LAB University of Applied Sciences

# Web-animaatiot

## CASE: Fazer Leipomot Oy, työturvallisuuspelin animaatiot

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
Kevät 2021
Riina Olenius

**Tiivistelmä**

| Tekijä(t) | Julkaisun laji | Valmistumisaika |
|---|---|---|
| Olenius, Riina | Opinnäytetyö, AMK | Kevät 2021 |
| | Sivumäärä | |
| | 33 | |

Työn nimi

**Web-animaatiot**

CASE: Fazer Leipomot Oy, työturvallisuuspelin animaatiot

Tutkinto

Insinööri (AMK), Tieto- ja viestintätekniikka

Toimeksiantajan nimi, titteli ja organisaatio

Fazer Leipomot Oy


Tiivistelmä

Työn tavoitteena oli tutkia web-animaatioiden kehittymistä erilaisten animointitapojen ja sovellusten avulla vuosien aikana. Case-osuuden tavoitteena oli käydä läpi, mitä menetelmiä ja sovelluksia Fazer Leipomot -työturvallisuuspelin animaatioissa käytettiin. Työssä käytetty materiaali löytyi verkon eri artikkeleista ja teoksista.

Työssä edettiin tekemällä vertailuja erilaisten animointitapojen ja tiedostokokojen välillä sekä tarkasteltiin mitä animointitapoja ja menetelmiä voitiin käyttää kunkin sovelluksen kohdalla. Työssä käytiin myös läpi, kuinka ongelmatilanteet vaikuttivat laatuun sekä suorituskykyyn animaatioissa ja verkkosivustoilla. Lisäksi työssä kerrottiin lyhyesti, mikä animointitapa ja sovellus on kyseessä kussakin tilanteessa. Työn käytännön osuudessa käsitellään Fazer Leipomoille toteutetun sovelluksen visuaalisten osuuksien animointia, jotta saatiin vertailukohde tämän päivän web-animaatioita varten. Case-osuudessa myös kerrottiin tuotetuista animaatioista ja niitä verrattiin tiedostojen kokojen välillä.

Työn tuloksena todetaan, että aiheen myötä saadaan käsitys siitä, miten tänä päivänä web-animaatioita suositellaan toteutettavan ja mitä ohjelmia niiden toteuttamiseen käytetään. Animaatioissa on otettava laatuun ja suorituskykyyn toimintamalli, jossa niitä yritetään parantaa kaikilla mahdollisilla tavoilla, jotta ne toimisivat verkkosivustoilla odotetulla tavalla. Animaatioprojektissa päädytään johtopäätökseen, että animaation laadulla, suorituskyvyllä ja tiedostokoolla on tärkeä merkitys toiminnallisesti, jotta se toimisi verkkosivustoilla parhaimmalla mahdollisella tavalla.



Asiasanat

web-animaatio, Adobe Animate, laatu ja suorituskyky, tiedostokoko, Fazer Leipomot -animaatio

**Abstract**


| Author(s) | Type of Publication | Published |
|---|---|---|
| Olenius, Riina | Thesis, UAS | Spring 2021 |
| | Number of Pages | |
| | 33 | |


Title of Publication

**Web Animations**

Fazer Bakery Ltd. Case, Animations for Work Safety Game

Name of Degree

Engineer (UAS), Information and Communication Technologies

Name, title and organization of the client

Fazer Bakery Ltd.


Abstract

The objective of this study was to examine the evolution of web animations it is behavior of animating and the use of applications in the past few years. The objective of the empirical part was to observe and process the methods of animating and the use of applications on project of Fazer Bakery Ltd. The materials used in this paper were gathered from literature and Internet articles.

The theory section compared different animation methods and the file sizes of their respective formats. Each theory section starts with an overview of the animation methods and applications. Furthermore, it examined which animation methods and techniques are being used in applications. Moreover, it observed and processed the problematic situations of quality and performance in animations and browsers.

The empirical part consists of explaining and observing an animation project for Fazer Bakery Ltd. It compared the final versions of the project's animations to today's web animations. In addition, animation methods and the sizes of their respective outputs are compared.

The results of the study can be used to implement web animations with recommended animation methods and the paper provides recommendations on which applications should be used to create web animations. Thus, animations should be developed with a compromise between animation quality, and performance in mind if they are supposed to be used on websites. Based on the animation project's results, the quality, performance and file size of the animations are the most important functional factor in web animation development for websites.



Keywords

web animation, Adobe Animate, quality and performance, file size, animation of Fazer Bakery

Sisällys

# 1 Johdanto

Opinnäytetyön tavoitteena oli tutkia ja selvittää, miten web-animaatiot ovat kehittyneet vuosien aikana. Työssä haluttiin selvittää, ovatko web-animointitavat mahdollisesti muuttuneet jotenkin vuosien aikana ja mitä ohjelmistoja tai työkaluja käytetään niiden toteutukseen. Opinnäytetyössä haluttiin myös selvittää, mitä tämän päivän animointitavoista käytetään missäkin tilanteessa ja käytetäänkö tiettyjä tapoja muita enemmän.

Animaatiosovelluksien kohdalla pohdittiin, mitä niistä kannattaa käyttää ja missä tilanteessa kutakin ohjelmaa käytettäisiin animoinnissa. Samalla tutkittiin, kuinka animaation tekeminen eroaa sovelluksien välillä ja voitaisiinko erilaisia ohjelmia käyttää animaation eri vaiheissa, vai tuleeko samaa ohjelmaa käyttää alusta loppuun animaation teossa. Lisäksi työssä käytiin läpi, sopisivatko kaikki ohjelmat samaan tarkoitukseen vai onko ohjelmat suunniteltu erilaisiin käyttötapoihin ja mitä animointimetodeja tutkittavat ohjelmat käyttävät. Työssä haluttiin myös saada selvyys siitä, käyttävätkö ohjelmat vain osaa vai tiettyjä animointitapoja animaation tuotoksessa.

Opinnäytetyössä pohdittiin laadun ja suorituskyvyn mahdollisia ongelmatilanteita web-animaatioissa ja miten näihin ongelmiin saataisiin ratkaisu. Tutkittiin myös, missä tilanteissa suorituskyky ja laatu mahdollisesti laskevat verkkosivustolla. Lisäksi käytiin läpi, mitä virheitä kannattaa välttää suorituskyvyn ja laadun suhteen.

Tiedostoformaateista haluttiin selvyys siitä, kuinka ne eroavat koon osalta. Haluttiin myös selvittää, vaikuttaako tiedostotyyppi tai millä sovelluksella web-animaatiota tuotetaan. Lisäksi pohdittiin, havaitaanko niissä huomattavia eroja vai ei, ja vaikuttaako kokoerot animaation latausaikoihin verkkoselaimessa.

Case-osuudessa keskitytään avaamaan havaintoja, joita ilmaantui Fazer Leipomot -animaatioprojektin aikana. Selvitettiin, mitä animaatiotapoja ja -työkaluja sen toteutukseen käytettiin. Osuudessa myös pohdittiin, oliko laadulla ja suorituskyvyllä vaikutuksia animaation tuotokseen. Kuinka tiedoston koko vaikutti lopputulokseen ja miten animaatiot toimivat itse selainpohjaisessa työturvallisuuspelissä. Lisäksi käytiin läpi, tuliko animaation tekemiseen jotain muutoksia riippuen siitä toimiko se verkkoselaimessa vai ei ja mitä ongelmatilanteita tuli vastaan valituilla animointitavoilla ja työkaluilla. Selvitettiin myös, kuinka niistä ongelmatilanteista saatiin ratkaisu, ja pohdittiin myös ongelmatilannetta, jossa ei saatu ollenkaan ratkaisua ja täytyikö muuttaa animaatiosuunnitelmaa ja toteutusta sen seurauksena.

# 2 Web-animaation kehitys

## 2.1 GIF-animaatiot

GIF-animaatiolla tarkoitetaan usein lyhyttä uudelleen toistettavaa animaatioleikettä, joka alkaa automaattisesti alusta animaation loputtua. Seuraavaksi selvitetään mistä GIF-animaation juuret alkoivat. Lisäksi kuvaillaan, kuinka sen kehitys eteni.

GIF-animaatio sai alkunsa kuvatiedostomuodosta. Lyhenne GIF tulee englannin sanoista “Graphics Interchange Format”. GIF oli ensimmäisiä yleisesti käytössä olleita kuvaformaatteja. (Eppink ym. 2014, 298.) Siitä se kuitenkin kehittyi eteenpäin.

Vuonna 1987 CompuServe-yhtiön insinööri Steve Wilhiten luoma GIF on kuvatiedostomuoto, joka käytti häviötöntä pakkausmenetelmää. Häviöttömyys erotti GIF-tiedoston muista staattisista kuvatiedostoformaateista, kuten esimerkiksi JPEG tai PNG. Sen lisäominaisuutena oli silmukkamenetelmä, jolla toistettiin peräkkäisiä kuvia yhä uudestaan. GIF-muodossa toteutettu animaatio voi olla mahdollisesti pienempi tiedostokooltaan kuin esimerkiksi videoformaattiin tuotettu lopputulos. Lisäksi GIF-tiedostolla ei käytetä samoja yleisiä resoluutiota kuin videotiedostolla. Internetin alkuaikoina GIF-animaatio oli ideaalinen tapa lisätä visuaalista sisältöä ja liikettä verkkosivustolle, kun kaistanleveys oli rajallinen ja video- ja kuvankäsittelyohjelmat olivat vähemmän kehittyneitä. (Milfner ym. 2017, 2.)

