittujen kuvapalveluiden lisäksi yhdistyisi pelaamisen ilo.

Hän lähestyi minua elokuussa 2012 tällä ajatuksella ja totesimme, että idea on omaperäinen ja ehdottomasti tutustumisen arvoinen.

Opinnäytetyön alkaessa oli vielä epäselvää mille yritykselle toteutus tehtäisiin. Toimeksiantajan harkitsi aluksi idean myymistä sellaisenaan kolmannelle osapuolelle, mutta päätyi lopulta kuitenkin pitämään idean web-palvelusta itsellään.

Tarkoituksena oli yhdessä jalostaa konseptia ideasta lähemmäksi toteutuskelpoista web-palvelua. Alusta lähtien oli selvää, että resurssit tulisivat olemaan rajalliset niin toimeksiantajan kuin opinnäytetyön tekijän osalta. Resurssien puutteilla tarkoitetaan tässä sekä aikataulullisia että osaamiseen liittyviä haasteita.

Opinnäytetyön tekijän omat tavoitteet olivat media-alan ammattilaisilta uuden oppiminen sekä toteutettavan palvelun kehittämistä vaativa tutkimustyö. Tavoitteena oli myös luoda toimiva prototyyppi tai teknologiademo.

1.4 Aiheen rajaus

Picquer on konsepti viihteellisestä web-sovelluksesta, ja vastaavanlaista sovellusta ei ole olemassa. Opinnäytetyön kiinnostavin ja myös samalla haastavin vaihe oli rakentaa jotain sellaista, mitä ei ole aikaisemmin toteutettu. Opinnäytetyö olisi voitu kirjoittaa sovelluksen kaupallisesta potentiaalista ja sen teoreettisesta sisällöstä. Käytettävissä olevien resurssien puitteissa ja opinnäytetyön tekijän omien preferenssien myötä pidettiin fokus voimakkaasti todellisten teknisten haasteiden ratkaisemisessa.

1.5 Muut kuvapalvelut

Sosiaaliset kuvapalvelut nauttivat suurta suosiota internet-käyttäjien keskuudessa. Tunnetuimmista sosiaalisista kuvapalveluista 2000-luvulla ensimmäisiä oli Photobucket. Photobucket tarjoaa palveluun rekisteröityneelle 500 gigatavua (Photobucket 2012). Photobucketilla oli lehdistötiedotteensa mukaan vuonna 2011 noin 100 miljoonaa käyttäjää sekä kahdeksan miljardia palveluun siirrettyä kuvaa tai videota (Imaging resource 2011). Samoihin aikoihin Photobucketin kanssa syntynyt Flickr-kuvapalvelu myytiin Yahoolle 35 miljoonan dollarin hintaan. Rahallisesti arvokkain kuvapalvelun myynti tapahtui vuoden 2012 huhtikuussa Facebookin ostaessa Instagram-nimisen kuvapalvelun noin miljardilla dollarilla. (Primac, D. 2012). Facebook lasketaan myös kuvapalveluksi vaikka se ei siksi profiloidukkaan. Facebookiin siirretään joka sekunti noin 3000 valokuvaa. Per päivä tämä tekee noin 250 miljoonaa valokuvaa (Anson Alex 2012). Facebookin vahvuus on ehdottomasti sen helppous ja yksinkertainen tosiasia, että lähes kaikki käyttävät sitä. Suosion myötä kaupallinen potentiaali kuvapalveluissa on valtava.

Internetissä on lisäksi lukuisia muita erilaisia sosiaalisia kuvapalveluita, joiden päämäärä on mahdollistaa käyttäjän ottamien valokuvien ja videoiden jakaminen verkossa. Jotta palvelusta saadaan sosiaalinen, on palvelun käyttäjillä lisäksi mahdollisuus kommentoida ja jakaa kuva tai video edelleen muiden kesken. Suosituimmat kuvapalvelut eroavat varsin vähän toisistaan ominaisuuksillaan. Erityisesti tänä päivänä merkittävä ominaisuus on

määritellä kuva-albumin näkyvyys tietylle käyttäjäryhmälle. Toteutuksia kuvan edelleen jakamiselle eri palveluiden välillä on monia. Esimerkiksi Google Picasan ollessa vahvasti integroitu Google+ -palveluun, voidaan kuvat jakaa suoraan Google+:n nk. *circleihin*, ympyröihin. Picasa tarjoaa lisäksi hyvin kattavat mahdollisuudet muokata valokuvia ennen kuin käyttäjä siirtää ne muiden nähtäville. Monessa kuvapalvelussa voi käyttäjä antaa suoran linkin kuva-albumiinsa ilman, että kuvat muutoin näkyisivät kenellekkään. Lisäksi albumit voidaan salasanasuojata. Facebook on kuvapalveluna varsin yksinkertainen ja Picasan kaltaisia kuvan muokkauksia ei voida toteuttaa. Lisäksi Facebookissa käyttäjän tulee ymmärtää tarkkaan mihin hän kuvansa on jakamassa ja kellä kaikilla on oikeus kuvien tarkastelulle.

Picquer-sovelluksen kehitystyössä on syytä ottaa mallia jo markkinoilla olevista kuvapalveluista ja hyödyntää järkevästi niiden parhaimpia ominaisuuksia.

2 SOVELLUSKONSEPTI

2.1 Yleistä

Kuten luvussa 1.2 esiteltiin, oli sovelluksen idea toteuttaa uudenlainen kuvapalvelu, jossa yhdistyy sosiaalisena palveluna kuvien jakaminen sekä eräänlainen peliaspekti.

Opinnäytetyötä kirjoittaessa ei toimeksiantajan määrittely toteutettavasta tuotteesta ollut valmis. Toimeksiantaja oli määritellyt kuitenkin sovelluksen eräänlaiset ääriviivat. Hyvin löyhä toimeksianto toi mukanaan paljon haasteita, mutta myös suuresti vapautta toteutuksen suhteen. Tämä tosiasia ei kuitenkaan estänyt opinnäytetyöntekijää pohtimaan teknologiavalintoja ja prototyypin toteuttamista.

Kuviossa 1 on tekijälle toimitettu kuva halutusta tuotteesta. Kysymyksessä on ensimmäinen promootiokuva, jonka pohjalta myös prototyyppiä käytiin myöhemmin toteuttamaan.

KUVIO 1. PICQUER.com - Reshaping the planet -promootiokuva

Kuvassa on kolmiulotteinen maapallo, jonka päällä on kuvapinoja (palkkeja). Palkkien korkeus vaihtelee sen mukaan kuinka paljon kohteessa on kuvia. Tämä itsessään on yksi merkittävä asia tiedon esittämisen kannalta.

2.2 Sovelluksen ominaisuudet

Luvussa 2.2.1 on selvitetty Picquer-websovelluksen ideointivaiheen ominaisuuksia. Ominaisuudet ovat listattu erikseen kuvapalvelutoiminnallisuuden ja pelitoiminnallisuuden osalta.

2.2.1 Kuvapalvelunäkökulma

Palvelun ollessa sosiaalinen tulee siinä olla ajanmukainen käyttäjähallinta ja sisään -ja uloskirjautumismahdollisuus. Picquer-palveluun tulee pystyä kirjautumaan joko Facebook-tunnuksilla tai vaihtoehtoisesti omilla järjestelmän ylläpitämillä tunnuksilla.

Palveluun siirretty kuva kohdistetaan sovelluksen maapallolle aina perustuen kuvan GPS-paikkatietoihin. Käyttäjä pystyy lisäksi lisäämään valokuvia järjestelmään ottaen huomioon tyypillisimmät kuvaformaatit. Kuvan lisäyksen yhteydessä on mahdollisuus lisätä kuvateksti. Lisäksi käyttäjällä on mahdollisuus poistaa kuvia palvelusta. Tähän liittyy myös "hygieniaominaisuudet", jossa käyttäjäyhteisöllä olisi mahdollisuus ilmoittaa asiattomasta sisällöstä.

Tiedon havainnointi maapallolla ja näyttäminen tulee olla suodatettavissa. Näitä voisivat olla esimerkiksi vain omien kuvien, viimeisimpien palveluun lisättyjen kuvien, ensimmäisen kuvan ja kaikki alueella olevien kuvien näyttäminen. Muuta esitettävää tietoa ovat maapallolle muodostuvat kuvapinot sekä muut graafiset elementit kuten apukursori, reittiviivat ja säteet.

Käyttäjällä on mahdollisuus kontrolloida, kuinka kuvat maapallolla näytetään, esim. näytetäänkö kuvapino vai ainoastaan "nuppineula" karttapallolla.

Käyttäjällä tulee olla mahdollisuus pyörittää maapalloa sekä zoomata lähemmäksi ja kauemmaksi. Internetin käyttäjät ovat omaksuneet Google Maps ja Google Earth-kontrollit, joten näiden hyödyntäminen olisi loogista.

Facebook-integraatio voitaisiin huomioida *Tykkäys* -ja kommentointi toiminnallisuuksilla.

Mahdollisesti toteutettavia ominaisuuksia kuvatoiminnallisuuden osalta ovat seuraavat:

- Yhteistyö eri digitaalisten kuvien tulostamiseen keskittyvien yritysten kanssa. Näitä ovat esimerkiksi Ifi tai Kodak.

- Valokuvien siirtäminen palveluun sähköposti -ja MMS-protokollien avulla.
- Flicker-kuvapalvelusta tuttu suodatusmahdollisuus, jossa käyttäjä voi nähdä vain kavereiden yksityiset kuvat.

(Halme 2012c.)

2.2.2 Pelinäkökulma

Lisäarvoa palvelulle tuo ehdottomasti onnistunut peliaspekti. Picquer-web sovelluksen nimi juontaa englanninkielen sanoista *Picture,* kuva ja *Conquer,* valloittaa (Halme 2012c.). Pelin idea olisikin yhdistää kuvien siirtäminen palveluun ja maantieteellisten alueiden valloittaminen. Pelin henki olisi toteuttaa tutkimusmatkailua ja maapallon eri kolkkien saavuttamista ennen muita käyttäjiä. Mielenkiintoiset kohteet osoitetaan valloitetuksi palveluun siirretyn kuvan avulla.

Hyvä esimerkki sosiaalisten peli ja hyötypalveluiden joukossa on kuitenkin suosittu *Foursquare*-niminen mobiilipalvelu, jossa ainoa toiminto on kirjautua sisään eri kohteisiin (nk. *Check-in)*. Lisäksi Foursquare-palvelussa on mahdollisuus jakaa kanssakäyttäjille vinkkejä esimerkiksi hyvistä ruokapaikoista. Foursquare-palvelussa on alusta lähtien ollut pisteiden keräyspeli ja eräänlainen *pormestari*-peli. Käyttäjä saa pisteitä riippuen siitä, minkälaisiin kohteisiin sisäänkirjaudutaan. Pormestariksi pääsee, mikäli käyttäjä on riittävän usein kirjautunut yhteen ja samaan paikkaan viimeisen viikon aikana. Hyödyllisen sosiaalisen tietopainoitteisen palvelun ja pelin yhdistäminen ei ole siis millään tavoin ennenkuulumatonta ja suuri yleisö tähän on tottunut.

Sovelluksen peliosuuden toteuttaminen käytännössä sekä pelimekaniikan ideointi on iso erillinen kokonaisuus, jota tämä opinnäytetyö lukijalle hahmottaa User Stories -luvussa. Opinnäytetyön yhteydessä toteutettavan prototyypin osalta pelimekaniikkaa ei toteutettu.

2.3 User Stories

Web-palvelujen konseptia ja rakennetta on usein helpointa lähestyä ns. User Storyn kautta. Hyvät käyttäjätarinat voivat toimia myös toiminnallisuusmäärittelynä kehittäjälle. (Halme 2012b.)

2.3.1 User Story 1 - Kirjautunut käyttäjä

Käyttäjä kirjautuu palveluun Facebook- tai muilla Open ID -tunnuksillaan. Hän syöttää selaimessa omalta laitteeltaan GPS-paikkatiedollisen valokuvan. Virheilmoitus näkyy, mikäli tiedostossa ei ole paikkatietoa. Kuvaan on mahdollisuus lisätä korkeintaan 200 merkin teksti,

joka voi sisältää HTML:ää. Kun kuva on syötetty Picqueriin, se sijoittuu "korttina" 3D-karttapallolle. Jokainen lisätty kuva nostaa korttipakan tai muun graafisen pylvään korkeutta. Tarkoituksena on visualisoida, missä käyttäjät ovat käyneet.

Kirjautunut käyttäjä voi lajitella omat kuvansa ja hallinnoida niitä. Hallinnointinäkymän tulisi olla mahdollisimman lähellä eri kuvapalveluista tuttua tapaa, joissa pieniä näyttökuvia (thumbnail) voi siirrellä, kopioida ja poistaa ruudulla.

Kirjautunut käyttäjä voi lähettää omia kuviaan eteenpäin sähköpostilla tai johonkin yhteistyökumppanin palveluun. Kuvan voisi esimerkiksi kehittää paperikuvaksi tai muuksi kuvatuotteeksi, kuten T-paidaksi.