Tänä päivänä GIF-kuvatiedostomuodosta on hyötyä muuhunkin tarkoitukseen. GIF-tiedostoja kohdataan käyttäjien yksityisillä tietokoneiden näytöillä. Niitä voidaan luoda, käyttää, lähettää, kerätä, kopioida, muokata tai suorittaa. Tänä päivänä GIF-tiedostotyyppiä usein käytetään GIF-animaatioina tai muuten vain lyhyinä, mykkinä, silmukkoina tai nimettöminä liikkuvina kuvina. Niillä on tyypillisesti tekijä, jota ei tunneta tai paljasteta. Yksittäinen käyttäjä voi kohdata ne henkilökohtaisella näytöllä, jossa niitä saattaa ympäröidä teksti tai muu media. Lisäksi niitä käytetään kuvaamaan käyttäjien identiteettiä eli ne ovat avatarin kaltaisia pikkukuvia, jossa tapahtuu liikettä. (Eppink ym. 2014, 298.)

Tästä voidaan todeta, että GIF-animaatioita käytetään yhä paljon tänä päivänä. Niitä näkyy kaikkialla joka päivä eikä niitä voi olla huomaamatta. Kuten jo mainittiin, GIF-animaatioita käytetään mm. tervehdykseksi tai osoittamaan mielialaa erilaisissa tilanteissa. Tämän perusteella voidaan olettaa, että niitä käytettäisiin vieläkin hyvin paljon esimerkiksi sosiaalisen median verkkosivustoilla ja viestintävälineissä.

## 2.2 CSS-transitiot

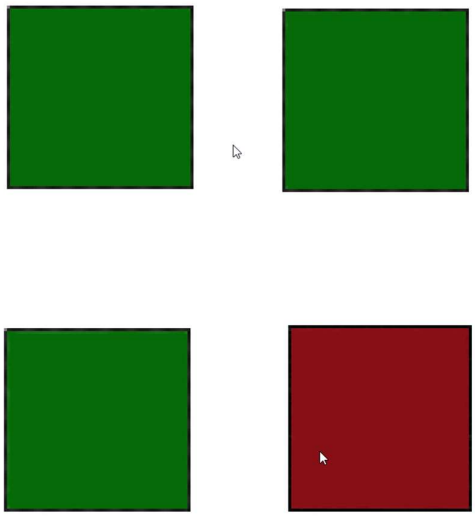
CSS termillä tarkoitetaan tyylikieltä, jolla voidaan tyylitellä HTML-dokumentti. CSS määrittelee, miltä HTML-elementit tulevat näyttämään verkkosivustolla. (Refsnes Data 2021a.) Lyhenne CSS tulee englannin sanoista ”Cascading Style Sheets”.

Transition avulla voidaan määrittää animaation nopeus sekä luoda viiveitä tarpeen mukaan. Niiden avulla voidaan myös muuttaa käytössä olevien elementtien tilaa ja käyttäytymistä animaatiossa. Transitiot auttavat kehittäjiä toteuttamaan sujuvan animaation. Lisäksi transitioilla voidaan näyttää käyttäjille verkkosivuston tilan tai prosessin muutokset. (Shenoy ym. 2013, 105.) Esimerkkinä transitiosta olisi painikkeen pieni muutos, kun hiiri viedään sen päälle. Kuten kuvassa 1 esitetään, style-tagin alle voidaan lisätä pari riviä koodia, jolla saadaan aikaan painikkeen pienet muutokset. Tässä koodissa määritetään painikkeelle skaalauksen ja rotaation muutos, kun hiiri viedään sen päälle.

```
<style>
.button {
margin: 15px;
transition: transform 0.3s;
}
.button: hover {
transform: scale(1.05) rotate(-2deg);
}
</style>
```

Kuva 1: Painikkeen pienet muutokset transitio esimerkki

Toisena transitio esimerkkinä voisi olla kaksi vihreää suorakulmaista alkiota vierekkäin, kun hiiri siirretään oikean suorakulmaisen alkion päälle niin sen väri muuttuu punaiseksi. Oikeanpuoleinen suorakulmainen alkio pysyy punaisena, kunnes hiiri siirretään pois sen päältä (kuvio 1). Transitiolla voidaan myös määrittää, kuinka nopeasti väri vaihtuu suorakulmaisesta alkiosta. Tämän lisäksi transitiolla on muita ominaisuuksia, joita se voi käyttää hyödykseen kuten mm. fontin koko, reunojen säde ja maksimi leveys sekä korkeus. (Shenoy ym. 2013, 107.) Aiemmin oli huomioitava mm. se, että valtaselain Internet Explorer ei tukenut SVG-formaattia, joten CSS-kielen transformaatioita ei voinut hyödyntää (Shapiro 2015, 108).



Kuvio 1: Suorakulmaisen alkion värin muutos (mukailtu Shenoy ym. 2013, 107)

## 2.3 CSS-animaatiot

CSS-animaatiolla tarkoitetaan CSS-tyylittelyllä luotuja animaatioita. Tällä tekniikalla voidaan animoida muun muassa tekstejä, kuvioita, kuvia ja verkkosivustoa itsessään. Sillä saadaan verkkosivusto ns. heräämään eloon sekä antamaan käyttäjälle interaktiivisuuden tunnetta verkkosivustoa selatessaan.

Kun tuotetaan CSS-animaatiota, on olemassa vain rajattu määrä ominaisuuksia, joita on mahdollista käyttää CSS-tyylittelyn avulla (Shapiro 2015, 10). CSS-animaatiot voidaan ajatella sarjana transitioita. Ensimmäinen askel omien animaatioiden luomisessa on määritellä keyframe-pisteet, jotka asettavat transition alku- ja lopetus kohdat. Yksinkertaisimmassa animaatiossa on kaksi keyframe-pistettä, yksi alussa ja yksi lopussa, kun taas monimutkaisemmissa animaatioissa on useita keyframe-pisteitä lohkojen välissä. (Gasston 2014, 175.)

CSS-animaatio menetelmällä hallitaan koko käyttöliittymän interaktiivisuutta CSS-tyylittelyn avulla, joka saattaa nopeasti muuttua monimutkaiseksi. CSS-animaatiolla ei ole realistista animaation ajan kontrolloijaa, joka olisi hyödyllinen animaation suunnittelussa käyttöliittymään, jossa animaatiot reagoisivat käyttäjän syötteeseen. CSS-animaatio menetelmällä luovutaan myös fysiikkapohjaisesta liikkeestä animaation objektien kanssa, joka olisi antanut niille mahdollisuuden käyttäytyä kuten objektit oikeassa maailmassa. Lisäksi verkkoselainten vanhemmat versiot eivät välttämättä tue CSS-animaatioita. (Shapiro 2015, 4.)

## 2.4 HTML Canvas

HTML Canvas -elementillä voidaan piirtää erilaisia polkuja ja muotoja. Sillä voidaan myös luoda ympyröitä, kaaria ja viivoja. Piirrettyihin polkuihin voidaan muun muassa lisätä hiireen vaikuttavia tapahtumia ja interaktiivisuutta. (Makzan 2011, 186.) On kuitenkin ymmärrettävä, että HTML Canvas-elementti ei piirrä grafiikkaa itse, vaan se käyttää JavaScript-ohjelmointikieltä hyödykseen (Refsnes Data 2021b).

Canvas-elementtiin on myös mahdollista luoda monimutkaisia animaatioita JavaScriptin avulla. Animaatio syntyy ja kehittyy koodirivi kerrallaan, kun canvas-elementtiin piirretään kohta kohdalta. HTML Canvas -animaation suurin rajoitus on se, kun muodot on piirretty niin ne pysyvät sellaisinaan, joten jokainen muoto on piirrettävä uudestaan animaation aikana. (MDN Web Docs & Mozilla 2021.)

Perus HTML Canvas -animaatio aloitetaan tyhjentämällä canvas-elementti clearRect()-metodilla, jos taas piirrettävät muodot täyttävät koko canvas-elementin tätä vaihetta ei tarvitse tehdä. Canvas-elementin alkuperäinen tila on myös tallennettava, jos jotain ominaisuutta muutetaan (mm. tyylit, transformaatio, jne.), joka vaikuttaa sen tilaan ja halutaan varmistaa, että sen alkuperäistä tilaa käytetään jokaisen freimin piirtämiseen. Tämän jälkeen piirretään animoidut muodot, jossa varsinainen frame rendering -prosessi määritellään. Lisäksi canvas-elementin alkuperäinen tila on palautettava ennen uuden freimin piirtämistä, jos se on tallennettu aiemmin. (MDN Web Docs ym. 2021.)

Animaatiota kontrolloidaan piirretyillä muodoilla canvas-elementissä. Tämä määritellään suoraa käyttämällä canvas-metodeja tai kutsumalla mukautettuja funktioita koodissa. Normaalissa tilanteessa nämä tulokset vain ilmestyvät canvas-elementtiin, kun JavaScript saa ajettua prosessin loppuun asti. Esimerkiksi animaatiota ei ole mahdollista toteuttaa for-luupilla. Tällä tarkoitetaan, että animaation tuottamiseen tarvitaan tapa, jolla piirto funktioita voidaan määrittää ajan kuluessa. (MDN Web Docs ym. 2021.)

On olemassa funktioita, joilla ajoitettu päivitys voidaan toteuttaa:

- setInterval(function, delay)
- setTimeout(function, delay)
- requestAnimationFrame(callback).

Tämän listan ensimmäinen funktio aloittaa ajamalla sille määritetyn funktion joka kerta ja sille määritetyllä viiveellä millisekunneissa. Toinen funktio ajaa määritetyn funktion sille määritetyllä viiveellä millisekunneissa. Kolmas funktio kertoo selaimelle, että animaatio

halutaan esittää ja pyytää selainta kutsumaan sille määritetyn funktion, joka päivittää animaation ennen sen seuraavaa piirtovaihetta. (MDN Web Docs ym. 2021.)

## 2.5 SVG-kuvat

SVG-kuvalla tarkoitetaan vektorikuvaa, jonka tarkkuus ei muutu, vaikka sitä skaalattaisiin tai sen kulmaa käännettäisiin. SVG-kuvan kokoa voidaan muuttaa ja sen alkuperäiset tiedot säilyvät samana riippumatta siitä, kuinka monta kertaa sen muotoa muutetaan. SVG-kuvat ovat hyödyllisiä animaation teossa siitä syystä, että niiden tarkkuus ei laske vaikka mitä animaation aikana tapahtuisi.

SVG-elementit eroavat HTML-elementeistä, sillä tavalla, että niillä on uniikit tagit, attribuutit ja käyttäytyminen, joka antaa niiden määrittää graafisia muotoja. SVG-elementit voivat luoda kuvia koodaamisen tuloksena. Lisäksi SVG-elementtejä voidaan ohjelmoinnilla tyylitellä ja animoida monimutkaisia muotoja käyttäen hyväksi JavaScript -ja CSS-koodia. (Shapiro 2015, 104.)

Kuten HTML-elementeille, SVG-elementeille voidaan asettaa tapahtumakäsittelijöitä, jotka reagoivat käyttäjän syötteeseen. Tämä tarkoittaa sitä, että verkkosivuston grafiikasta voidaan tehdä interaktiivinen. Esimerkiksi kaikki nappulat verkkosivustolla voidaan muuttaa animoiduksi grafiikaksi. (Shapiro 2015, 104.)

Lisäksi monella kuvankäsittelyohjelmalla voidaan viedä tehdyt työt SVG-formaattiin, joka voidaan nopeasti kopioida ja liittää HTML-ohjelmointikoodin sekaan. Kuvankäsittelyohjelmia, joilla voidaan tehdä tämä prosessi, on esimerkiksi Photoshop-, Sketch- ja Inkscape-sovellukset. Nämä kolmannen osapuolen ohjelmat auttavat kehittäjiä tai käyttäjiä tuottamaan SVG-formaatissa olevia objekteja ilman, että kehittäjien tai käyttäjien täytyisi olla taiteilijoita tai grafiikkasuunnittelijoita. (Shapiro 2015, 104.)

# 3 Sovellukset web-animaatioiden luontiin

## 3.1 Adobe Animate

Adobe Animate -sovellus on animaationluontiohjelma, jolla voidaan luoda animaatioita keyframe-pisteitä käyttäen. Animaatioita voidaan myös animoida yksi keyframe-piste kerrallaan tai antaa ohjelman laskea animaation kulku keyframe-pisteestä pisteeseen. Animaatiot syntyvät joko ohjelmaan tuoduilla objekteilla tai alusta alkaen piirretyillä muodoilla.

Animate-ohjelmalla voidaan mm. tuottaa HTML Canvas -ohjelmointikoodia suoraan tuotetusta animaatiosta ohjelman sisällä. Ohjelma tulostaa HTML-tiedoston, joka sisältää JavaScriptiä ja Canvas-elementin. Ohjelmakoodi sisältää jokaisen animaatio vaiheen, jota käytettiin animaation toteutuksessa. Lisäksi ohjelma tulostaa automaattisesti JavaScript -tiedoston, joka sisältää kaiken tarvittavan ohjelmakoodin, mitä HTML-tiedosto tarvitsee toimiakseen. JavaScriptillä voidaan myös lisätä interaktiivisuutta Animate-ohjelman sisällä tai sitä voidaan lisätä jälkeenpäin valmiiseen julkaistuun tiedostoon (Russell 2019).

Animate-ohjelmalla toteutetaan myös multimediaa. Multimediaa voidaan luoda monelle alustalle ja monella toistomenetelmällä. On ymmärrettävä etukäteen mihin tarkoitukseen lopullista animaatiota halutaan käyttää, jotta tiedetään mikä dokumenttityyppi valitaan, kun luodaan uusi tiedosto Animate-projektiin. (Russell 2019.) Tästä voidaan todeta, että dokumenttityypillä on tärkeä merkitys toteutuksen suhteen.

On kuitenkin ymmärrettävä, että kaikkia ominaisuuksia ei tueta kaikilla tiedostotyypeillä. Esimerkiksi HTML Canvas -dokumentti ei tue 3D Rotation- tai Translation-työkaluja. Työkaluja, joita ei tueta tietyn dokumenttityypin kanssa ei voi valita Animate-käyttöliittymässä. (Russell 2019.)

Animaation valmiissa lopputuloksessa käytetään playback -tai runtime -ympäristöä, joka toistaa sen sisällön. Animaation sisältö voidaan toistaa mm. HTML5:lla ja JavaScriptillä verkkoselaimessa. Animaatio voidaan myös upottaa YouTubeen tai sitä voidaan toistaa sovelluksena mobiililaitteella. Ensin tehdään valinta, joka määrittää mitä dokumenttityyppiä kannattaa käyttää missäkin tilanteessa. (Russell 2019.)

Kaikki Animate-projektit tallennetaan FLA- tai XFL-tiedostomuotoon riippumatta siitä, millä playback -ympäristöllä ja dokumenttityypillä se on toteutettu. Ainoa huomattava ero dokumenttityyppien välillä on niiden konfiguraatio julkaista erilaisia valmiita tiedostoja tarvittavaan tilanteeseen. Animate-ohjelma tukee vain ActionScript 3.0. -versiota, joten jos tarvitaan vanhempaa versiota, niin voidaan käyttää Flash Professional CS6 -ohjelmaa tai

aiempaa. (Russell 2019.) On kuitenkin huomioitava, että Flash Professional CS6 -ohjelmaa ei välttämättä enää käytetä tänä päivänä ja tuki flash-animaatioille on poistumassa selaimista.

ActionScript 3.0 -versio valitaan silloin, jos halutaan luoda animaatiota ja interaktiivisuutta, johon halutaan luoda animaatio, joka julkaistaan HD-video muodossa. Se on viimeisin ohjelmointikieli versio, jota käytetään Animate-ohjelmassa sekä se on samankaltainen JavaScript-ohjelmointikielen kanssa. Kun valitaan ActionScript 3.0 -dokumenttityyppi, niin se ei tarkoita, että on pakko käyttää ActionScript-ohjelmointikoodia. Lisäksi ActionScript 3.0 -versiota käytettiin paljon Flash Playerille toteutetuille videoille ennen vuotta 2020. ActionScript 3.0 -dokumentti on kuitenkin monipuolisin dokumenttityyppi Animate-ohjelmassa, joka tukee laajan variaation piirto- ja animaatio-ominaisuuksia. Siitä voidaan julkaista mm. sprite sheet -elementtejä, PNG-kuvasarjoja tai valmiita korkearesoluutioisia videoita. (Russell 2019.) Tänä päivänä verkkoselaimet eivät kuitenkaan enää tue Flash Player -laajennusta.

## 3.2 After Effects

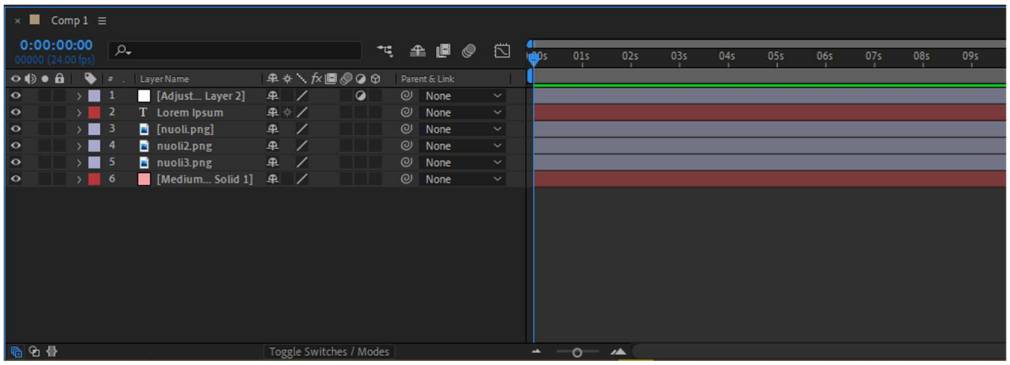
Adobe After Effects -sovellus on videonkäsittelyohjelma, jolla voidaan tehdä jälkikäsittelyä videoleikkeelle tai luoda animaatioita aivan alusta saakka. Videoleikkeitä voidaan muokata, leikata tai niihin voidaan lisätä monipuolisia efektejä. Tarpeen mukaan videoleikkeisiin voidaan myös lisätä ääntä.

Kun työskennellään After Effects -ohjelman kanssa, on huomioitava, että tätä ohjelmaa yleensä käytetään animoimaan tai yhdistelemään materiaalia, joka on luotu toisessa ohjelmassa. Esimerkiksi materiaalia voidaan tuottaa ja tuoda ohjelmista kuten Adobe Photoshop ja Illustrator. (Jerron ym. 2014, 82.) Kun kyseessä on monimutkaisempi grafiikka, After Effects -ohjemalla tuotetaan harvemmin omaa materiaalia ohjelman sisällä.