Kirjautunut käyttäjä voi lähettää omaan kuvatiliinsä kuvan myös MMS-viestinä tai sähköpostina. (Halme 2012b.)

2.3.2 User Story 2 - Vierailija web-palvelussa

Kuvien tarkastelu on mahdollista ilman sisäänkirjautumista. Vierailija voi hiirellään pyörittää 3D-karttapalloa ja klikata häntä kiinnostavia paikkoja. Vierailija voi valintansa pohjalta nähdä esim. 1-200 km:n säteellä olevat kuvakortit, ja ne tulevat näkyviin esimerkiksi karusellina.

Käyttäjä voi lajitella kuvia ainakin seuraavien kriteerien mukaan: uusin kuva, ensimmäinen kuva, kaikki alueen kuvat (alueen määrittely). Paikkatietoa ei voi manipuloida palvelussa jälkeenpäin.

Kaikki käyttäjät ja vierailijat voivat halutessaan antaa Facebook-tykkäyksiä ja kommentoida kuvia esimerkiksi käyttäjän seinällä. Näiden toimintojen näkyvyys itse Facebookissa riippuu käyttäjän omista Facebook-asetuksista. Vierailija voi lisäksi raportoida asiattomasta kuvasta. (Halme 2012b.)

2.3.3 User Story 3 - Pelillisyys ja tuloslistat

Tärkein motivaatio käyttää Picquer-palvelua on kilpailu siitä, kuka on jossain paikassa ensimmäinen tai käynyt siellä useimmin. Paljon matkustavat voivat kilpailla siitä, kuka on käynyt useammassa paikassa. Näin palvelua voidaan käyttää samalla tavalla kuin toimiston seinällä olevaa paperista maailmankarttaa, johon työnnetään nuppineula jokaisen matkan merkiksi.

Palvelun etusivulla näkyvät Top 10 -listat käyttäjistä, jotka ovat lisänneet eniten kuvia. Jokainen syötetty kuva voisi antaa esim. 10 pistettä henkilökohtaiseen pistetiliin. Jos kuva poistetaan, pisteet vähenevät saman verran. (Halme 2012b.)

3 TOTEUTUSTEKNIIKAT JA TEKNOLOGIAT

Jotta sovellukseen tarvittavat ominaisuudet voitaisiin toteuttaa verkossa toimivaksi web-sovellukseksi, tarvitaan teknologioita. Teknologioita hyödynnetään eri työkaluilla ja toteutustekniikoilla. Sovelluksen ideaa käsiteltäessä havaittiin pian, että valmiissa web-sovelluksessa tultaisiin käyttämään suurta joukkoa erilaisia niin asiakas kuin palvelinpäädyn tekniikoita.

Luvuissa 3.1 - 3.3 käsitellään tarkemmin potentiaalisia teknologioita, joita toteutus vaatii.

3.1 3D-grafiikka

Alusta lähtien oli selvää, että toteutuksen tuli olla helppokäyttöinen ja toimia internet-selaimessa. Sovelluksen tuli myös sisältää kolmiulotteisia elementtejä, jotka ovat visuaalisesti näyttäviä ja sovelluksen käyttäjää luokseen houkuttelevia. Konseptin kannalta merkittäviä kolmiulotteisia elementtejä olivat maapallo ja tiedon visualisoinnin kannalta oleellista valokuvista koostuvat pylväät. Lisäksi sovelluksen piti pystyä toistamaan kaksiulotteista grafiikkaa (2D-grafiikka) 3D-grafiikan seassa, esimerkiksi näytettävien valokuvien muodossa.

3.1.1 3D-grafiikka internetselaimessa

3D-grafiikan tuottamisella (ja sen *upottamisella*, eng. *embedding*) käyttäjän internetselaimeen on ollut nyt noin 15 vuotta kestänyt värikäs historia. Vuonna 1997 julkaistiin VRML (Virtual reality Modeling Language) ISO-standardi, joka mahdollisti täysipainoisen kolmiulotteisen interaktiivisen vektorigrafiikan teksturointineen. Monet tuohon aikaan alaa seuranneet muistavatkin ensimmäiset *3D-chat* huoneet, jotka usein vaivoin toimivat tuolloin yleisillä modeemiyhteyksillä. VRML standardin kehitys jatkui myöhemmin X3D-nimisenä projektina. XML-pohjainen X3D standardin spesifikaatio julkaistiin vuonna 2005. (X3D FAQ 2005.) X3D oli valmistuessaan huomattavasti kypsempi tekniikka 3D-grafiikan tuottamiselle. XML-pohjaisuus helpotti kehittäjiä laajentamaan sovelluksiaan monipuolisimmaksi. Esimerkiksi tietokantayhteydet tai muut ulkoiset lähteet voitiin nyt ottaa käyttöön X3D-koodin tuottamisessa. (Bringing 3D to the Web 2002.)

Halu 3D-grafiikan esittämiselle selaimessa oli ilmeinen. Lukuiset kehittäjäyhteisöt ja mainostajat olivat 3D:stä selainikkunassa kiinnostuneita ja 2000-luvulla syntyikin varteenotettavia tekniikoita kuten Java 3D ja Flash. Suurimmat ongelmat eri tekniikoiden jalkauttamisessa oli yksinkertaisesti käytettävien laitteistojen suorituskyky. 3D-grafiikan suosion leviämisen tiellä olivat vielä laskentatehollisesti hitaat tietokoneiden prosessorit, muistit sekä matalakaistaiset tietoliikenneyhteydet.

Kehittäjäyhteisölle käännekohta 3D-grafiikan käytölle oli laitteiston grafiikkapiirin (GPU, Graphics processing unit) käytön mahdollistaminen internetselaimessa. Suurimmat selainvalmistajat vasta hiljattain toivat tuen GPU kiihdytetylle grafiikalle. Suurimpien selainvalmistajien listalla ovat Microsoftin Internet Explorer 9 (The Windows Blog 2011), Mozilla Firefox 4 (Firefox 4 gets yet another final test build release. 2011.) ja Google Chrome 10 (Tom's Guide- Web Life 2011). Kaikki julkaisivat GPU tekniikkaa tukevansa selaimensa maaliskuussa 2011.

Tähän asti 3D:n mahdollistavat standardit ja tekniikat edellyttivät erillisten lisäosien käyttöönottamista. Merkillepantavaa on, että tänä päivänä kynnys erillisten lisäosien asentamiselle pelkän sisällön näyttämisen vuoksi on käyttäjälle varsin korkea. Poikkeuksia ovat Adobe Flash, sekä Microsoftin Silverlight.

3D-tekniikoiden kehitys jatkuu selainympäristössä nyt kiihtyvällä vauhdilla. GPU-kiihdytys on edesauttanut erilaisten kehitysympäristöjen ja API -rajapintojen syntyä. Kehitysympäristöjen valikoima on tänä päivänä jo varsin kattava. Adoben Flash version 11 julkistamisen myötä kehittäjille tuotiin Stage3D:ksi kutsuttu kehitysympäristö. Microsoft tarjoaa puolestaan haastajana Silverlight-työkalua. Merkittävä kiinnostus internet-kehittäjäyhteisössä on noussut WebGL:ää kohtaan.

Sekä Stage3D, että WebGL tarjoavat omat rajapintansa ja pääsyn laitteistokiihdytetylle (GPU) tasolle.

Opinnäytetyössäni prototyyppi toteutettiin WebGL-tekniikalla.

Tämän luvun teksti oli luonnollisesti vain pintaraapaisu niistä vaiheista, jotka ovat mahdollistaneet 3D-grafiikan esteetttömän tuomisen jokaisen käyttäjän internetselaimeen. Esteettömyydellä tarkoitetaan poistunutta tarvetta asentaa tietokoneelle erillisiä lisäosia (plugineita) ja yksinkertaisesti koneiden suorituskyvyn kehittymistä pisteeseen, joka ei hidasta käyttäjän laitteistoa liiaksi. Koko kehityksen ekosysteemi, kehittäjäyhteisö ja selainvalmistajat, ovat oppineet hyödyntämään vasta nyt HTML5:n tuomia mahdollisuuksia.

3.1.2 WebGL

Lyhyesti WebGL on laitteistokiihdytetty kolmiulotteinen grafiikkastandardi, joka hyödyntää HTML5 standardin Canvas-elementtiä. WebGL (Web Graphics Library) on JavaScript API, joka mahdollistaa 2D ja 3D grafiikan piirtämisen (renderöinnin) WebGL -tekniikkaa tukeviin internetselaimiin ilman erillisiä laajennuksia. (Khronos Group n.d.).

WebGL standardi mahdollistaa sen, että GPU-kiihdytettyjä elementtejä voidaan käyttää muiden HTML-elementtien kanssa. WebGL sovellukset koostuvat ohjaavasta koodista, joka on toteutettu JavaScript-ohjelmointikielellä, sekä *shader*-koodista (GLSL-kieli), jota ajetaan käyttävän laitteiston grafiikkapiirillä (GPU). (Raportti WebGL-käyttöliittymän toteutuksesta 2010).

WebGL standardi ei ole opinnäytetyön kirjoitushetkellä vielä osa HTML5 standardia. WebGL standardin taustalla on voittoa tavoittelematon Khronos Group. Samaisen yhdistyksen käsialaa on myös yksi maailman yleisin 3D-grafiikkarajapinta nimeltä OpenGL (Khronos Group OpenGL n.d.). Khronos Group perusti vuonna 2009 WebGL työryhmän, jonka tehtävänä on tämän standardin suunnittelu ja kehittäminen. Työryhmä sai tukijoikseen heti alusta alkaen suurimmat selaintoimittajat kuten Mozillan, Googlen, Operan sekä Applen. Mainittakoot, että Microsoft ei ole ilmoittautunut tukemaan kyseistä standardia tai sen kehitystä. WebGL API pohjautuu saman yhdistyksen OpenGL ES (OpenGL Embedded Systems)-stardardiin. OpenGL ES luotiin tuomaan 2D-ja 3D-grafiikka sulautetuille järjestelmille, kuten matkapuhelimet ja pelikonsolit (Khronos Group OpenGL ES n.d.).

WebGL tekniikka on tällä hetkellä versionumerossaan 1.0, joka julkaistiin 3.3.2011. Mainittakoot, että WebGL toteutukset ovat tämän opinnäytetyön kirjoitushetkellä vahvasti kokeellisia.

WebGL API ja ulkoiset kirjastot

WebGL API:a kutsutaan matalan tason API:ksi. Tämä johtuen siitä, että WebGL mahdollistaa kehittäjän pääsyn suoraan käsiksi GPU:hun ja vieläpä varsin intuitiivisesti. 3D-grafiikan renderöinti käyttäjän näytölle on itsessään monimuotoinen tekniikka, jota tämä opinnäytetyö ei sen tarkemmin käsittele. Lyhyesti sanottuna WebGL:n avulla kehittäjä pääsee vaikuttamaan *renderöintiputkeen* (*eng. rendering pipeline*). Termi renderöintiputki on omaksuttu Timo Korkaman Powerpoint-esitelmästä nimeltä Varjostinohjelmointi (Korkama, T. n.d.). Renderöintiputkessa 3D-grafiikan peruspalikat (kuten verteksit, eng. *vertices* ja kolmiot, eng. *triangles*) käsitellään GPU:ssa verteksitaulukon (*eng. vertex array)*, verteksivarjostinten (*eng. vertex shader)*, rasteroinnin (*eng. rasterizing)* ja fragmenttivarjostinten (*eng. fragment shader)* kautta kuvapuskuriin (*eng. framebuffer)*. Kuvapuskuri on renderöintiputken lopullinen päämäärä, joka on muutakin kuin pelkkä 2D-kuva. Se pitää tässä vaiheessa sisällään väri-informaation, syvyystiedon, sapluunapuskurin (*eng. Stencil buffer)*, sekä läpinäkyvyyden (*eng. alpha)*. (Dev.Opera 2011).

WebGL:ää hyödyntävät, kehitystyötä tehostavat, kolmannen osapuolen API -kirjastot madaltavat WebGL:n oppimiskynnystä. Kehityskirjastoja oli kirjoitushetkellä lukuisia ja opinnäytetyössä keskityttiin yhteen kehittyneimpään sekä kehittäjäyhteisöltään aktiivisimpaan, Three.js-kirjastoon.

Three.js

Three.js on projekti, jonka päämääränä on kehittää samannimistä 3D JavaScript -kirjastoa. WebGL on itse asiassa vain yksi kolmesta renderöintitekniikoista, joita Three.js-kirjasto voi hyödyntää. Kaksi muuta ovat SVG ja Canvas. Three.js-projekti pyrkii vähentämään 3D-grafiikkaohjelmointiin liittyvää monimutkaisuutta olemalla oma luokkakirjastonsa laajentaen WebGL-rajapintaa. Opinnäytetyöhön toteutettu prototyyppi hyödynsi Three.js-kirjastoa.

Seuraavassa on taulukoitu Three.js-kirjaston tärkeimpiä luokkia ja metodeita. Taulukkojen tiedot perustuvat Three.js:n viralliseen dokumentaatioon. (Three.js Documentation 2012). Alla olevaan taulukkoon on valittu luokat, joiden avulla voidaan toimiva WebGL näyttämö toteuttaa.