After Effects -projektissa animaatiot tai videonkäsittelyt koostuvat joko yhdestä tai useasta kompositiosta. Kompositiot voidaan luoda alusta alkaen tai tuoda muista After Effects -projekteista. Ohjelma myös luo automaattisesti komposition Photoshop -tai Illustrator -tiedostosta, jos se sisältää useamman layer-tason. (Jerron ym. 2014, 83.) Tämä osoittaa sen, että kompositiolla on tärkeä merkitys After Effects -projektissa.

After Effects -ohjelmassa käytetään layer-elementtejä. Layer-elementit toimivat hieman eri tavalla verrattuna Photoshopin layereihin. Seuraavaksi selitetään lyhyesti, kuinka layer-elementit eroavat kyseisestä kuvankäsittelyohjelmasta.

Jokainen kuvamateriaali, joka sijoitetaan kompositioon, on erillisellä layer-elementillä. Nämä layer-elementit ovat näkyvillä aikajanapaneelilla ja jokainen layer-elementti sisältää omat ominaisuutensa, joita voidaan manipuloida tai animoida muista elementeistä riippumatta. Jokainen layer-elementti myös sisältää alku- ja loppupisteen, joka määrittää elementin keston. (Jerron ym. 2014, 89.) Kuvassa 2 kuvataan miltä aikajanapaneeli ja sen layer-elementit näyttävät After Effect -ohjelmassa.




Kuva 2: Esimerkki After Effect -ohjelman aikajanapaneelista ja sen layer-elementeistä

## 3.3 Photoshop

Adobe Photoshop -sovellus on kuvankäsittelyohjelma, jolla voidaan käsitellä ja muokata kuvia tai kuvioita. Ohjelmalla voidaan myös toteuttaa grafiikkaa alusta loppuun asti. Siitä löytyy muitakin hyödyllisiä ominaisuuksia, joita voidaan käyttää hyödyksi myöhemmin animointiprosessin yhteydessä.

Photoshop-ohjelma tarjoaa työkaluja, joilla voidaan lisätä tekstiä kuviin tai grafiikkaan. Tekstin kokoa voidaan muuttaa. Lisäksi tekstiä voidaan muokata visuaalisten efektien avulla. (Dayley & Dayley 2013, 5.) Jos efektit teksteille tehdään tällä ohjelmalla, niin niitä ei välttämättä tarvitse lisätä enää myöhemmin, kun tarvittaessa siirrytään toiseen käsittelyohjelmaan jatkamaan työskentelyä.

Toinen alue, jossa ohjelma auttaa web-animaation kannalta on sen tapa valmistella kuva verkkosivustoa varten. Photoshop tarjoaa työkaluja, joilla kuvatiedostoja saa säädettyä helposti kuvan koon, tiedostomuodon ja värien määrän osalta siten, että ne soveltuvat käytettäviksi verkkosivuilla. On myös olemassa työkalu, jolla voidaan pilkkoa kuva klikattavaksi alueiksi. Se luo tarvittaessa HTML-ohjelmointikoodin, jolla voidaan manipuloida pilkottuja alueita verkkosivustolla. (Dayley ym. 2013, 5.) Tästä voidaan todeta, että Photoshop-ohjelman avulla voidaan saada kuvista verkkosivusto ystävällisempiä ja niitä voidaan myöhemmin käyttää animaation toteutusvaiheessa.

Vaikka Photoshop-ohjelmalla on paljon pätevyyttä vektoripolkujen tekoon, on suotavaa työskennellä Adobe Illustrator -ohjelmalla niiden kanssa. Adobe Illustrator -ohjelma kykenee työskentelemään monen vektoriobjektin kanssa samaan aikaan sujuvasti. (Dayley ym. 2013, 6.) Tästä voidaan päätellä, että Photoshop-ohjelmalla voidaan työskennellä vektoriobjektien kanssa, mutta on olemassa muita käsittelyohjelmia, joilla se sujuu luontevammin.

Photoshop-ohjelmalla voidaan myös tarvittaessa lisätä animaatiota kuviin. Animoidut kuvat voivat lisätä visuaalista ilmettä ja se antaa mahdollisuuden luoda lyhyitä animoituja pätkiä verkkosivustolle. (Dayley ym. 2013, 6.)

Photoshop-ohjelmassa on olemassa aikajanapaneeli, joka hallitsee animaation tai videoleikkeen aikajanaa. Aikajanapaneelilla voidaan liikkua ja valita tietty ajankohta mahdollisten tiedostojen ja listattujen leikkeiden välillä, jotka on laitettu aikajanalle. Näin aikajanalla päästään käsiksi leikkeiden ominaisuuksiin, joka antaa animoida jokaista leikettä projektissa erilaisilla tavoilla riippuen siitä minkä tyyppinen leike on valittuna. (Dayley ym. 2013, 878.)

## 3.4 jQuery

jQuery on JavaScript-kirjasto, jolla voidaan luoda animaatioita kirjaston koodimetodeja käyttäen. jQuery-kirjastolla voidaan yksinkertaistaa JavaScript-ohjelmointia. (Refsnes Data 2021c.) Tämä tarkoittaa käytännössä sitä, että jQuery-kirjastolla voidaan käyttää koottua kokoelmaa JavaScript-koodeja, joille on annettu tietyt toiminnot (MacLees 2014, 29).

jQuery sisältää yksinkertaista ja lyhyttä ohjelmointikoodia, jolla saadaan tarvittavat JavaScript toiminnot sujuvasti ja helposti tehtyä. jQuery-kirjasto myös on viimeistelty huolellisesti, jotta sen ohjelmointikoodi toimisi kaikilla erilaisilla verkkoselaimilla. Lisäksi on muistettava, että jQuery ei ole oma ohjelmointikielensä vaan se on JavaScript-kirjasto, sillä on kuitenkin samat säännöt ja sillä koodataan samalla tavalla kuin JavaScriptillä. jQuery ei ole kuitenkaan haastava oppia, vaan se tekee JavaScript-ohjelmointikoodin kirjoittamisesta paljon helpompaa, vain parilla koodirivillä voidaan saada paljon aikaan. (MacLees 2014, 29.)

jQuery-kirjastolla voidaan mm. luoda pudotusvalikko nopeasti sekä se voi olla myös animoitu ja se toimii sujuvasti monella verkkoselaimella. Näin esimerkkinä jQuery-kirjastolla usein valitaan elementit, joilla halutaan työskennellä, kun lisätään efektejä navigaatiovalikkoon. Aloitetaan valitsemalla navigaatiovalikon sisältämät alkiot ja tässä käytetään selectors-työkaluja, jolla valitaan kaikki ne elementit, joita halutaan työstää verkkosivustolla. jQuery on lainannut selectors-työkaluja CSS-tyylikielestä CSS3-versioon asti ja ne toimivat jopa verkkoselaimissa, jotka eivät välttämättä edes vielä tue CSS3:sen selectors-työkaluja.

Vaikka CSS-tyylikieli tarjoaakin laajan variaation selectors-työkaluja, jQuery-kirjastolla on muutama oma ominaisuutensa, jolla voidaan helposti päästä käsiksi vain niihin elementteihin, joita halutaan työstää tietyssä vaiheessa. (MacLees 2014, 31.) jQuery ei kuitenkaan tarjoa kattavaa tukea SVG-objektien tyyliominaisuuksien animointiin, eikä esitystapaan liittyvien attribuuttien muokkaamiseen (Shapiro 2015, 108).

jQuery-kirjastolla voidaan esimerkiksi lisätä tekstiä HTML-dokumentin body-elementtiin HTML-ohjelmointikoodia hyväksi käyttäen ja se saadaan hoidettua yhdellä koodirivillä. Jos tämä tuotetaan JavaScript-ohjelmointikoodilla, niin JavaScriptillä tarvitsisi kirjoittaa monta koodiriviä verrattuna jQuery-kirjastoon. (MacLess 2014, 31–32.) Tämän esimerkin avulla voidaan todeta, että jQuery-kirjastolla saadaan tuotettua ohjelmointikoodia helposti ja nopeasti.

Animaation tuottaminen jQuery-kirjastolla on myös parin koodirivin päässä ja tätä käydään läpi esimerkin avulla. Jos halutaan animoida kuva ilmestymään esiin, niin voidaan käyttää jQuery-kirjaston valmista fadeIn-funktiota ja, jos taas halutaan häivyttää kuva pois näkyvistä, voidaan käyttää fadeOut-funktiota sen tekemiseen (MacLess 2014, 32.) Kuvassa 3 näkyy jQuery esimerkki fadeIn- ja fadeOut-funktioista. Kun painetaan nappulaa, jolle on määritelty fadeOut-funktio, niin teksti haihtuu pois näkyvistä sille määritetyllä viiveellä millisekunnin tarkkuudella ja sen jälkeen paragraph-elementille kerrotaan, että sitä ei ole enää olemassa. Tästä seuraa tapahtuma, jossa nappulat siirtyvät paragraph-elementin paikalle. Jos taas painetaan nappulaa, jolle on määritelty fadeIn-funktio niin paragraph-elementti varaa itselleen paikan nappuloiden yläpuolelta ja sen jälkeen se ilmestyy näkyviin sille määritetyllä viiveellä millisekunnin tarkkuudella. (Refsnes Data 2021d.)

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $(".btn1").click(function(){
    $("p").fadeOut(1000);
  });
  $(".btn2").click(function(){
    $("p").fadeIn(1000);
  });
});
</script>
</head>
<body>

<p>This is a paragraph.</p>

<button class="btn1">Fade out</button>
<button class="btn2">Fade in</button>

</body>
</html>
```

Kuva 3: jQuery esimerkki fadeIn- ja fadeOut-funktioista (mukailtu Refsnes Data 2021d)

# 4 Laatu ja suorituskyky

Animaatiot vievät todella paljon resursseja verkkoselaimesta toimiakseen. On olemassa paljon tekniikoita, joilla voidaan auttaa selainta toimimaan niin tehokkaasti kuin mahdollista. (Shapiro 2015, 118.) Seuraavaksi käydään läpi, minkälaisia ongelmia liittyy laatuun ja suorituskykyyn, ja minkälaisia tekniikoita ja menetelmiä voidaan käyttää hyödyksi laadun ja suorituskyvyn parantamiseksi.

Käyttöliittymäsuunnittelun näkökulmasta ei ole ongelmaa löytää suorituskykyyn liittyviä artikkeleita, joiden avulla voidaan rakentaa mobiililaitteille ystävällisiä verkkosivustoja. Päinvastoin kehittäjät voivat olla tietämättömiä, kuinka olisi kannattavinta testata tai seurata käyttöliittymän suorituskykyä. Lisäksi suorituskykytestien laatiminen voidaan kokea usein ylivoimaisena verkkoselaimesta ja laitteesta riippuen. Tämä tarkoittaa käytännössä sitä, että on olemassa laaja määrä erilaisia verkkoselaimia ja laitteita, jotka saattavat vaikeuttaa suorituskyvyn mittaamista ja tasapainottamista. Kun otetaan huomioon, että nämä alustat päivittyvät jatkuvasti, niin kehittäjät saattavat luovuttaa ja sivuuttaa suorituskykyyn liittyviä ongelmia. Kehittäjät uskottelevat itsellensä, että laitteet kehittyvät nopeammiksi, kun käyttäjät jatkuvasti päivittävät laitteistoaan, silloin kehittäjien verkkosivustoista tulisi merkittävästi suorituskykyisempiä. Näin todellisuudessa ei kuitenkaan tapahdu, vaan älypuhelimet, jotka vastaanotetaan kehittyvässä maailmassa jäävät suorituskyvyn jalkoihin. (Shapiro 2015, 118.)

Toisena esimerkkinä suorituskyvylle on olemassa realistinen tapahtuma, jossa saatetaan systemaattisesti tehdä virheitä, kun tehdään testauksia verkkosivustolle laitteilla, jotka on määritelty toimimaan ideaalisesti latauksen aikana. Todellisuudessa on ymmärrettävä, että käyttäjillä on monta sovellusta ja selainvälilehtiä auki saman aikaisesti. Käyttäjien laitteistot siis tekevät ylimääräistä työtä prosessoimalla monen monta tehtävää kerralla, joka ikinen hetki kuin mahdollista. Kuten esimerkistä voidaan jo päätellä, sovellus ei välttämättä tule toimimaan samalla suorituskyvyllä todellisessa tilanteessa kuin odotettiin. (Shapiro 2015, 120.)

Esimerkki ongelmana käyttöliittymän suorituskyky laskee, kun käytetään get-elementtejä ja set-elementtejä monta kertaa peräkkäin. Tätä ongelmaa kutsutaan layout thrashing -virheeksi. Vaikka verkkoselaimet ovat yleensä varautuneita tästä ongelmasta, niiden täytyy tehdä paljon enemmän töitä, jos on asetettu yksi get-elementti ja sen jälkeen yksi set-elementti peräkkäin. Tämä ongelma liittyy siihen, että verkkoselainten evästeet mitätöityvät set-elementin käytöstä. Tästä ongelmasta selvitään käyttämällä paljon get-elementtejä peräkkäin ja sen jälkeen listataan set-elementit peräkkäin koodin sekaan. Kuvassa 4 esitetään oikeaoppinen tapa listata get- ja set-elementit. (Shapiro 2015, 121–122.)