TAULUKKO 1. Three.js-luokkia, jotka esiintyvät useimmissa Three.js-projekteissa

Luokka	Luokan esittely
THREE.Scene	Scene-luokka mahdollistaa näyttämön *pystyttämisen*. Näyttämölle sijoitetaan kaikki kaksi -ja kolmiulotteiset objektit käyttäjän nähtäville. Scene-luokan *add*-metodi lisää objektin näyttämölle.
THREE.PerspectiveCamera	Kameraluokkia on useita, mutta ko. luokka mahdollistaa perspektiivikameran käyttöönoton. Perspektiivikamerassa voidaan kameran sijainnin lisäksi määritellä *field of view,* näkökenttä.
THREE.WebGLRenderer	Three.js tukee useampia renderöintitekniikoita, joista WebGL on yksi. Tällä luokalla suoritetaan 3D-grafiikan renderöinti näytölle erillisen *render*-metodin avulla.

Yllä esiteltyjen luokkien avulla voidaan luoda toimiva WebGL-näyttämö. Jotta näytölle saadaan 3D-grafiikkaa, tarvitaan objekteja. Seuraavassa taulukossa esitetään muutamia Three.js-kirjaston omia primitiivisiä geometriaobjekteja.

TAULUKKO 2. Tyypillisiä Three.js-kirjaston geometrialuokkia.

Luokka	Luokan esittely
THREE.CubeGeometry	Mahdollistaa kuutiogeometrian luonnin. Parametreinä tälle voidaan antaa esimerkiksi sivujen koot sekä tahon segmenttien määrä. Segmentit jakavat kuution geometrian pienemmiksi kuutioiksi.
THREE.SphereGeometry	Mahdollistaa pallogeometrian luonnin.
THREE.PlaneGeometry	Mahdollistaa tasogeometrian luonnin. Taso (ts. pinta) on suorakulmio, jolla ei ole korkeutta.
THREE.TextGeometry	TextGeometry luokalla voidaan tehdä kirjaisimien avulla suoraan kolmiulotteisia geometrisiä kappaleita.

Taulukossa 3 esitellään muutamia objektien materiaalin luontiin liittyviä luokkia. Materiaali määrittelee kuinka eri objektin pinnat näkyvät käyttäjälle.

TAULUKKO 3. Tyypillisiä Three.js materiaaliluokkia.

Luokka	Luokan esittely
THREE.MeshBasicMaterial	Mahdollistaa objektin pinnalle yksinkertaisen materiaalin, jossa ei ole heijastuksia. Prototyypissä näkyvät korttipinot, palkit, käyttävät ko. luokkaa. Luokka mahdollistaa lisäksi ulkopuolisen tekstuuritiedoston lataamisen halutulle objektin pinnalle. Ulkopuolisten tekstuurien käyttö on mahdollista myös muissa materiaaliluokissa.
THREE.MeshLambertMaterial	Mahdollistaa Lambert-materiaalin käyttämisen. Lambert-materiaali reagoi ulkoisiin valonlähteisiin, mutta ei luo aitoa heijastusefektiä, eli nk. Specular highlighting-efektiä. Lambert-materiaali näyttää renderöitynä hyvin mattapintaiselta.
THREE.MeshPhongMaterial	Mahdollistaa Phong-materiaalin käytön. Phong-materiaali huomioidaan renderöinnissä eri tavalla kuin

	yllämainitut. Phong-materiaalin renderöinnissä huomioidaan objektin pinnan muoto sekä valonlähde ja sen sijainti. Lopputuloksena materiaalissa on nähtävissä aidon oloinen valon heijastus (Specular highlighting). Phong-materiaali sopii esimerkiksi muovin tai posliinin renderöintiin.

Opinnäytetyön sovelluksen prototyypissä hyödynnettiin Three.js-kirjastoa.

3.1.3 WebGL-selaintuki

WebGL-tekniikkaa hyödyntävien selainten tuki ei ole luonnollisesti täydellinen. Microsoft ei ole ottanut minkäänlaista kantaa WebGL:n tukemiselle selaimissaan tulevaisuudessa. WebGL:ää tukee osaksi tai kokonaan iso joukko työpöytä- ja mobiiliselaimia. Seuraavassa on listattuna kirjoitushetkellä tuetut selaimet.

Työpöytäselaimet:

- Mozilla Firefox (osittainen tuki)
- Google Chrome
- Safari (osittainen tuki)
- Opera (osittainen tuki)

Mobiiliselaimet:

- Firefox for Android
- Opera Mobile 12 (osittainen tuki).

(When can I use n.d.)

Selaimilla on lisäksi omat “mustat listansa” liittyen tuettuihin grafiikkapiireihin. Mozilla -selain esimerkiksi estää kaiken grafiikkakiihdytyksen NVIDIA-valmistajan grafiikkapiireiltä, mikäli käyttäjällä on joko liian vanhat grafiikka-ajurit tai vanhentunut käyttöjärjestelmä. Näin estyy myös WebGL:llä tuotetun grafiikan toistaminen.

Internetistä löytyvät *WebGL Stats* -palvelu esittää grafiikan keinoin, että WebGL:ää tukevien selainten osuus olisi 8.9.2012 noin 65%. (WebGL Stats 2012). Mainittakoot, että *When can I use WebGL*-sivuston mukaan tuki olisi noin 28%, osittainen tuki noin 25% ja yhteenlaskettu tuki noin 53%. *When can I use WebGL*-sivusto kertoo aloittaneensa WebGL tukitilastojen keräämisen elokuusta 2012 lähtien. (When can I use n.d.)

On huomioitava, että WebGL-tekniikan tuki ja käyttäjän näytönohjaimilla olevat grafiikan tuottamiseen liittyvät ominaisuudet ja puutteet ovat täysin eri asioita. Toisin sanoen, kaikki WebGL:llä toteutut grafiikkapiiriä hyödyntävät ominaisuudet ja tekniikat eivät toimi kaikilla WebGL:ää itsessään tukevissa selaimissa. Kyseinen prosentti ei sinäänsä anna koko kuvaa WebGL:n toimivuudesta johtuen näytönohjainten ja laitteistojen monimuotoisuudesta.

3.2 Merkkaus-ja ohjelmointikielet

Web-sovellukset toteutetaan HTML-merkkauskielellä. Picquerin tapauksessa käytetään HTML:n viimeisintä standardia, HTML5 -versiota.

3D -grafiikka on vain yksi osa sovelluksen teknologiakokonaisuutta ja lokeroituu asiakaspuolen teknologiaksi. Jotta 3D grafiikkaa voidaan käsitellä, tarvitaan esimerkiksi WebGL tekniikkaa. Ohjelmointikielinäkökulmasta WebGL-komennot kirjoitetaan JavaScript -kielellä.

Picquer on sosiaalinen palvelu ja edellyttää teknologialtaan luonnollisesti kommunikointia ulkopuolisten palvelinten kanssa. Lopullisessa tuotteessa tämän valinta on varmastikin ratkaiseva ja oleellinen. Yleisin palvelinpään ohjelmointikieli lienee PHP5, jota opinnäytetyössäni tule käyttämään. PHP voidaan upottaa suoraan HTML:ään ja omaa valmiit kirjastot mm. MySQL tietokantayhteyksille. Tietojen säilöminen palvelimella tulee olemaan välttämätöntä ja valmiissa tuotteessa tietoa on varmastikin suuri määrä. Prototyyppi -tarkoitukseen MySQL on sopiva, mutta lopullisen tuotteen osalta muiden tietokantojen hallintajärjestelmiä on syytä tutkia enemmän.

3.3 GPS eli Global Positioning System

Picquer -web -sovelluksen perusajatukseen kuului käyttäjän ottamien digitaalisten kuvien mukana tulevan paikkatiedon hyödyntäminen 3D-maapallolla. Käyttäjän paikkatiedot tulevat asiakaslaitteeseen GPS -satelliiteista.

GPS eli Global Positioning System on satelliitteihin perustuva navigointitekniikka. Parhaillaan maapallon ilmakehässä kiertää 24 GPS-satelliittiä, jotka ilmoittavat laitteen käyttäjän sijainnin karttapallolla. Tyypillisesti siviilikäyttöisen GPS signaalin tarkkuus maanpinnalla on 10-15 metriä. (Johnny Appleseed n.d.).

Picquerin tulisi pystyä poimimaan käyttäjän digikuvasta paikkatieto, tallentaa se tietokantaan ja hyödyntää paikkatietoa 3D-maapallolla sovelluksessa. Huomioitavaa on myös koordinaattipohjaisten tietojen käsittely matemaattisessa mielessä. Maanpäälliset koordinaatit koostuvat pituuspiiristä (*eng. longitude)* ja leveyspiiristä (*eng. latitude)*. On

kriittistä pystyä laskemaan tietojen perusteella myös etäisyyksiä maapallolla. Maanpäällisten koordinaattien välistä etäisyyttä tarvitaan esimerkiksi tutkittaessa käyttäjän valintoja 3D-maapallolla. Esimerkki tästä voisi olla mahdollisuus listata kaikki lisätyt kohteet lähimmän 20 kilometrin säteellä. Maanpäällisten koordinaattien välisiä etäisyyksiä voidaan laskea nk. haversinin (*eng. haversine)* funktiolla hyödyntäen tietoa leveys -ja pituuspiiristä (Movable Type Scripts n.d.).

Käyttäjän laitteen GPS tietoja voidaan vääristää tarkoituksenmukaisesti. Sovelluksen käyttäjä voisi siis "huijata" sijaintinsa toisaalle, vaikkapa hyvin harvinaiseen ja eksoottiseen paikkaan. Tämä on hyväksyttävä ja mahdotonta täysin estää - mutta toisaalta on muistettava, että kysymyksessä on kuitenkin viihteellinen verkkopalvelu.

4 IDEASTA PROTOTYYPPIIN JA PROTOTYYPIN TEKNINEN TOTEUTUS

4.1 Prototyypin tehtävä

Prototyypin luominen idean pohjalta kuului olennaisena osana opinnäytetyöhön.

Kysymykseen, että miksi prototyyppi tehdään, esittää John Clark blogi-kirjoituksessaan kiteytetysti ohjelmistosuunnittelun näkökulmasta vastauksen asiaan. Suunnittelu -ja toteutustyö sovelluskehityksessä on ongelmantäyteinen. Sovelluksen toteuttaminen on vaikea työ saada onnistumaan täydellisesti. Suunnittelijoilla tulee olla ymmärrys ongelmasta myös bisneksen kannalta, jota toteuttava työ tähtää ratkaisemaan. Sovelluskehittäjät ja suunnittelijat ovat usein "tuomareina" rauhoittelemassa eri osapuolien välisessä kädenväännössä sitä, mitä ja miten oikeastaan ongelma ratkaistaan. Jokaisella sidosryhmällä on oma päämääränsä sekä näkemyksensä sen saavuttamisesta. Tällaisen määrittelyvaiheen läpikäyminen on jo itsessään "miinakenttä" ja liian usein siirrytään hätiköidysti kehitystyössä eteenpäin. Vaatimukset toteutettavalle järjestelmälle eivät ole riittävän yksiselitteisiä. Väärinkäsitys sidosryhmien sisällä vallitsee helposti ja toteutettavat ominaisuudet eivät ole riittävän pitkälle pohdittuja.

Prototyyppi itsessään on erilaisten mallien luomista, joka helpottaa ihmistä hahmottamaan jotakin, tässä tapauksessa ohjelmistosuunnittelun osalta. Paperilla kaikki, paitsi yksinkertaisemmat suunnitelmat, ovat ihmiselle hankala ymmärtää ja hahmottaa. Ihmiset ovat yleisesti ottaen huonoja hahmottamaan ongelmaa päässään, joka pohjautuu kirjoitettuun tekstiin. Ymmärrämmekin paljon tehokkaammin ongelman, mikäli voimme käyttää omia silmiämme ongelman hahmoittamiseksi. Ilman, että rakennamme jotain valmiiksi, voimme prototyypin avulla saada erittäin hyvän käsityksen siitä, mitä ollaan luomassa. Tässä piilee prototyypin voima.

Sovelluskehityksessä me emme voi fyysistä, kosketeltavaa, pienoismallia luoda, mutta sama asia kuitenkin pätee tähän: visuaaliset esimerkit ja mallit tulevasta auttaa katsojaa ymmärtään. (Clark, J. 2009)

4.2 Picquer prototyyppi

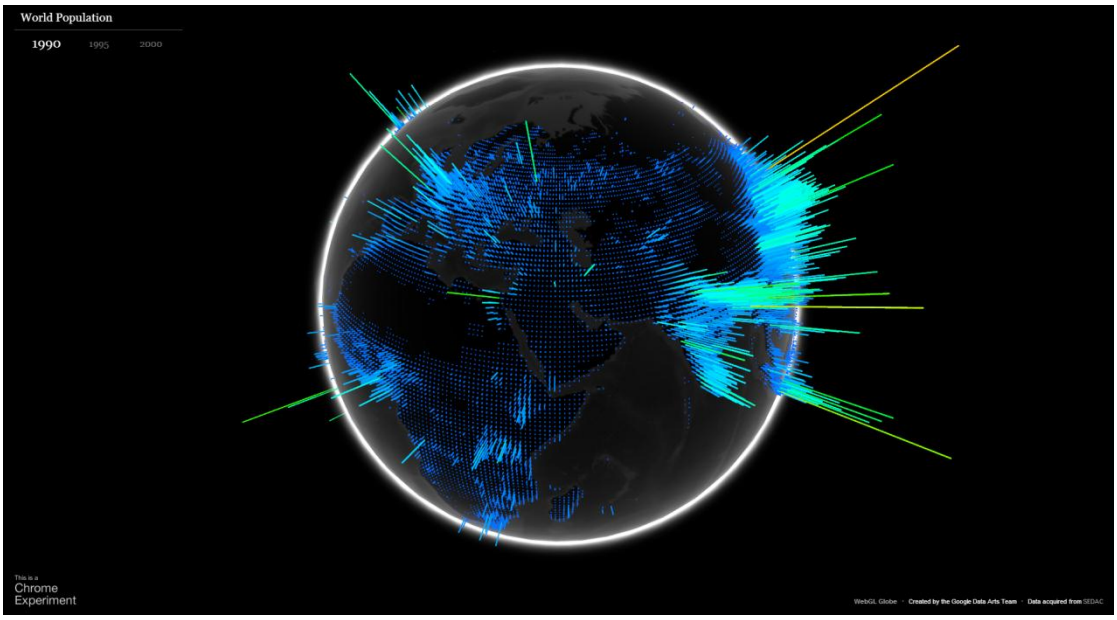
Opinnäytetyön kirjoittajalle prototyypin tekeminen oli suuri haaste, jota edelsi epävarmuus ohjelmointitaidoista ja valittujen tekniikoiden käyttöön liittyvä osaamattomuus. Hyvin pian toimeksiannon saamisesta ja idean kuulemisesta aloitettiin prototyypin työstäminen ilman tarkkaa tietoa siitä, mitä sovellus tulisi pitämään sisällään. Prototyypin aloittaminen näin pian

oli kuitenkin perusteltua, koska selvää oli, että lopullisessa tuotteessa tulisi olla 3D-karttapallo eri web-teknologioilla toteutettuna. Tämä yksittäinen ominaisuus kannusti aloittamaan nopeasti selvitystyön mahdollisuuksista 3D-grafiikkaan internetselaimessa. Kuten luvussa 3.1.1 todettiin, näyttävä, sujuva, aidosti kolmiulotteinen grafiikka jokaisen käyttäjän internetselaimessa ei ole vanha asia.

3D-maapalloista ja karttapalveluista internetselaimessa tunnetuin lienee Googlen toteuttama Google Earth -palvelu. Huomioitavaa on, että kyseinen palvelu käännettiin toimimaan merkittävästi suorituskykyisemmän WebGL -teknologian päälle vasta loppuvuonna 2011 (Google Earth Blog, 2011).

Toinen kiinnostava 3D-maapallototeutus on WebGL-tekniikalla toteutettu WebGL Earth. Kirjoitushetkellä kyseinen avoimeen lähdekoodiin pohjautuva 3D-karttapallo on beta-vaiheessa. Mielenkiintoista WebGL Earth -palvelussa on myös vapaaseen lähdekoodiin perustuvat erilaiset karttavaihtoehdot kuten Open Street Maps (Open Street Maps, 2012).

Varsinaisen kimmokkeen prototyypin aloittamiselle löytyi kuitenkin Chrome Experiments (Chrome Experiments, n.d.) sivustolta. Sivustolla oli projekti nimeltä The WebGL Globe (The WebGL Globe 2012), jonka on toteuttanut Google Data Arts Team. Ruutukaappaus kyseisestä projektista näkyy kuviossa 2.



KUVIO 2. The WebGL Globe. Toteuttanut Google Data Arts Team.

Heidän toteuttama projekti oli hyvä lähtökohta myös omalle toteuttamalleni prototyypille. He käyttivät projektissaan kiinnostavaa WebGL -tekniikkaa, toteutusta helpottavaa Three.js -

kirjastoa, sekä JSON -tyyppistä dataa visualisoidessaan dataa näyttävästi omiksi palikoikseen karttapallolla.

Kyseinen WebGL Globe demonstraation idea oli visualisoida käyttäjälle ihmisten määrää eri puolilla maapalloa korkeudellaan vaihtelevilla pylväillä.

WebGL Globen koodi oli tarkoituksella kaikille nähtävillä ja opiskeltavissa. Kyseisen totetutuksen löytyminen antoi Picquer-prototyypin toteutuksen aikaisessa vaiheessa uskoa siihen, että projekti on opinnäytetyön kirjoittajallekin mahdollinen.

Seuraavissa luvuissa selvitetään kuinka prototyyppi rajattiin ja minkälainen rajauksen suhde olisi valmiin sovelluksen välillä. Tämän jälkeen selvitetään lukijalle prototyypin yleinen rakenne käyttöliittymän, tietokantayhteyden sekä GPS-informaation käsittelyn osalta. Näissä luvuissa viitataan prototyypin rakentamiseen liittyneisiin teknisiin sekä teknologisiin valintoihin, joita jälleen myöhemmissä luvuissa tarkennetaan edelleen. Luvussa 4.7 Prototyypin ohjelmakoodi esitetään kooditasolla prototyypin teknisen toteutuksen tärkeimmät kohdat.

4.3 Picquer-prototyypin rajaus

Valmis Picquer-sovellus tulisi toimimaan useassa päätelaitteessa. Päätelaitteita, joilla Picquer-sovellusta voitaisiin käyttävää ovat erilaiset älypuhelimet ja tabletit. Yksi päätelaite on luonnollisesti tavallinen työpöytäselain. Prototyyppi toteutettiin toimimaan ainoastaan työpöytäselaimessa. Eri kannettavien laitteiden tuomia haasteita kuten mm. käyttöliittymäsuunnitteluun ja laitteiden väliseen suorituskykyyn liittyviä ongelmia ei ratkaistu.

Tietokantasuunnittelu tämän mittakaavan kuvapalvelussa on merkittävä osa-alue. Valmiin tuotteen tietokanta tarjoaisi tukirangan käyttäjähallinnalle ja valtavalle määrälle kuvainformaatiota. Tämä toisi mukanaan myös palvelintilaan liittyvät haasteet. Tietokantahallintajärjestelmän sekä relaatiotietokannan tai oliotietokannan välinen valinta olisi lopullisessa tuotteessa myös kriittinen. Prototyyppiä varten toteutettiin yksinkertainen MySQL relaatiotietokanta, ilman käyttäjähallintaa. Prototyypin tietokanta koostui yhdestä taulusta, sekä helposti käsiteltävissä olevasta määrästä kenttiä. Näiden avulla pystyttiin kuitenkin toteuttamaan esimerkki tiedon havainnollistamisesta karttapallolla.

Picquer-prototyyppi toteutettiin toimimaan Google Chrome selaimella. Chrome-selainta pidetään parhaiten WebGL-tekniikkaa toistavana selaimena (ConceivablyTech 2011). Useamman selaimen tuki ei prototyyppivaiheessa nähty olennaiseksi.

Kiteytetysti voidaan sanoa, että toteutettu prototyyppi oli käytännössä teknologiademonstraatio GPS-informaation käytöstä, sen käsittelyssä sovellusprototyypissä ja sen visualisointi 3D-karttapallolla. Tämän ongelman ratkaiseminen prototyypin avulla oli alusta alkaen toimeksiantajalle ja opinnäytetyöntekijälle merkittävin.

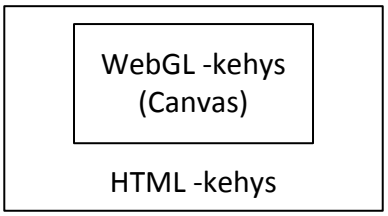
4.4 Picquer-prototyypin rakenne

Seuraavissa alaluvuissa esitellään Picquer-prototyypin rakenne yksinkertaisiksi paloiksi purettuna. Luvuissa viitataan prototyypissä käytettyihin tekniikoihin, joista kerrotaan myöhemmin luvussa 4.5.

4.4.1 Käyttöliittymän rakenne

Käyttöliittymä mahdollistaa käyttäjän pääsyn sovelluksen toiminnallisuuksiin.

Picquer-prototyypin käyttöliittymä koostuu kahdesta kokonaisuudesta, eli kehyksestä. Jako on looginen erilaisten teknisten lähtökohtien vuoksi. Käyttöliittymän rakenne on esitetty yksinkertaistetusti kuviossa 3.



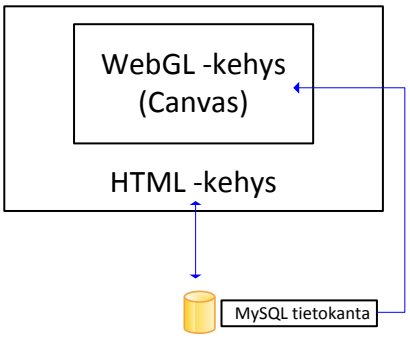
KUVIO 3. Käyttöliittymän perusrakenne.

Tyypillisen HTML-sivuun, omana elementtinään, toteutettiin HTML5:en Canvas-elementtiä hyödyntävä WebGL-kehys. WebGL-kehys on oikeastaan näkymä, jonka tehtävänä on esittää sille syötettyä informaatiota. WebGL-kehys näytetään iframe HTML-elementin sisällä.

Tämän perusrakenteen päälle toteutettiin käyttöliittymätoiminnallisuus ja tiedon näyttämisen visualisointiin liittyvät, voimakkaasti mm. jQuerya hyödyntävät, elementit.

4.4.2 Tietokantayhteyden rakenne

Prototyypin tietokantana käytettiin MySQL-tietokantaa. Kuviossa 4 on esitetty tietokannan suhde käyttöliittymäkehyksiin.

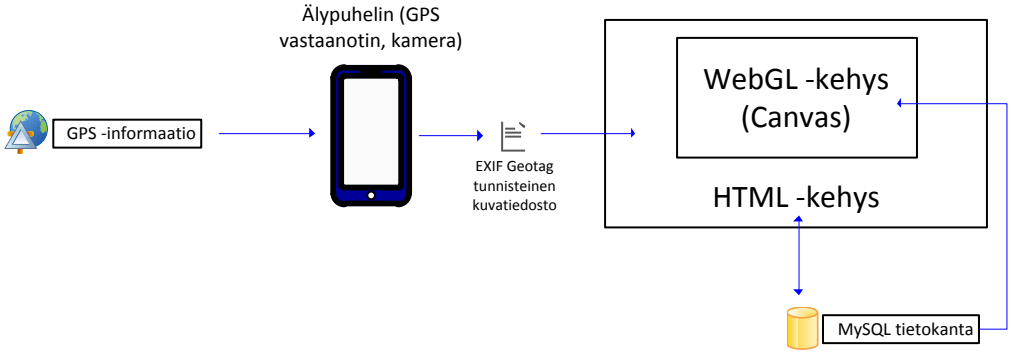


KUVIO 4. Tiedon liikkuminen tietokantayhteys huomioiden

Kaaviossa olevat nuolet kuvaavat tietokantayhteyttä ja tiedon kulun suuntaa kehysten välillä. HTML-kehyksen kautta syötetään tietokantaan tietoa ja palautetaan sitä. WebGL-kehyksessä tietokannan tietoa ainoastaan esitetään tietokantakyselyiden avulla.

4.4.3 GPS-tieto prototyypin rakenteessa

Picquer-prototyyppiin toteutettiin mahdollisuus käsitellä GPS-informaatiota sisältävän kuvan EXIF tunnistetta (eng. Exchangeable image file format header). Kuviossa 5 laajennetaan yllä esiteltyä rakennetta.



KUVIO 5. GPS-tiedon liikkuminen prototyypissä WebGL-kehykseen.

Kuvio esittää älypuhelimen avulla tuotetun GPS-informaatiota sisältävän kuvatiedoston siirtymistä HTML-kehykseen. HTML-kehyksessä tieto käsitellään, välitetään tietokantaan ja lopulta visualisoidaan WebGL-kehyksessä. Kuten edellisessä kaaviossa, HTML-kehyksen ja tietokannan välillä tietoa käsitellään kaksisuuntaisesti.

Prototyypissä toteutettiin sovelluksen ominaisuuksina mahdollisuus siirtää palveluun GPS-informaatiota sisältävä kuva työpöytäselaimessa. Prototyypin kannalta merkityksetöntä oli tapa ja laite, kuinka EXIF-tieto tiedostoon on tullut. EXIF-otsikon geoinformaatio selvitetään HTML-kehyksessä, talletetaan tietokantaan ja esitetään WebGL-kehyksessä.

4.5 Valitut tekniikat

Tämä luku esittelee lukijalle tarkemmin prototyyppiin valittuja tekniikoita. Jako tekniikoiden esittelyssä on tehty asiakas -ja palvelinpuolen teknologioiden välillä. Asiakaspuolen teknologialla (eng. *client-side technology*) tarkoitetaan paikallisesti, käyttäjän laitteistossa, suoritettavaa sovellusta tai tekniikkaa. Palvelinpuolen teknologioilla (eng. *server-side technology*) vastaavasti tarkoitetaan palvelimella, käyttäjän laitteiston ulkopuolella suoritettavaa sovellusta tai tekniikkaa. Palvelinpuoli voi kommunikoida toisten palvelin -sekä asiakaspuolen teknologioiden kanssa.