```
var currentTop = $("element").css("top"); // Get
var currentLeft = $("element").css("left"); // Get
$("element").css("top", currentTop + 1); // Set
$("element").css("left", currentLeft + 1); // Set
```

Kuva 4: Suositeltu get- ja set-elementti järjestys (Shapiro 2015, 122)

Jos taas animaatioluupissa tapahtuu layout thrashing -virhe, niin suorituskyky pahentuu entisestään. Tässä tilanteessa animaatioluuppi yrittää usein saavuttaa 60 kuvaa sekunnissa kuvanopeuden, jossa ihmissilmä saa vaikutelman siitä, että animaatiolla on sujuva liike. Tällä tarkoitetaan, että jokainen kuvanvaihto animaatioluupissa täytyy tapahtua 16,7ms eli 1 sekunti jaettuna 60 kuvalle. Jokaisen askeleen väli pitenee, jos layout thrashing-virhe tapahtuu animaatioluupin aikana. Tämän lopputuloksena animaatio voi mahdollisesti pirstoutua verkkosivustolla. (Shapiro 2015, 121.) Tästä voidaan todeta, että layout thrashing-virhettä suositellaan välttämään tapahtumasta animaatioissa verkkosivustoilla.

## 5 Tiedostojen koko

SVG-kuvat pakkaantuvat todella hyvin. SVG-kuvissa määritelty grafiikka on pienempi tiedostokooltaan verrattuna PNG- tai JPG-tiedostoihin. Tämä parantaa sivuston latausaikoja huomattavasti. (Shapiro 2015, 104.) Tästä esimerkistä voidaan todeta, että tiedoston koolla on väliä verkkosivuston latausaikojen nopeuteen.

Vaikka SVG-kuvilla on etuja pienen tiedostokoon puolesta, se saattaa ylikuormittaa verkkoselaimia antaen niille enemmän tehtäviä samalla kerralla kuin, mihin ne ovat yleensä tottuneet. Tämä tarkoittaa sitä, että animoituja SVG-kuvia ei suositella käytettävän, jos se sisältää liian paljon elementtejä, jonka seurauksena voi olla verkkosivuston hidastuminen. Taulukossa 1 kuvataan SVG-kuvien vahvuudet, joihin kuuluu pieni tiedostokoko, paras mahdollinen tarkkuus ja joustavuus uudelleen editointiin. Toisaalta sen heikkouksiin on listattu verkkoselaimen suorituskyvyn mahdollinen ylikuormittaminen rendaamisessa ja huono yhteensopivuus vanhempien verkkoselainten kanssa. (PlotDB Ltd 2019.)

| Hyvät puolet | Huonot puolet |
|---|---|
| Tiedostokoko | Suorituskyvyn ylikuormittuminen |
| Laatu | Yhteensopivuus: vanhoilla selaimilla |
| Joustavuus editointiin | |

Taulukko 1: SVG-kuvien vahvuudet ja heikkoudet (mukailtu PlotDB Ltd 2019)

GIF-animaatiot ovat yleisesti melko suuria, jos niitä verrataan tiedostokoon perusteella, sillä ne sisältävät useita freimejä yhden tiedoston sisällä (PlotDB Ltd 2019). On kuitenkin ymmärrettävä, kun GIF-animaatio on kyseessä, se suositellaan laittamaan ohjelmointikoodin alkupäähän, jotta käyttäjät välttyisivät odottelulta hitaammilla Internet-yhteyksillä. Tämän takia GIF-animaatiot suositellaan esiladattavaksi verkkosivustolle sekä niistä kannattaa estää niille mahdollisesti annetut ohjausominaisuudet, kunnes ne ovat latautuneet kokonaan. (Schmidt 2001, 190.)

Taulukossa 2 kuvataan GIF-kuvaformaatin vahvuudeksi mm. hyvä yhteensopivuus eri verkkoselainten kanssa. Toisaalta sen heikkouksiin kuuluu huono laatu liikkuvissa kuvissa ja sen tuotoksen tiedostokoko voi olla merkittävästi melko suuri. Tästä voidaan todeta, että GIF-animaatioita suositellaan käyttämään, jos halutaan tukea vanhempia verkkoselaimia. (PlotDB Ltd 2019.)

| Hyvät puolet | Huonot puolet |
| --- | --- |
| Yhteensopivuus: kaikilla selaimilla | Laatu |
| | Tiedostokoko |

Taulukko 2: GIF-kuvaformaatin vahvuudet ja heikkoudet (mukailtu PlotDB Ltd 2019)

GIF-tiedoston kokoa voidaan myös pienentää vähentämällä kuvien värien määrää. Niiden määrä voidaan vähentää jopa 32 väriin menettämällä vain hiukan kuvan laadusta. Tämä voidaan esimeriksi helposti toteuttaa Photoshop-ohjelmalla. (Nicholson, 1998.)

# 6 Case: Fazer Leipomot -animaatio

## 6.1 Tavoite

Fazer Leipomot -animaatio liittyi LAB-ammattikorkeakoululta saatuun ICT-projektikurssiin. Animaatiota työstettiin useammassa kurssissa. Fazer Leipomot -projektissa haluttiin toteutettavan työturvallisuusopetuspeli. Sitä käytettäisiin mahdollisesti henkilöstön perehdyttämiseen työturvallisuusohjeiden kanssa organisaation sisällä. Fazer Leipomot -organisaatio käyttäisi opetuspeliä mm. kesätyötekijöiden perehdyttämiseen ja jo olemassa olevien työntekijöiden olisi mahdollista myös kerrata käytännön ohjeita. Projektissa haluttiin myös työturvallisuuspeliin animaatioita jokaisen kategorian taustalle.

Projektissa saatiin työtehtäväksi keskittyä animaatioiden suunnitteluun ja toteuttamiseen. Ensin perehdyttiin Fazer Leipomot -organisaation antamiin materiaaleihin sekä organisaation antamiin objekteihin animaatiota varten. Materiaalin ohjeiden avulla suunniteltiin käsikirjoitus ja objekteja käytettiin animaation toteuttamisen vaiheessa. Suunnitteluprosessi lähti siitä, minkälaisia näkymiä kussakin työturvallisuuspelin animaatiokategoriassa olisi ja miten käyttäjille voitaisiin näyttää, mitä kussakin kategoriassa pitäisi ottaa huomioon. Jokaisen animaatiokategoria näkymän piti liittyä työturvallisuuskategorian aiheeseen ja siitä piti sitten suunnitella miten animaatio tapahtuisi kussakin animaatiokategoriassa. Projektin alussa mietittiin, minkälaisia animaatioita voisi olla kussakin työturvallisuuskategorian lohkossa ja otettiin huomioon, että animaation täytyi sopia kyseiseen työturvallisuuskategoriaan sekä sen täytyi olla realistinen työturvallisuuteen nähden.

Käsikirjoituksen suunnitteluvaiheessa ideoitiin, minkälaisia vaiheita animaatioissa olisi jokaisen animaatiokategorian kohdalla. Ideoinnissa auttoi työturvallisuusohjeiden lukeminen ja niihin perehdyttäminen. Sen seurauksena saatiin käsitys siitä, miten animaation vaiheet tulisi suunnitella sekä mitä asioita otetaan huomioon suunnittelussa. Suunnitteluvaiheeseen ei kuulunut värimaailma tai grafiikka, sillä materiaali saatiin Fazer Leipomot -organisaatiolta.

Animaation käsikirjoitus suunnitelman valmistuttua otettiin yhteyttä Fazer Leipomot -organisaation vastaavaan ja esitettiin suunnitelma siitä, mitä ideoita saatiin aikaan ja mihin animaatiot keskittyisivät kussakin työturvallisuuskategoriassa. Fazer Leipomot -vastaava hyväksytti käsikirjoituksen sekä lisäksi saatiin ehdotuksia kunkin animaatiokategorian osalta enemmän. Asiakkaan ehdotukset kirjattiin ylös ja niiden pohjalta yritettiin suunnitella ja toteuttaa animaatiota asiakkaan toiveiden mukaisesti.

Seuraavaksi mietittiin, millä ohjelmalla animaatiot toteutettaisiin ja mitä menetelmiä animaation vaiheissa käytettäisiin. Ohjelman valintaa mietittäessä päädyttiin kahteen vaihtoehtoon ja toteutus tapahtuisi joko Adobe After Effect -tai Adobe Animate -sovelluksella. Loppujen

lopuksi valittiin Adobe Animate -ohjelman käyttö animaation toteuttamisessa. Projektissa ei vielä tässä vaiheessa valittu animaation menetelmiä.