4.5.1 Asiakaspuolen teknologiat

JavaScript

JavaScript on järjestelmäriippumaton ja kevyt olio-ohjelmointikieli. JavaScript ei itsenäisenä ohjelmointikielenä ole erityisen hyödyllinen ellei sitä käytetä upotettuna muiden tuotteiden ja sovellusten kanssa. Näitä ovat esimerkiksi internetselaimet. (Mozilla Developer Network 2012). Opinnäytetyössä JavaScript liittyi selainohjelmointiin. Mainittakoon, että JavaScriptiä ei käytetä pelkästään asiakaspuolen teknologiana. JavaScript-ohjelmointikieltä käytetään myös palvelinohjelmoinnissa. Palvelinohjelmoinnin sovellutuksia on esimerkiksi Node.js, joka käyttää omaa JavaScript moottoria. JavaScript mielletään kuitenkin yleisesti ottaen asiakaspuolen selainohjelmoinnin *de factoksi* ja on yksittäisenä ohjelmointikielenä myös kaikkein eniten käytetyin.

JavaScriptin käyttäminen selainohjelmoinnissa on perusteltua siksi, että se vähentää palvelinriippuvaisuutta. JavaScriptin avulla voidaan esimerkiksi tarkistaa käyttäjän syötteitä selaimessa suoraan ilman, että tietoa tarkistettaisiin aina palvelimelta. JavaScriptillä päästään siis suoraan käsiksi selaimen käyttäytymiseen, jo siinä vaiheessa ennenkuin mitään oikeastaan näkyy loppukäyttäjä ruudulla. JavaScriptiä ei pidetä täysipainoisena ohjelmointikielenä (vrt. Java) vaan skripti-kielenä. Tämä johtuu siitä, että JavaScriptistä puuttuu useita tyypilliseen täysiveriseen olio-ohjelmointikieleen liittyviä ominaisuuksia. JavaScriptin puutteita ovat sen tiedostoon kirjoittamismahdollisuudet. Tämä on tietoturvallisuuteen liittyvä ominaisuus. Toisekseen sen ominaisuuksiin ei kuulu verkkotekniikoiden hyödyntäminen eikä prosessoreja hyödyntävää monisäietukea. (Tutorial’s point 2012a.).

Opinnäytetyöni prototyypissä JavaScriptiä käytetään merkittävissä määrin. Lisäksi prototyypissä käytettiin lisäksi kahta JavaScript kirjastoa: jQueryä, sekä WebGL Three.js-kirjastoa.

jQuery

jQuery on JavaScript kirjasto, josta on tullut Web-kehittäjien keskuudessa erittäin suosittu. JQuery on tehty helpottamaan mm. HTML:n DOM-käsittelyä, tapahtumien hallintaa ja HTML elementtien animointia. Lisäksi jQuery tukee AJAX:ia omine kehitystyötä tehostavilla funktioilla. jQuery merkittävimpiä ominaisuuksia on myös monipuolinen selaintuki. Web-sovelluskehityksessä eri selainten väliset erot mm. DOM-käsittelyssä voivat olla tuskastuttavan epäloogisia kehittäjän kannalta, joita jQuery omalta osaltaan pyrkii ratkaisemaan (Tutorial's point 2012b). Jquery tarjoaa kehittäjille paljon valmiita käyttöliittymäelementtejä. Näitä jQuery UI-elementtejä ovat mm. erilaiset napit, kalenteri-elementit, *sliderit* ja valmiit dialogit. Lisäksi jQuery tekee elementeistä käytettäviä sekä visuaalisesti näyttäviä UI-elementtien ollessa suoraan loppukäyttäjän liikuteltavissa ja käsiteltävissä.

JQuerya ja sen API:n mukana tulevia UI-elementtejä käytetaan toteutetussa prototyypissä. Päätös käyttää jQuerya opinnäytetyössä oli yksinkertaisesti sen ketteryys. Tällä saatiin nopeasti aikaiseksi näyttäviä ja käytännöllisiä käyttöliittymäelementtejä.

WebGL Three.js

Luvussa 3.1.2 WebGL esiteltiin tekniikka, joka mahdollistaa HTML5:en Canvas-elementissä 3D-grafiikan näyttämisen. Three.js JavaScript-kirjaston valinta prototyypin toteuttamiseen ei olisi ollut ainoa vaihtoehto. Toinen potentiaalinen tekniikka olisi ollut Adobe Flashin Stage3D. Työssä ei käytetty resursseja sen tutkimiseen siitä kumpi olisi ollut parempi valinta prototyypin toteuttamiseen. Mainitsemisen arvoista on, että Web-kehittäjäyhteisölle Adoben päätöksellä lopettaa Flashin kehitys mobiileille päätelaitteille oli suuri negatiivinen vaikutus. HTML5 ei ole valmis ja Adoben ekosysteemi AIR:in ja Flex:in johdolla täyttää kehittäjien tarpeet vielä pitkään. Tästä huolimatta, moni kehittäjä voi päätyä välttämättään pysymistä Adoben leirissä tai ylipäätänsä valita aloittaa kehitystyö Adoben tuotteilla. Painopiste tekniikan valinnassa kehittäjille yleisesti on HTML5 myönteinen, johon WebGL:n suosion kehittyminen luonnollisesti liittyy. Flash-teknologian maine on kokenut myös kolauksen Apple-laitevalmistajan päätöksestä olla tukematta Flashia. Lisäksi Flash-sovellusten hakukonenäkyvyys on ongelmallinen. Edellämainitut asiat vaikuttivat myös opinnäytetyön tekijän mielenkiinnon keskittymisestä WebGL-teknologian pariin.

Prototyypissä WebGL:ää laajentavan Three.js:n avulla toteutettiin 3D-kiihdytetty kehyksessä toimiva 3D-grafiikka.

4.5.2 Palvelinpuolen teknologiat

Prototyypissä toteutettiin yksinkertaisella tasolla käyttäjän syöttämän tiedon tallentaminen tietokantaan. Tietokannasta tieto vietiin 3D-kehykseen toteuttakseen tiedon visualisionti.

PHP

PHP on suosittu skriptikieli, jota voidaan upottaa HTML-koodin joukkoon. PHP eroaa JavaScriptistä siten, että ohjelmakoodi suoritetaan ainoastaan palvelimella. PHP koodi luo palvelimella HTML-koodia ja palauttaa sen selaimeen.

PHP:n avulla voidaan toteuttaa täysipainoisia selainpohjaisia Web-sovelluksia, jossa käsitellään käyttäjän syötteitä näytöllä ja suoritetaan monipuolisia tapahtumia palvelimella. (What is PHP 2012). Mainittakoot, että esimerkiksi sosiaalinen verkkopalvelu Facebook on toteutettu voimakkaasti PHP:hen luottaen.

Yleinen käyttökohde PHP:llä on sen sisäänrakennetut MySQL -tietokantafunktiot. Prototyypissä PHP:tä käytettiin MySQL-tietokantaa hyödyntäen GPS-paikkatietojen ja kuvatietojen tallentamiseen. PHP suoritti myös tallentamisen lisäksi myös tietokantahaut, jotta tieto voitiin välittää 3D-kehykseen.

MySQL

Kuten yllä jo todettiin, MySQL on eräs tietokannan hallintajärjestelmä. Vapaan lähdekoodin relaatiotietokannoista MySQL on yksi eniten käytetyimmistä. Tiedon tallentamisen lisäksi MySQL mahdollistaa käyttäjä -ja käyttäjäoikeushallinnan.

Prototyypissä toteutettiin yksitauluinen MySQL tietokanta, joka piti sisällään saraketietoina tiedoston nimen, tiedoston tyypin, tiedoston koon sekä GPS-paikkatiedon.

4.5.3 Autodesk 3ds Max ja JSON-konversio

Autodesk-yrityksen 3ds Max on digitaalisen grafiikkaan ja sisällön tuottamiseen kehitetty monipuolinen ohjelmisto. 3ds Maxin avulla voidaan mallintaa, animoida ja renderöidä 3D-grafiikkaa.

Prototyypissä 3ds Maxia käytettiin demonstraatiomielessä hyvin suppeasti. WebGL:n ja Three.js -JavaScript kirjaston avulla voidaan toteuttaa ohjelmointikooditasolla hyvin monipuolisia malleja eli objekteja. Erikseen 3D-mallintamiseen keskittyvällä ohjelmistolla voidaan 3D-malleja tuottaa intuitiivisesti kuitenkin merkittävästi tehokkaammin. 3ds Max ohjelmaan on saatavilla erillinen ilmainen Three.js plugin, liitännäinen, joka mahdollistaa suoraan 3D-malliin kuuluvan informaation konvertoinnin JSON-formaattiin. JSON on yksi

tiedonsiirtomuoto ja on helppokäyttöinen yhdessä JavaScriptin kanssa. Kuten on esitelty, WebGL-grafiikkaa tuotetaan JavaScriptillä yhdessä Three.js -kirjaston avulla. Tällä kirjastolla JSON-tyyppinen tieto voidaan tuoda näytölle suoraan omana kolmiulotteisena objektinaan. Prototyypissä käytännössä todettiin tekniikan toimivuus ja JSON:in avulla tuotettiin prototyypissä nähtävä, käyttäjän valintaa havainnollistava sininen kursori.

3ds Max ei ole ainoa 3D-kehitysympäristö, jolle edellämainittu JSON-konversio objektille voidaan tehdä. Erillinen liitännäinen on toteutettu myös ilmaiselle Blender 3D-työkalulle.

4.6 Prototyypin käyttöliittymä

Prototyypin käyttöliittymästä toteutettiin hyvin minimalistinen. Käyttöliittymän tuli koostua kahdesta eri kehyksestä. Perinteisen HTML-sivun sisälle toteutettiin 3D-grafiikkaa sisältävä kehys (ikkuna). Kuviossa 6, Picquer-prototyypin käyttöliittymä on 3D-kehys merkitty punaisella värillä. Tämän ulkopuolelle jäävä osa on tyypillistä HTML:ää.

KUVIO 6. Piquer-prototyypin käyttöliittymä.

Prototyypissä käyttäjä voi valita haluamansa tiedoston ja suorittaa Picquer-it!-toiminnon. Tämä toiminto siirtää tiedoston tietokantaan, joka lopulta tuotetaan 3D-kehykseen omaksi objektikseen. Käyttäjän valinnat 3D-kehyksessä tuovat esille käyttöliittymäelementtejä, jotka ovat toteutettu jQueryllä.

4.7 Prototyypin ohjelmakoodi

Tässä luvussa esitetään kooditasolla tärkeimmät prototyypissä toteutetut osa-alueet. Prototyypin tärkein ominaisuus oli mahdollisuus tuottaa alunperin GPS-sateellitista

kuvatiedostoon tulleen informaation esittäminen WebGL-tekniikalla toteutetulla 3D-karttapallolla. Lisäksi lukijalle selitetään tärkeimpiä koodiin kuuluvia funktioita.

Prototyyppiin kirjoitettiin noin 950 riviä ohjelmakoodia. Rivimäärä suhteutettuna toteutettuihin ominaisuuksiin pienenisi todennäköisesti tulevaisuudessa, sillä prototyypissä ei lähdetty siistimään koodia tehokkaammaksi, eikä siinä hyödynnetä koodin uudelleenkäyttöä.

4.7.1 GPS-tiedon poiminta PHP:llä

Käyttöliittymässä GPS-tiedon poiminta tapahtuu HTML-kehyksessä. Käyttäjä valitsee HTML-lomakkeella haluamansa kuvatiedoston siirtääkseen sen palveluun. Huomiotavaa on PHP:n exif_read_data -funktio. Tämän funktion avulla tutkitaan käyttäjän syöttämästä tiedostosta EXIF-headeria, joka on tiedostossa sijaitsevaa eräänlaista meta-tietoa. Tämän funktion käyttäminen edellyttää, että palvelimen PHP.ini-asetuksista on otettu käyttöön php_exif.dll ja php_mbstring.dll kirjastot. Kuviossa 7 on esitetty pätkä ohjelmakoodia, jossa GPS-tieto poimitaan tiedostosta.

```
<?php
            if(isset($_POST['upload']) && $_FILES['userfile']['size']
            > 0)

            {

            // Poimitaan koordinaatit tiedostosta EXIF headerista
            $coordinatesTmp =
            @exif_read_data($_FILES['userfile']['tmp_name']);

            $coordinatesLatitude =
            haeGps($coordinatesTmp['GPSLatitude']);

            $coordinatesLongitude =
            haeGps($coordinatesTmp['GPSLongitude']);

            $coordinates = $coordinatesLatitude . ',' .
            $coordinatesLongitude;


            // Palautetaan käyttäjälle tieto siitä, että sisältääkö
            // EXIF header informaatiota.

            echo $coordinatesTmp===false ? "  -No header data found.
            Choose a img with Geodata.<br />\n" : "  +Image contains
            headers<br />\n";

            // Poimitaan käyttäjän siirtämästä tiedostosta muut
            // tiedot, jotta ne voidaan tallentaa tietokantaan.

            $fileName = $_FILES['userfile']['name'];
            $tmpName  = $_FILES['userfile']['tmp_name'];
            $fileSize = $_FILES['userfile']['size'];
            $fileType = $ FILES['userfile']['type'];
```

```
            move_uploaded_file($_FILES["userfile"]["tmp_name"],
            "upload/" . $_FILES["userfile"]["name"]);
            echo "Stored in: " . "upload/" .
            $_FILES["userfile"]["name"];

            // Siistitään muuttujaa
            if(!get_magic_quotes_gpc())
            {
              $fileName = addslashes($fileName);
            }

            // Lisätään arvo tietokannan tauluun

            $query = "INSERT INTO files (name, size, type,
            coordinates ) " . "VALUES ('$fileName',
            '$fileSize', '$fileType', '$coordinates')";

            mysql_query($query) or die('Error, query failed');

            // Palaute käyttäjälle, että tiedosto on onnistuneesti
            // siirretty tietokantaan

            echo "<br>File $fileName uploaded
            <b>successfully</b><br>";

            }

?>
```

KUVIO 7. Ohjelmakoodi GPS-tiedon poimimiselle kuvatiedostosta

Ylläolevassa koodipätkässä käytetään funktiota haeGPS() leveys -ja pituuspiirien merkkijonojen siivoamiseksi. EXIF Headerissa oleva paikkatieto oli sen suoran jatkokäyttämisen kannalta ongelmallinen. Tämä funktio palauttaa leveys -tai pituuspiirin koordinaatin desimaalilukuna.

4.7.2 GPS-tiedon esittäminen 3D-kehyksessä

3D-kehys on HTML-tasolla toteutettu iframe-elementin sisään erilliseksi ympäristöksi. Prototyyppi kutsuu tässä JavaScript-koodissa kolmea funktiota, jotka kuviossa 8 lisäksi ohjelmakoodina esitetään. Ensimmäisessä, *init()*-funktiossa alustetaan 3D-ympäristö. 3D-kehyksessä prototyypin haluttiin pystyvän näyttämään karttapallolla koordinaatin sijainti ja tieto kuvasta, joka palveluun siirettiin. Tämä toteutettiin lähettämällä muuttujat erilliselle funktiolle, *addBar(),* joka poimii edellämainitut tiedot ja luo palkin maapallolle. Kolmanneksi kutsutaan *animate()*-funktiota, joka käynnistää WebGL-näytön päivityksen.

```
<script type="text/javascript">
            init();
            addBar(tiedot_array);
            animate();
</script>
```

KUVIO 8. 3D-kehyksen tiedon näyttämisen käynnistävät funktiot

Init-funktio

Init-funktio hyödyntää Three.js-kirjaston funktioita. Tässä funktiossa luodaan nk. scene, eräänlainen kulissi tai näyttämö, johon kaikki 3D-grafiikka luodaan. Lisäksi määritellään joukko tapahtumakuuntelijoita (eng. event listeners) tutkimaan käyttäjän valintoja ja toimintoja näytöllä. Kuunneltavia tapahtumia ovat esimerkiksi hiiren vasemman ja keskinapin painallukset tai hiiren rullan aktiviteettiin reagointi. Kuviossa 9 luodaan WebGL-karttapallo Three.js funktioilla, sekä valot näyttämölle.

```
// Luodaan näyttämö
scene = new THREE.Scene();

// Karttapallon lisäys:
// Luodaan aluksi geometria
geometry = new THREE.SphereGeometry( 200, 40, 30 );

// Luodaan materiaali
material = new THREE.ShaderMaterial({
            uniforms: uniforms,
            vertexShader: shader.vertexShader,
            fragmentShader: shader.fragmentShader
           });

 // Luodaan ”meshi”
maapallo = new THREE.Mesh( geometry, material );

// Lisätään maapallo näyttämölle
scene.add( maapallo );
```

KUVIO 9. Init-funktiossa toteutettu 3D-karttapallon luonti ohjelmakoodina

Huomioitavaa on, että koodipätkässä ei esitellä tarkemmin materiaalin luontia. Käytännössä materiaalimuuttuja koostuu verteksi -ja fragmentti varjostimille annetuista lisäparametreistä. Nämä parametrit on kirjoitettu GPU:lle natiivilla GLSL-kielellä ja on opinnäytetyöhön otettu Google Data Arts Team:in The WebGL Globe -toteutuksesta.

AddBar-funktio

Addbar-funktion rooli prototyypissä on viedä funktiolle tietokannasta poimittu parametri. Parametri on taulukko, joka sisältää erikseen siivottuja merkkijonoja. Lopputuloksena syntyy n-kappaletta suorakulmaisia särmiöitä, eli eräänlaisia palkkeja. Ohjelmakoodissa palkki on käännetty englanninkielen sanaksi bar. Nämä palkit renderöidään 3D-karttapallolle. Kuviossa 10 funktio käsittelee sille annetun taulukon indeksit, erittelee tiedoston nimen sekä koordinaatit ja nimeää syntyneet objektit. Funktiossa koordinaateista lasketaan matemaattisesti niiden sijainti pallolla. Lisäksi särmiölle, jokaiselle tahkolle annetaan materiaalit.

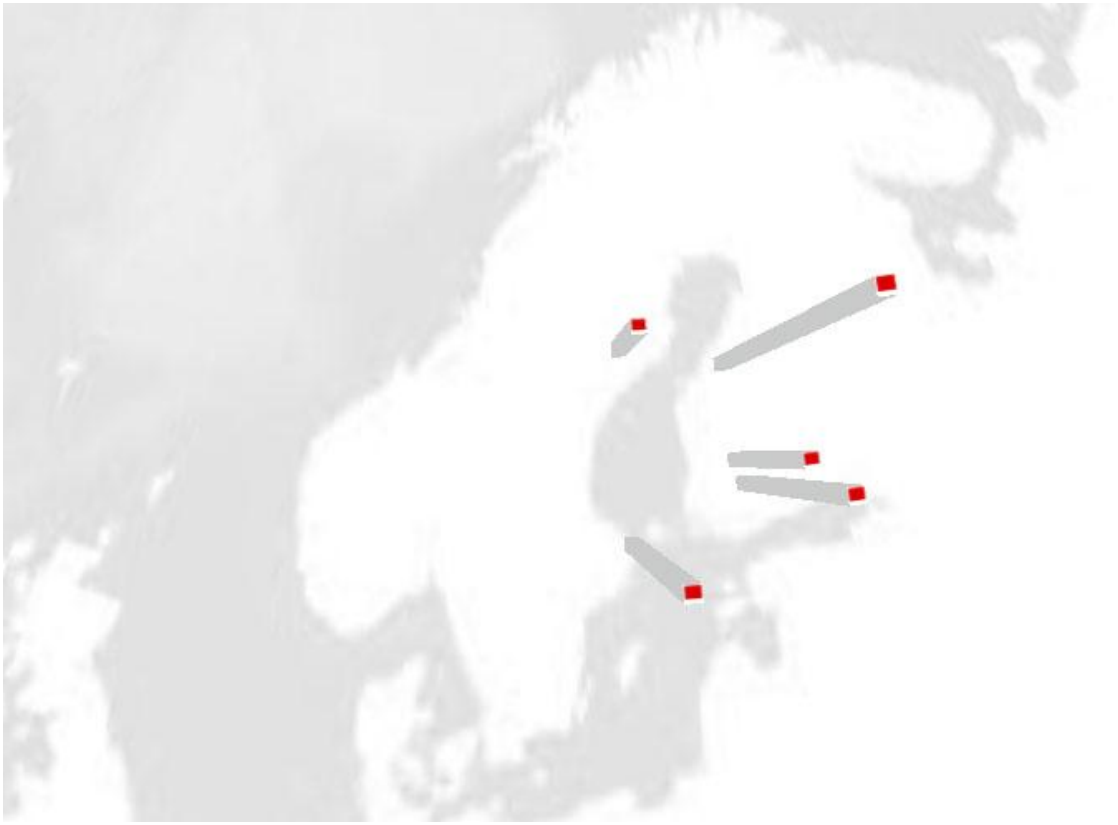
```
function addBar(coordinates) {
```

```
// Alustetaan apumuuttujat
var latitude;
var longitude;
var fileName;
var r = 200; // Pallon säde

// apumuuttuja "j", jolla pakotetaan objekteille
// yksilöivä ID-tieto, sekä suffiksi, esim. palkki1,
// palkki2, palkkiN.
var j = 0;


// Määritellään taulukot, joihin poimitaan for-silmukasta
// erikseen erikseen koordinaatit ja objektit myöhempää
// käyttöä varten.
collectedCoordinateArray = []; //
collectedObjektiArray = [];


// Määritetään materiaaleille taulukko, johon asetetaan
// tekstuurit. Silmukassa jokaiselle indeksillesijoitetaan
// materiaaliobjekti. Materiaalitaulukko annetaan
// parametrina myöhemmin luotavalle meshille.
var materials = [];

for ( var i = 0; i < 6; i ++ ) {
  materials.push( new THREE.MeshBasicMaterial( { color:
  0xffffff, map: THREE.ImageUtils.loadTexture( 'textures/'
  + i + '.gif'), overdraw: true } ) );
}

// Pääsilmukka alkaa, jossa tuotetaan uudet palikat
for (i = 0; i < coordinates.length; i += 3) {

// Prototyyppivaiheessa palkin korkeus arvottiin
var randomizedBarHeight = Math.floor(
            (Math.random()*80)+10
            );


// Pilkotaan iteraatiokierroksella funktiolle syötetty
// muuttuja ja sijoitetaan apumuuttujiin.
fileName = coordinates[i];
latitude = coordinates[i+1];
longitude = coordinates[i + 2];
j++;

// Luodaan uusi objekti (tuleva uusi "särmiö")
objekti = new THREE.Mesh( new THREE.CubeGeometry( 2, 2,
randomizedBarHeight, 1,1,1, materials ), new
THREE.MeshFaceMaterial() );

collectedCoordinateArray.push(
           objekti.id,latitude,longitude
           );
collectedObjektiArray.push(objekti);

// Lasketaan atsimuutti ja kallistus
var phi = (90 - latitude) * Math.PI / 180;
var theta = (180 - longitude) * Math.PI / 180;

// Sijoitetaan luotu objekti pallolla vastaamaan
// koordinaatteja
```

```
            objekti.position.x = r * Math.sin(phi) * Math.cos(theta)
            * -1;
            objekti.position.y = r * Math.cos(phi);
            objekti.position.z = r * Math.sin(phi) * Math.sin(theta)
            * -1;

            objekti.name = "bar" + [j];
            objekti.id = j;

            // Käännetään särmiö osoittamaan pallolla oikeaan
            suuntaan.
            objekti.lookAt(mesh.position);



            // Täydennetään objektin attribuutteja myöhempää käyttöä
            // varten
            objekti.originalLatitude = latitude;
            objekti.originalLongitude = longitude;
            objekti.originalCoordinatesArray = [latitude,longitude];
            objekti.fileName = fileName;

            // Lisätään luotu objekti näyttämölle
            scene.add( objekti );

        } // For-silmukka loppuu
}
```

KUVIO 10. Addbar-funktiossa toteutettu palkin lisäys ohjelmakoodina

Kuviossa 11 on nähtävillä 3D-karttapallolle yllä olevassa koodissa luotuja palkkeja.



KUVIO 11. 3D-karttapallolle ohjelmoituja palkkeja.

Prototyyppiin rakennettiin ominaisuudet, jotka pystyivät sijoittamaan karttapallolle käyttäjän syöttämän kuvatiedoston GPS-koordinaatit.

Animate-funktio

Animate-funktio kutsuu *requestAnimationFrame()* -JavaScript funktiota. Sen tehtävä on pyytää selainta päivittämään näkymää enimmillään 60 kertaa sekunnissa (fps), jolloin käyttäjä näkee sujuvan 3D-sovelluksen animointineen.