Työssä Adobe Animate -ohjelman käyttö ei ollut ennestään tuttua, joten sen opettelu aloitettiin YouTube-videoiden avulla. Opiskelussa keskityttiin siihen, miten kyseistä ohjelmaa käytetään ja miten sillä toteutetaan animaatioita. Opiskelun aikana kirjattiin myös ylös asioita, joita opittiin YouTube-videoita katsellessa. Kirjoitettiin ylös mm. miten työkaluja käytetään kyseisessä ohjelmassa ja minkälaisia pikanäppäin yhdistelmiä sillä voi käyttää. Ohjelman käytön opetteleminen oli ensin hieman vieras ajatus, mutta ymmärrettiin sen käytön idea myöhemmin paremmin, kun sen käytöstä tuli tuttua.

Ohjelman opettelemisen jälkeen seurasi jokaisen animaatiokategorioiden vaiheiden toteuttaminen. Animaation vaiheiden toteuttamisessa pidettiin mielessä, mitä kyseisen animaatiokategorian käsikirjoitukseen oli listattu. Jokaista kategoriaa työstettiin satunnaisessa järjestyksessä ja seurattiin käsikirjoituslistalla olevia suunnitelmia animaatioista. Joskus jonkun kategorian työstämisessä tuli ongelmia tai inspiraation puutetta, ja silloin ratkaistiin ongelma vaihtamalla keskittyminen seuraavaan animaatiokategorian toteuttamiseen. Sitten myöhemmin, kun saatiin taas inspiraatiota, niin jatkettiin työskentelyä edellisestä kategoriasta siitä vaiheesta mihin jäätiin. Projektin missään vaiheessa ei jääty jarruttelemaan tietyn kategorian kanssa vaan yritettiin aina saada tehtyä edes jotain kategoriaa eteenpäin. Kun saatiin joku kategorioista valmiiksi, ja siitä pyydettiin muuttamaan asioita, niin ruvettiin heti työstämään sitä ja animaatiota muutettiin niin kuin pyydettiin.

Animaation aloittamisen jälkeen tuli vastaan ongelmatilanteita aika ajoin, mutta niistä selvittiin loppujen lopuksi hyvin. Animaation ohella Adobe Animate -ohjelman käyttö tuli aina tutummaksi, mitä enemmän sitä käytettiin. Työssä käytettiin myös aika ajoin Adobe Illustrator -ohjelmaa apuna animaation toteuttamisessa. Kyseistä ohjelmaa tarvittiin välillä toteuttamaan SVG-kuvia animaation objekteja varten. Tuotetut SVG-kuvat siirrettiin Adobe Animate -ohjelman puolelle ja sommiteltiin animaatioon tarvittavat objektit valmiille taustalle. Objekteja animoitiin jokaisessa animaation kategoriassa.

## 6.2 Animaatio kategoriat

Käyttäytyminen-kategoriassa kuvattiin tilannetta, jossa esimieshahmo kehuu työntekijähahmoa hyvästä tehdystä työstä. Esimieshahmo nostaa kätensä ja näyttää työntekijähahmolle peukkua merkiksi siitä, että työntekijähahmo on tehnyt työt kuten pitääkin. Työntekijähahmo reagoi tilanteessa näyttäen tyytyväiseltä ja ylpeältä itsestään ilmeensä sekä eleidensä avulla (kuva 5). Esimieshahmon salkkuna kuvattu objekti liikahtelee puolelta toiselle, ja

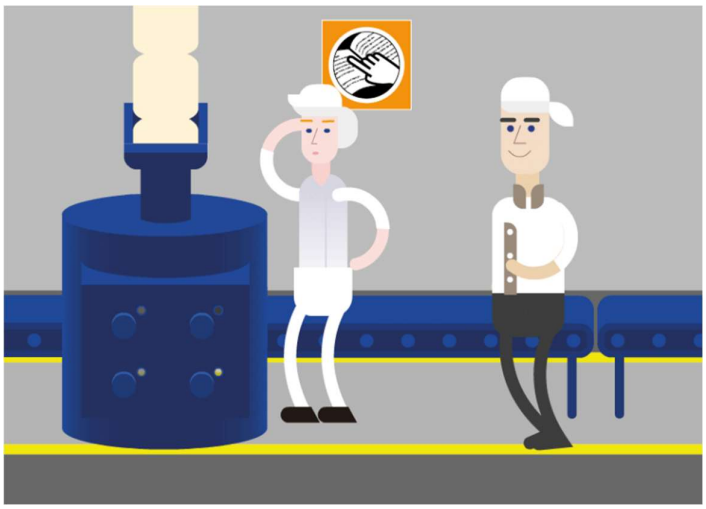
tämä ele tapahtuu, tasaisin väliajoin animaation edetessä. Hahmojen silmät myös animoitiin avautumaan ja sulkeutumaan tasaisessa tahdissa animaation aikana.

Kuva 5: Kuvakaappaus Käyttäytyminen-kategorian animaatiosta

Nostaminen-kategoriassa kuvattiin tilannetta, jossa työntekijähahmot kokoavat laatikoita kasaan oikeaoppisesti (kuva 6). Työntekijähahmot pinoavat kasan päälle laatikot ja poistuvat paikalta. Hahmoille on animoitu liikettä ja jalat liikkuvat kävelyliikkeen tahdissa. Hahmojen silmät avautuvat ja sulkeutuvat animaation aikana sekä hahmoille animoitiin nostoliike laatikon kasaamisvaiheessa. Nostoliikkeessä yritettiin kuvata realistista nostoasentoa, mutta hahmojen jalkojen monimutkainen animointi ei tuottanut tulosta. Loppujen lopuksi päädyttiin siihen, että hahmojen jalat eivät taivu polven kohdalta. Tämä ratkaisu tarkoittaa sitä, että nostoliike ei näytä animaatiossa luonnolliselta eikä oikeaoppiselta työturvallisuus ohjeisiin verrattuna.

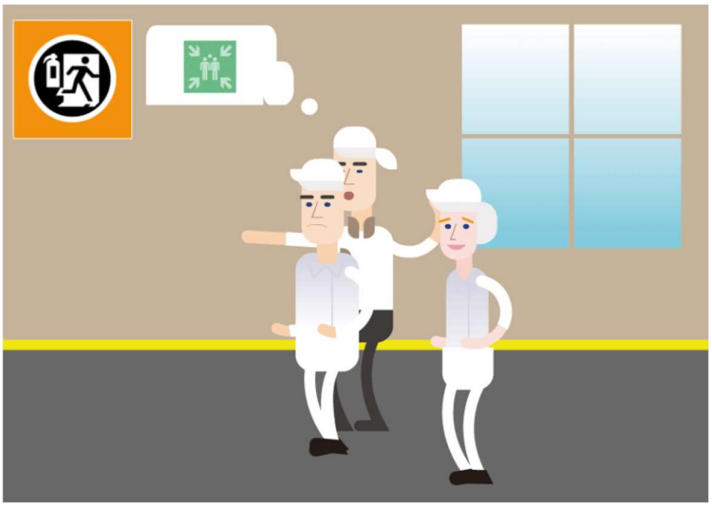
Kuva 6: Kuvakaappaus Nostaminen-kategorian animaatiosta

Ohjeet-kategoriassa kuvattiin tilannetta, jossa työntekijähahmo miettii, kuinka leipätyökone toimii ja pohtii mitä tukkeumatilanteessa pitäisi tehdä, jotta selvitään siitä aiheuttamatta minkäänlaista vahinkoa itselleen tai muille (kuva 7). Tämän animaation leipätyökoneen toiminta animoitiin näkymässä siten, että sen valot animoitiin vilkkumaan tasaisesti ja koneeseen laskeutuu taikinaa. Sen jälkeen kuvataan leipätyökoneen ongelmatilanne valojen epätasaisen vilkkumisen avulla, jossa taikinamaton nopeus hidastuu ja sen seurauksena kone pysähtyy paikalleen. Esimieshahmo ilmestyy näkymään tuon vaiheen jälkeen. Tässä yritettiin kuvata tilannetta, jossa esimieshahmo tulisi työntekijähahmon avuksi ja korjaisi mahdollisen ongelmatilanteen. Työntekijähahmon eleitä myös animoitiin animaation vaiheiden aikana. Kyseisen hahmon raajat animoitiin liikkumaan sekä hahmon silmät yritettiin animoida avautuvat ja sulkeutuvat tasaisesti animaation aikana.



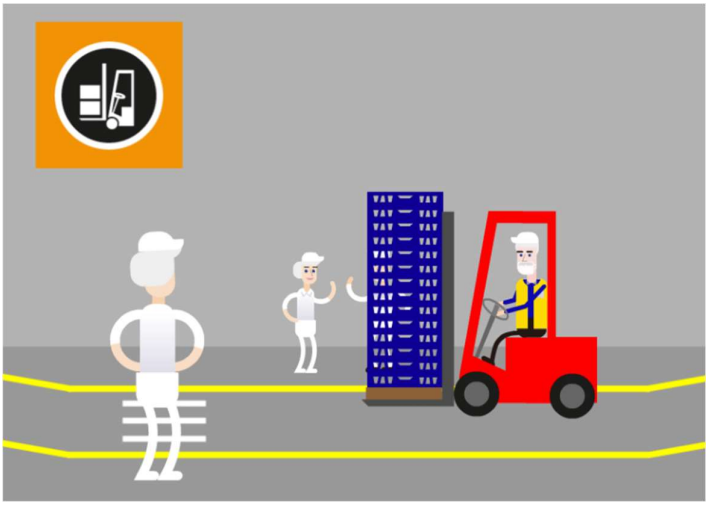
Kuva 7: Kuvakaappaus Ohjeet-kategorian animaatiosta

Poistuminen-kategoriassa kuvattiin tilannetta, jossa hätätilanne on käynnissä ja esimieshahmo opastaa työntekijähahmoja siirtymään kokoontumispaikalle (kuva 8). Kyseiset työntekijähahmot ns. kävelevät näkymän halki. Tämän animaation hahmot animoitiin liikkumaan animaatioruudun oikeasta kulmasta vasempaan kulmaan asti niin, että hahmot ilmestyvät ensin näkymälle ja sitten katoavat näkymältä kokonaan pois. Esimieshahmolle animoitiin liikettä sillä periaatteella, että esimieshahmo näyttäisi antavan neuvoja puheen avulla, joka on kuvattu puhekuplana näkymässä ja samalla esimieshahmo näyttäisi käsimerkein ohjeita työntekijähahmoille niiden liikkuessa ruudun halki. Tässä kategoriassa hahmojen silmät myös animoitiin avautumaan ja sulkeutumaan animaation aikana.



Kuva 8: Kuvakaappaus Poistuminen-kategorian animaatiosta

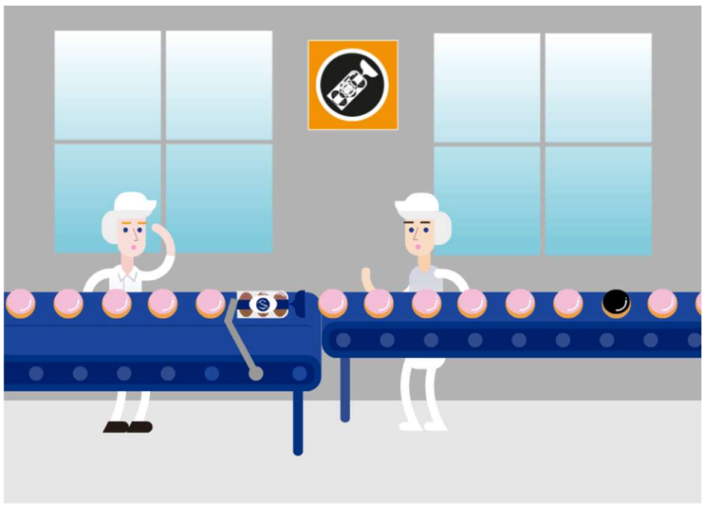
Trukkiliikenne-kategoriassa kuvattiin tilannetta, jossa trukkikone ajaa ohitse ja työntekijähahmot vilkuttavat sille (kuva 9). Tämän animaation trukkikoneobjekti animoitiin liikkumaan ruudun halki näkymään merkityllä reitillä. Trukin päällä on laatikko-objekteja, jotka ikään kuin tärisisivät sen liikkuessa. Kyseiset laatikot animoitiin hieman liikkumaan edestakaisin trukkiobjektin päällä ja ne objektit pistettiin seuraamaan trukkiobjektin liikkeitä. Tässä kategoriassa laitettiin yksi työntekijähahmo tarkkailemaan tapahtuvaa tilannetta sivummalta ja kaikkien työntekijähahmojen silmät animoitiin avautumaan ja sulkeutumaan animaation aikana.



Kuva 9: Kuvakaappaus Trukkiliikenne-kategorian animaatiosta

Tuoteturvallisuus-kategoriassa kuvattiin tilannetta, jossa on vierasesine leipätyökone hihnalla, tässä tilanteessa on leipäpussi, väärä tuote tai jokin muu esine (kuva 10). Tämän animaation työntekijähahmot seuraavat huolestuneiden eleiden avulla tapahtunutta