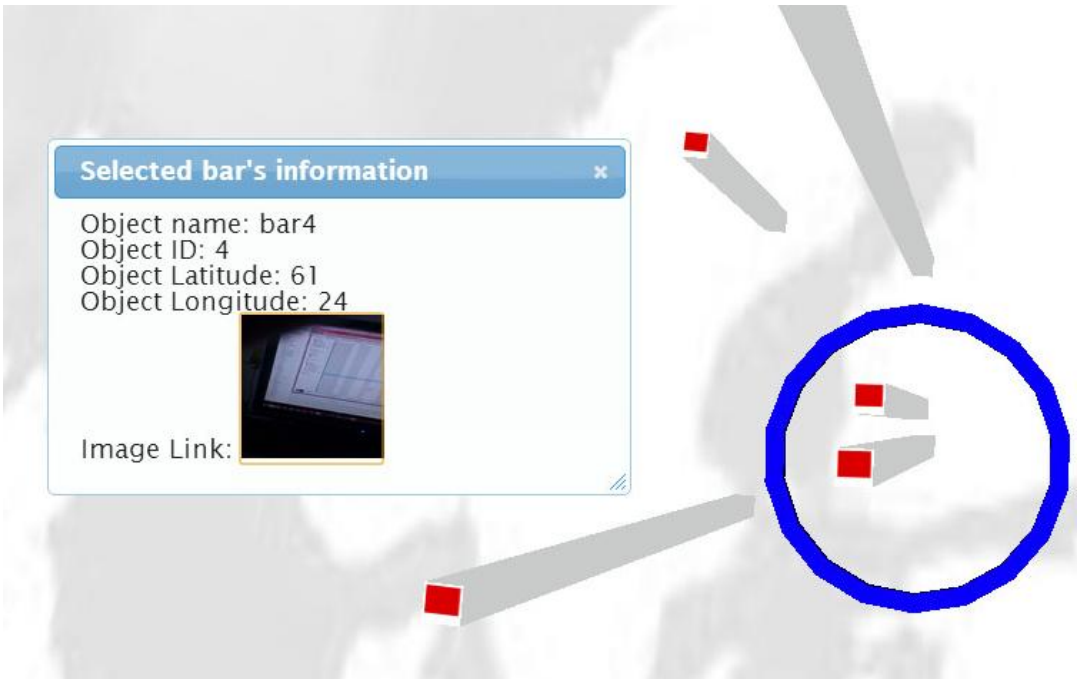
Animate()-funktio kutsuu *render()*-funktiota, joka käytännössä laskee ja piirtää kaiken 3D-grafiikan selaimessa, jokaiselle päivitettävälle kuvalle. Render()-funktiossa lisäksi päivitetään kameran sijainti näyttämöllä. Kamera toimii eräänlaisena silmänä, jolla tässä tapauksessa karttapalloa katsotaan. Kameraa voidaan kontrolloida hyvin monipuolisesti ohjelmakoodista käsin. Esimerkiksi hiiren rullan käyttäminen käynnistää tapahtuman kuuntelijan avulla toiminnon, joka muuttaa kameran sijaintia näytöllä (ts. zoomi). Jotta kameran sijainnin liikkuminen näytöllä olisi sujuvaa, joudutaan näyttämö renderöimään uudestaan. Tämän vuoksi on järkevää päivittää tieto kameran sijainnista kyseiseen funktioon.

Animate()-funktiota kutsutaan niin pitkään kuin sovellus on käynnissä. Tämä on loogista, sillä haluamme sovelluksen pysyvän animoituna ja renderöitynä koko käytön ajan.

4.7.3 Objektin havaitseminen karttapallolla

Sen jälkeen kun prototyyppiin oli toteutettu mahdollisuus sijoittaa haluttuun paikkaan paikkatietoa kuvaava palkki, oli pystyttävä tutkimaan käyttäjän tekemiä valintoja näytöllä. Tutkiminen on kuvaava sana siinä mielessä, että näytölle renderöidyt objektit eivät ole tässä vaiheessa esimerkiksi HTML:ää, johon voitaisiin päästä käsiksi koodissa HTML:n DOM-rakennetta hyödyntäen. Karttapallo ja siinä olevat objektit ovat jatkokäsittelyn kannalta varsin näkymättömiä dynaamisessa 3D-karttapallossa, sillä niiden rooli on oikeastaan vain näyttää sitä tietoa mitä sille on syötetty. Haaste on siinä, kuinka saada ympäristön objektit reagoimaan esimerkiksi käyttäjän hiiren painalluksiin ja kertoa sovellukselle, että se on valittu. Lisäksi prototyyppiin haluttiin toteuttaa tapa tutkia käyttäjän valinnan vaikutusta siten, että pääsisimme käsiksi useampaan kuin yhteen objektiin kerralla. Huomioitavaa on se, että objektilla on ainoastaan paikkatieto, leveys -ja pituuspiirikoordinaatit, sekä ID-tieto. Ratkaisu ei ole se, että tallennamme tietokantaan jatkuvasti dataa siitä, missä karttapallolla on tai ei ole objekteja. Prototyypissä ongelma ratkaistiin tutkimalla käyttäjän valintoja. Kun käyttäjä painoi hiiren nappia, tutkittiin suoraviivaisesti kohde, mihin käyttäjän valinta osui. Kuten mainittiin, karttapallolla olevat objektit eivät kuuntele valintaansa, vaan tutkiminen tapahtui erillisellä *säde*-objektilla. Säde (eng. ray) voidaan kuvitella esimerkiksi lasersäteenä,

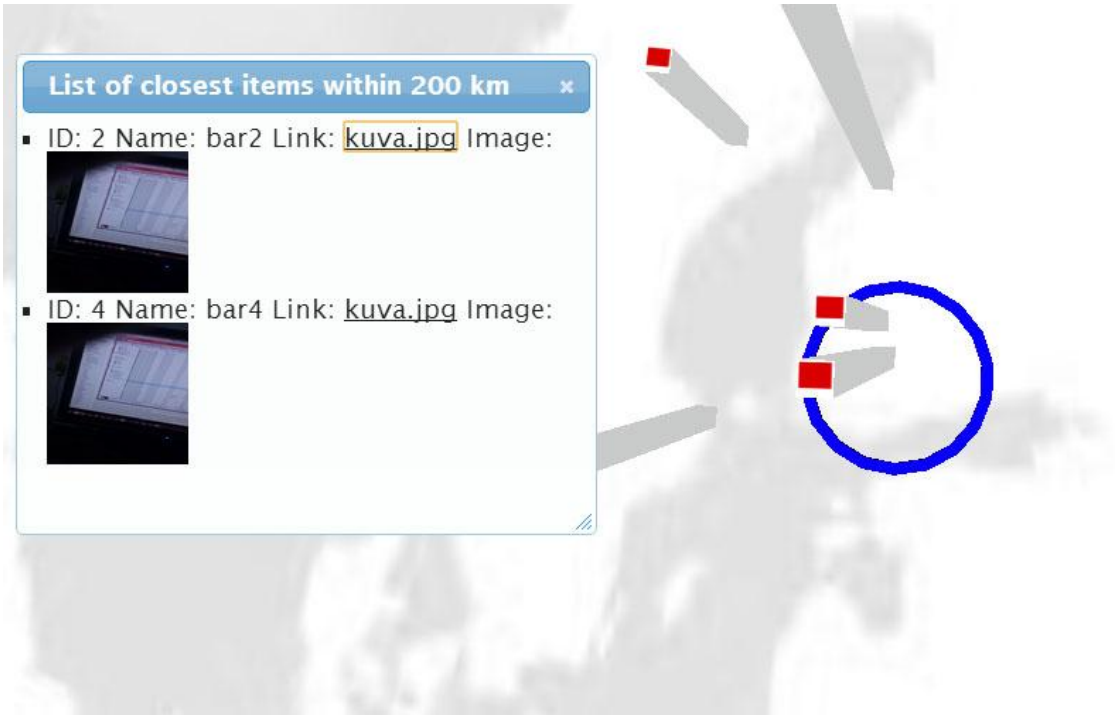
joka lähtee hiiren kursorin päästä lävistäen näyttämön. Säde on oikeastaan vektori (eng. vector), joka tekee suoran säteen, viivan, näyttämön kamerasta kohti käyttäjän hiiren kursoria. Tämä ratkaisu mahdollistaa kameran ja objektien vapaan asettelun näyttämölle. Ratkaisevaa objektin havaitsemisen kannalta on mihin objektiin säde ensimmäisenä osuu. Kun säde osuu objektiin, päästään ohjelmakoodissa käsiksi sen attribuutteihin. Kuviossa 12 on käyttäjälle palautettu objektin tiedot. Prototyyppivaiheessa kiinnostavaa oli objektin tietojen löytämisen lisäksi yhteys alkuperäiseen palvelimelle siirrettyyn digikuvaan.



KUVIO 12. Valinnan ja objektin havaitsemisen myötä annettiin käyttäjälle palaute

Toinen kiinnostava ominaisuus, joka prototyyppiin toteutettiin oli mahdollisuus tutkia samanaikaisesti useampaa objektia halutulla alueella. Edellisessä kappaleessa esitettiin tekniikka, jolla yksittäisiä kohteita karttapallolla voitiin tutkia. Tekniikka oli paljolti sama myös alueella sijaitsevien useamman objektin tutkimisessa.

Toteutuksessa hyödynnettiin matemaattista haversini-kaavaa, joka soveltuu tähän käyttötarkoitukseen hyvin. Haversinin funktiossa tutkitaan pallomaisella kappaleella kahden koordinaattiparin välisiä etäisyyksiä. Prototyyppissä käytössä oli maapallon säde, käyttäjän kursorin osoittama koordinaattitieto, sekä koordinaattitiedot maapallolla olevista objekteista. Käytännössä ohjelma toteutti silmukan, jolla tutkittiin yksitellen kaikki ennaltamääritetyn säteen sisälle osuvat objektit. Ennaltamääritelty säde prototyypissä oli 200km. Kaikki säteen sisälle osuvat objektit sijoitettiin taulukkoon, joka voitiin esittää listana käyttäjälle. Kuviossa 13 on ruutukaappaus prototyypistä, jossa käyttäjän valintaan on kohdistunut kaksi eri objektia, eli palkkia.



KUVIO 13. Esitetään käyttäjälle kaikki 200km säteen sisälle osuvat objektit, eli kohteet

5 JATKOKEHITYS

Picquer web-sovelluksen jatkokehitys on opinnäytetyön kirjoitushetkellä epävarmaa. Jatkokehitystä voi miettiä useammalta kannalta. Lopullista, kaupallista, tuotetta silmälläpitäen ohjelmakoodin jatkokehitys tulisi vaatimaan koodin uudelleenkirjoittamisen monestakin syystä. Ensinnäkin prototyyppiin toteutettu koodi ei täytä ohjelmistosuunnittelun hyväksi todettuja oppeja. Luokka-ajattelua ei käytetä ja työn laajentaminen prototyypissä nähdyn ohjelmakoodin päälle tekisi kehityksestä entistä vaikeampaa. Prototyypissä käytetty Three.js-JavaScript kirjasto kehittyy kovaa vauhtia. Opinnäytetyön kirjoitusajan kuluessa kyseisestä kirjastosta on julkaistu jopa 13 uutta releasea, versiopäivitystä. Uuden version käyttöönotto todettiin prototyypissäkin haastavaksi ja se ei onnistuisi ilman isoja muutoksia. Kun mukaan otetaan tietokannan jatkokehitys ja onnistuneen käyttöliittymän luonti suorituskyvyt huomioiden, on toimeksiantajan syytä harkita alan ammattilaisen apua.

Mikä sitten on Picquer-sovelluksen jatkokehityksen päämäärä? Tähän pohdintaan voidaan liittää englanninkielinen käsite *minimum viable product*. Käsite on peräisin Eric Riesin teoksesta Lean Startup vuodelta 2011. (Ries, R. 2011). Riesin mukaan *minimum viable product* on tietynlainen rakennettavan tuotteen havainnollistava tapa tai tuotos. Tapauskohtaisena tämä voidaan tehdä ennen prototyyppiä tai edes, että riviäkään koodia on tuotettu. Malliesimerkkinä tästä on Dropbox-palvelu. Dropbox on suosittu tiedostojen jako pilvipalvelu. Dropbox sovellus noudatti Lean Startup-filosofiaa ja tuote käytännössä markkinoitiin aluksi pelkällä muutaman minuutin esittelyvideolla. Dropbox-sovelluksen kehityksestä ei tässä vaiheessa ollut mitään takeita onnistumisesta.

Minimum viable product on myös sovelluksen kaupalliseen toteutukseen liitettävä strategia, jossa tuotteesta toteutetaan *jotain* käyttäjien ja sijoittaijien mielenkiinnon herättämiseksi. Tämä osa voi olla esimerkiksi yksinkertainen sovelluksen nk. *landing page*-internetsivu, johon käyttäjä ohjataan ensimmäisenä. Sivulla voidaan esimerkiksi näyttää sovelluksen kiinnostavia ominaisuuksia eri medioiden keinoin, kuten Dropboxin tapauksessa. Markkinoinnin kannalta on hyvin tärkeää tietää, minkälaista reaktiota tai *hypeä* sivusto ja tätä myötä myös sovellus saa aikaiseksi. On suotavaa, että Picquer-sovelluksen jatkokehityksessä *minimum viable product*-filosofia pidetään mielessä. Opinnäytetyön yhteydessä rakennettu prototyyppi voisi olla Picquerin *minimum viable product*.

Yllämainittua *minimum viable product*-termiä pohditaan sovelluskehityksessä kiteytetysti Emre Sokullun artikkelissa "How to create a Minimum Viable Product" Techcrunch-teknologiasivustolla. Artikkeli nostaa esille hyviä ideoita ja ominaisuuksia Picquer-

sovelluksenkin jatkokehitystä ajatellen. Hän listaa artikkelissaan mm. työkaluja ja vinkkejä, jolla sovelluskehittäjä ja hänen toimeksiantajansa voivat tuotteensa viedä tasolle, jolla herättää käyttäjäkunnan mielenkiinnon.

1. Toteutetaan sovellus Facebook-alustalle. Käyttäjähallinta on iso kokonaisuus, joka on Facebookissa jo valmiiksi.

2. Käytetään valmiita JavaScript-kirjastoja, joiden tehtävä on auttaa sovelluksen kehitystyössä siten, että sovellus toimii käyttöliittymätasolla mobiililaitteilla sekä työpöytäkoneissa.

3. Sijoita sovellus pilveen. Tehokkaiden pilvipalveluiden hinnat ovat tänä päivänä niin alhaiset, että omien palvelinten ylläpito ei ole mielekästä.

4. Käytä jQueryä. JQuery on toteutettu toimimaan lukuisille eri selaimille ja nopeuttaa entisestään sovelluksen kehitystyötä.

5. Muista keskittyä ydintoiminnallisuuteen. Ominaisuuksia ei pidä yrittää toteuttaa liian montaa.

6. Käytä SaaS (Software as a service) palveluita. Internetissä on löydettävissä tahoja, jotka hoitavat kokonaisvaltaisesti esimerkiksi sovelluksen kuukausilaskutuksen, asiakaspalvelun tai sivuston kävijäanalytiikan.

7. Sijoita sovelluksen viralliset paperidokumentit suoraan erilliseen palveluun esimerkiksi lakimiehesi saatavaksi.

8. Luo sovelluksesta video. Video esittää käyttäjälle palvelun useasti paremmin kuin raskas JavaScript demo.

9. Unohda Internet Explorer. Mikäli rahasta ja ajasta on puutetta, jätä Internet Explorer tuki kokonaan pois. Interet Explorer käyttäjät profiloituvat aloitteleviin käyttäjäryhmiin, jolloin he todennäköisesti eivät ole myöskään potentiaalisia uuden palvelun käyttäjiä.

10. Kaikkien ominaisuuksien ei kannata olla valmiita heti alusta alkaen. Esimerkiksi palvelun rekisteröitymiseen tai käyttömaksuihin liittyvät ominaisuudet voidaan toteuttaa alussa pelkän sähköpostin avulla. Kasvata toteutettavien ominaisuuksien listaa ajan myötä.

(Sokullu 2012)

Edellämainituista kohdista voitaisiin luoda pohja seuraavalle strategiselle liikkeelle kohti toimivaa sovellusta. Toimeksiantajan harkinnan alla on pohdinta myös ylipäätään siitä, kuinka pitkälle ideaa eriasteisine toteutuksineen halutaan viedä. On mahdollista, että koko sovelluksen idea voidaan kaupallistetaan ja myydään konseptina toiselle taholle.

6 TULOKSET JA POHDINTA

Tuloksena kaupallista potentiaalia kantavalle Picquer web-sovellukselle syntyi prototyyppi, jonka tarkoitus oli tutkia ja ratkaista haasteita, joihin kehitystyössä tultaisiin aikaisessa vaiheessa törmäämään. Prototyyppi keskittyi HTML5:en mahdollistavan WebGL:ään perustuvan 3D-grafiikan esittämiseen internet-selaimessa. Näyttävän, tietokoneen grafiikkapiiriä hyödyntävän 3D-grafiikan tuominen selaimiin tulee epäilemättä näyttelemään tulevien vuosien aikana valtavaa roolia internetin kehityksessä. WebGL:n selaintuki kasvaa hyvää vauhtia, mutta matka on kuitenkin vielä pitkä siihen, että saavutettaisiin Adobe Flash:iin verrastettavissa oleva yli 95% selaintuki (StatOwl 2012). 3D-grafiikan ollessa lähtökohtaisesti raskaampaa tietokoneen ja näytönohjaimen prosessorille vastaan tulee pian käyttäjän tietokoneen suorituskykyhaasteisiin. 3D-grafiikan toistaminen riittävän sulavasti tulisi edellyttämään myös käytettävien kolmiulotteisten tekniikoiden kehittimien osalta kehitystyötä. Tällä hetkellä webbikehittäjä voi olla enemmän tai vähemmän kiinnostunut loppukäyttäjän laitteiston suorituskapasiteetista. Toivottu ominaisuus voisi olla joko API:in tai vaikkapa selaimeen sisäänrakennettu näytettävän grafiikan skaalautuvuus. Skaalautuvuudella tarkoitetaan tässä sitä, että tekniikka lähtökohtaisesti ottaa huomioon loppukäyttäjän laitteiston suorituskyvyn ja mukautuu tämän mukaan. Työssä ei tutkittu Stage3D:n nykytilaa ja soveltuvuutta Picquer-kontekstiin. Adoben ekosysteemin kehitys on mielenkiintoinen aihe sinänsä, mutta kaikki merkit näyttäisivät web-kehittäjän näkökulmasta osoittavan HTML5:en ja JavaScript-toteutuksien suuntaan. Adobelta, jos keneltä, on syytä odottaa tulevaisuudessa houkuttelevia kehittimiä juuri HTML5:een ja tämän tarjoamien ominaisuuksien, kuten 3D-grafiikkaan Canvas-elementissä, suhteen.

Yhdessä toimeksiantajan kanssa saatiin lisäksi ensimmäinen käsitys siitä, minkälaisia ominaisuuksia sovellus voisi pitää sisällään. Kaikki opinnäytetyön toteutuksen aikana havaitut asiat antoivat osapuolille perspektiiviä siitä, miten suuresta projektista tulisi olemaan kysymys. Picquer-sovelluksen peliaspektin sekä toimivan kuvapalvelun kokonaisuuksien yhdistäminen käyttäjäkunnalle viihdyttävästi ja teknisesti kypsästi, on varmasti yksi kehitystyön suurimmista haasteista. Useat sovellukset lähtevät usein yksinkertaisesta ideasta. Picquerin idea yhdistää peli ja kuvapalvelu on hyvä ja omaa potentiaalia molempien kokonaisuuksien jatkokehitystä ajatellen. Kolmansien osapuolien, kuten digikuvien tulostamiseen keskittyvien yritysten, mukaanottaminen on hyvä esimerkki.

Opinnäytetyö pyrki alusta asti vastaamaan toimeksiantajan kysymykseen: onko tämä sovellus mahdollinen? Kaikki oleellisimmat tekniset ominaisuudet olivat ohjelmoitavissa ja

toteutettavissa. Jo pelkästään tämä havainto näyttää mielestäni vihreää valoa sille, että toimeksiantajan ei kannata jättää Picquer-konseptia pöydälle pölyttymään.

LÄHTEET

Anson Alex. 2012. Teknologiasivuston artikkeli 20.2.2012. Viitattu 19.11.2012. http://ansonalex.com/infographics/facebook-user-statistics-2012-infographic/

Answers n.d. Monialainen kysymys-ja vastaussivusto. What is the Difference between web application and web service? Viitattu 17.9.2012. http://answers.com/Q/What_is_the_Difference_between_web_application_and_web_service

Bringing 3D to the Web. 2002. Cnet -uutissivuston artikkeli 26.2.2002. Viitattu 17.9.2012. http://news.cnet.com/2100-1023-844985.html

Chrome Experiments, n.d. 3D-grafiikkaan selaimessa keskittynyt demo-sivusto. Viitattu 20.10.2012. http://www.chromeexperiments.com/

Clark, J. 2009. Why prototype? -artikkeli 7.9.2009. Viitattu 20.10.2012. http://softwareprototyping.net/why-prototype/

Conceivably Tech. 2011. Teknologiasivusto Conceivably Tech:in Chrome Is The Fastest WebGL Browser, Says Facebook-artikkeli 25.2.2011. Viitattu 13.11.2012. http://www.conceivablytech.com/5848/products/chrome-is-the-fastest-webgl-browser-says-facebook

Dev.Opera. 2011. An introduction to WebGL. Viitattu 17.9.2012. http://dev.opera.com/articles/view/an-introduction-to-webgl/

Firefox 4 gets yet another final test build release. 2011. The Channel teknologiasivuston artikkeli 21.3.2011.Viitattu 17.9.2012. http://www.channelregister.co.uk/2011/03/21/firefox_4_release_candidate_2/

Google Earth Blog. 2011. The Amazing things about Google Earth. Viitattu 20.10.2012. http://www.gearthblog.com/blog/archives/2011/10/google_maps_moves_a_bit_closer_to_g.html

Halme 2012a. Liiketoimintamalli. Sähköpostiviesti 17.10.2012. Vastaanottaja Martti Halme.

Halme 2012b. User stories. Sähköpostiviesti 18.10.2012. Vastaanottaja Martti Halme.

Halme 2012c. Toimeksiantajan haastattelu Helsingissä 24.8.2012.

Imaging resource. 2011. Digi -ja järjestelmäkameroihin keskittyvä sivusto. Uutisartikkeli 1.6.2011. Viitattu 19.11.2012. http://www.imaging-resource.com/NEWS/1306959537.html

Johnny Appleseed n.d. What is GPS? Viitattu 7.10.2012. http://www.ja-gps.com.au/what-is-gps.aspx#howitworks

Karlsen, L. 2010. Tietotekniikkaan keskittyvä keskustelu-ja apufoorumi Stack Overflow. What is the difference between web service and web application? 14.5.2010. Viitattu 17.9.2012. http://stackoverflow.com/questions/2832059/what-is-the-difference-between-webservice-and-webapplication

Khronos Group n.d. WebGL-OpenGL ES 2.0 for the Web. Viitattu 12.9.2012, http://www.khronos.org/webgl/

Khronos Group OpenGL ES. n.d. Khronos Group yhdistyksen esittely OpenGL ES-standardista. Viitattu 17.9.2012. http://www.khronos.org/opengles/

Khronos Group OpenGL n.d. Khronos Group yhdistyksen esittely aiheesta OpenGL. Viitattu 17.9.2012. http://www.khronos.org/opengl/

Korkama, T. n.d. Powerpoint-esitelmä aiheesta Varjostinohjelmointi. Viitattu 17.9.2012. http://www.cs.helsinki.fi/u/tkorkama/Seminaarin%20esittely.ppt

Movable Type Scripts n.d. Calculate distance, bearing and more between Latitude/Longitude points. Viitattu 7.10.2012. http://www.movable-type.co.uk/scripts/latlong.html

Mozilla Developer Network. 2012. JavaScript Overview. Viitattu 7.11.2012. https://developer.mozilla.org/en-US/docs/JavaScript/Guide/JavaScript_Overview

Open Street Maps. 2012. Open Street Maps palvelun oma wiki-sivusto. Viitattu 20.10.2012. http://wiki.openstreetmap.org/wiki/Main_Page

Photobucket. 2012. Photobucket-kuvapalveluun rekisteröitymissivu. Viitattu 11.12.2012. https://secure-beta.photobucket.com/register

Primac, D. 2012. CNNMoney uutissivusto artikkeli 9.4.2012. Viitattu 27.11.2012. http://finance.fortune.cnn.com/2012/04/09/breaking-facebook-buying-instagram-for-1-billion/

Ries, R. 2011. The Lean Startup. Viitattu 30.11.2012.

Sokullu. 2012. Emre Sokullun artikkeli 13.7.2012, How to create a minimum viable product Tech Crunch-teknologiasivustolla. Viitattu 15.11.2012. http://techcrunch.com/2012/07/13/how-to-create-a-minimum-viable-product/

StatOwl. 2012. Internetin tilastoihin keskittyvä sivusto. Viitattu 11.12.2012. http://www.statowl.com/flash.php

The WebGL Globe. 2012. Chrome Experiment -WebGL demo. Viitattu 20.10.2012. http://www.chromeexperiments.com/globe

The Windows Blog. 2011.Windows Internet Explorer blogisivut. Viitattu 17.2012. http://windowsteamblog.com/ie/b/ie/archive/2011/03/09/a-more-beautiful-web-launches-on-march-14th.aspx

Three.js Documentation. 2012. Three.js JavaScript 3D-kirjaston virallinen dokumentaatio. Viitattu 7.12.2012. http://mrdoob.github.com/three.js/docs/53/

Tom's Guide- Web Life. 2011.Tom's Guide teknologiasivuston artikkeli 9.3.2011. Viitattu 17.9.2012. http://www.tomsguide.com/us/google-chrome-apple-safari-firefox-ie9,news-10412.html

Tutorial's point. 2012a. JavaScript Overview. Viitattu 7.11.2012. http://www.tutorialspoint.com/javascript/javascript_overview.htm

Tutorial's point. 2012b. jQuery Overview. Viitattu 7.11.2012. http://www.tutorialspoint.com/jquery/jquery-overview.htm

WebGL-käyttöliittymän toteutuksesta 2010. Raportti. Viitattu 12.9.2012, http://code.google.com/p/webhierarkia/wiki/WebGLRaportti

WebGL Stats. 2012. Who has WebGL. Viitattu 11.9.2012, http://webglstats.com/

What is PHP. 2012. PHP:n virallisten sivujen PHP:n esittelyluku. Viitattu 8.11.2012. http://www.php.net/manual/en/intro-whatis.php

When can I use n.d. When can I use WebGL? Viitattu 12.9.2012, http://caniuse.com/webgl

X3D FAQ. 2005. Web3D consortium. Viitattu 17.9.2012. http://www.web3d.org/about/faq/