ongelmatilannetta. Tässä animaatiossa hahmoille animoitiin hämmentyneet ilmeet ja huolestuneet eleet. Lisäksi työkoneen hihna animoitiin liikkumaan ja hihnalle laitettiin liukumaan munkkeja sekä väärä tuoteobjekti niiden sekaan.



Kuva 10: Kuvakaappaus Tuoteturvallisuus-kategorian animaatiosta

Ennakoiva turvallisuus -kategoriassa kuvattiin tilannetta, jossa työntekijähahmo huomaa vesilammikon lattialla ja miettii, että mitä pitäisi tehdä tapahtuneessa ongelmatilanteessa. Myöhemmin toinen työntekijähahmo saapuu paikalle ja näyttää mitä tehdä tilanteessa (kuva 11). Tässä animaatiossa animoitiin työntekijähahmo pohtimaan tapahtunutta tilannetta ja hahmon eleitä animoitiin niin, että hahmo ns. katselee vesilammikkoa edessään. Tässä animaatiossa on myös animoituna ajatuskupla, joka ilmestyy vähitellen esiin hahmon pohtiessa seuraavaa toimintaansa. Lisäksi tässä animaatiossa on toinen hahmo, joka ilmestyy ruudulle sille ajoitetussa kohdassa ja sen hahmon tehtävänä on avustaa ensimmäistä hahmoa tapahtuneessa ongelmatilanteessa. Tässä kategoriassa on myös animoitu hahmojen liikkeitä ja eleitä animaation aikana.

Kuva 11: Kuvakaappaus Ennakoiva turvallisuus -kategorian animaatiosta

Työympäristö-kategoriassa kuvattiin tilannetta, jossa työntekijähahmo huomaa leipäjauhokasan keskellä kulkuväylää ja rupeaa luuttuamaan sitä (kuva 12). Tämän animaation työntekijähahmo animoitiin ns. luuttuamaan jauhokasaa pois näkyvistä sekä hahmon kädessä olevalle harjalle on annettu liikettä, se liikkuu hieman puolelta toiselle vaakatasossa. Tässä animaatiossa huomioitavana yksityiskohtana keskityttiin siihen, että hahmon käsi oli tahdistettu seuraamaan harjan liikettä. Lisäksi tässä kategoriassa hahmon silmät ovat animoitu avautumaan ja sulkeutumaan tasaisesti animaation aikana. Tässä kategoriassa on myös leipäjauhokasa, joka on animoitu katoamaan tietyllä ajankohdalla, kun hahmo on pyyhkinyt sitä jonkin aikaa.

Kuva 12: Kuvakaappaus Työympäristö-kategorian animaatiosta

Liikkuminen-kategoriassa kuvattiin tilannetta, jossa työntekijähahmo kävelee merkittyjen reittien sisällä (kuva 13). Tämän animaation työntekijähahmon kävelyliike on animoitu ja

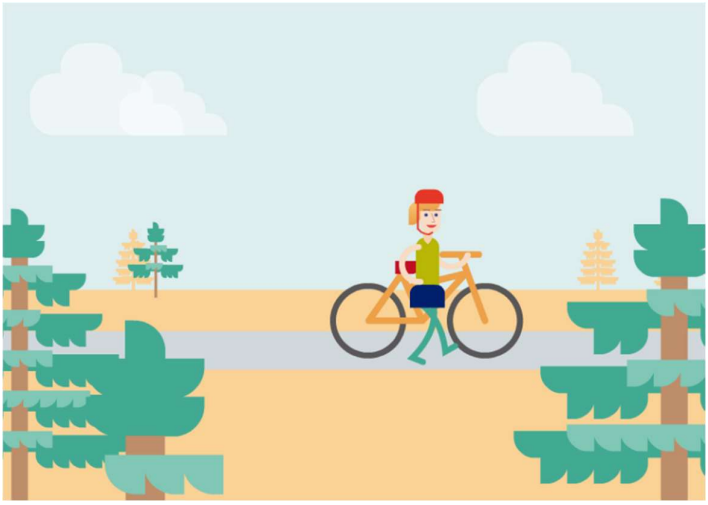
jalat liikkuvat tasaisesti liikkeen tahdissa. Tässä animaatiossa on myös animoituna leivät ja leipäpussit, jotka liikkuvat leipätyökoneen hihnalla tasaisin liikkein.

Kuva 13: Kuvakaappaus Liikkuminen-kategorian animaatiosta

Suojavälineet-kategoriassa kuvattiin tilannetta, jossa työntekijähahmo pukee kuulosuojaimet ja suojalasit ylleen (kuva 14). Tässä kategoriassa kuitenkin päädyttiin loppujen lopuksi siihen, että animoitiin kuulosuojaimet ja suojalasit hyppäämään työntekijähahmon ylle, joka tarkoittaa sitä, että tässä animaatiossa ei huomioitu realistisuutta. Tähän animaatio ratkaisuun päädyttiin siitä syystä, että ei saatu aikaan toista tapaa, jolla sen olisi voinut toteuttaa realistisesti tässä kategoriassa. Lisäksi tässä kategoriassa hahmon yksi käsi liikkuu puolelta toisella animaation aikana.

Kuva 14: Kuvakaappaus Suojavälineet-kategorian animaatiosta

Työhyvinvointi-kategoriassa kuvattiin tilannetta, jossa työntekijähahmo taluttaa polkupyörää pitkin kävelytietä (kuva 15). Tämän kategorian työntekijähahmon ja polkupyörän liike on animoitu ruudun halki ja hahmon jalat liikkuvat tasaisesti animaation aikana. Tässä animaatiossa on myös annettu liikettä pilville ja havukuusille. Lisäksi tästä kategoriasta voidaan todeta, että hahmon käsi heiluu myös tasaisesti animaation aikana. Alkuperäisen suunnitelman mukaan hahmon olisi pitänyt pyöräillä näkymän halki, mutta annettu aika ei riittänyt sen toteuttamiseen projektin aikana.



Kuva 15: Kuvakaappaus Työhyvinvointi-kategorian animaatiosta

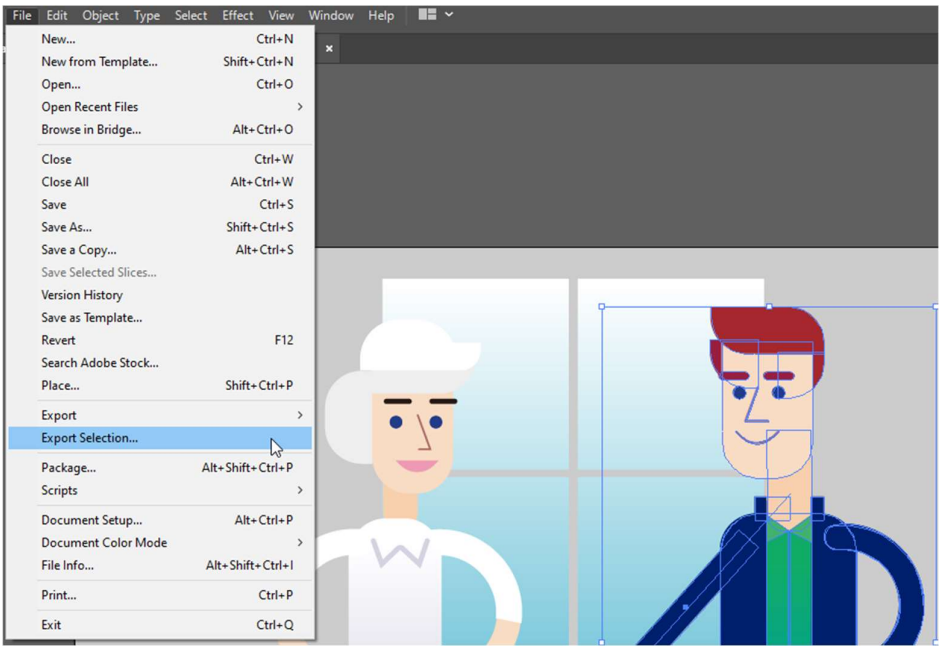
## 6.3 Animaationprosessi

Jokaisen animaatiokategorian -animaatioprosessi lähti samalla lailla käyntiin. Ensin aloitettiin käyttämällä Adobe Illustrator -ohjelmaa ja luotiin tarvittavat objektit SVG-formaatissa. Seuraavaksi käydään läpi, kuinka SVG-kuvat luotiin Adobe Illustrator -ohjelmalla.

SVG-objektin tuottaminen aloitettiin ensin valitsemalla kaikki path-elementit, jotka haluttiin SVG-objektiin ja luotiin niistä Group-elementti eli ryhmä, joka sisältää tässä tilanteessa polkuja. Kuten kuvassa 16 kuvataan, valitaan hahmon osat ja salkuksi kuvailtu objekti ja luodaan siitä ryhmä. Tässä kohtaa todettiin, että tarpeettomat polut lukitaan, jotta niitä ei valittu mukaan luotavaan SVG-objektiin. Esimerkkinä tässä tilanteessa on, että tausta saattoi tulla myös mukaan, kun tarvittavia polkuja valittiin, jonka perusteella päätettiin lukita tarpeettomat polut tiedostosta SVG-objektin luonnin aikana.

Kuva 16: Valitut polut SVG-kuvaa varten

Seuraavaksi valitut polut viedään SVG-kuvana ulos Adobe Illustrator -ohjelmasta. Kuten kuvassa 17 kuvataan, SVG-kuvan luonti prosessi löytyy hiirellä korostetusta valinnasta. Tämän valinnan jälkeen avautuu valintaruutu, jossa päätetään objektin vienti formaatti ja tiedoston tallennuspaikka. Tässä tilanteessa valittiin SVG-kuvaformaatti, josta jatkettiin Adobe Animate -ohjelman puolelle.




Kuva 17: Valittujen polkujen vienti toiseen ohjelmaan

Animaation vaiheiden kulku oli suunniteltu etukäteen ennen animaation aloittamista Adobe Animate -ohjelmassa. Kuten kuvassa 18 kuvataan, animaation vaiheet ensin mietittiin ja suunniteltiin kuvasarjana, jotta animaatioprosessin työskentely vaihe helpottuisi, jolla tarkoitetaan sitä, että ei ollut tarvetta pohtia animaation vaiheita kohta kohdalta. Projektissa nimaatioiden kuvasarja suunnitelmat oli toteutettu Adobe Illustrator -ohjelmalla.

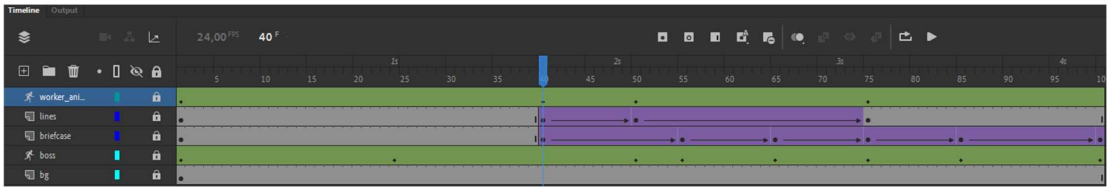
Kuva 18: Kuvasarja animaation vaiheista

Animaation luomisprosessi aloitettiin laittamalla kategorian taustakuva paikalleen animaatio näkymään. Sen jälkeen tuotiin Adobe Illustrator -ohjelmalla luodut SVG-kuvat Adobe Animate -ohjelman puolelle. Tuodut SVG-kuvat sommiteltiin näkymälle animaatiosuunnitelman mukaisesti. Seuraavaksi käydään läpi, miten animaation luonti prosessi oikein toteutettiin.

Esimerkiksi animaation luonti aloitettiin animoimalla hahmojen kädet ja mahdollisesti jalat, jos siihen oli tarvetta tietyn animaatio kategorian kohdalla. Hahmojen kädet animoitiin lisäämällä keyframe-piste aikajanapaneelin alkuun ja sitten ajankohtaa siirrettiin aikajanalla eteenpäin ja liikutettiin hahmon käsi siihen kulmaan, joka oli tarpeellinen tietyssä animaation kohdassa. Hahmojen jalkojen animoimisesta hyvänä esimerkkinä oli Liikkuminen-kategorian animaatio, jossa käytettiin mallikuvaa kävelyliikkeestä. Mallikuvassa oli kuvattu kävelyliike jokaisen kävelyaskeleen kohdalta ja siitä oli helppo työstää kävelyliike hahmolle.

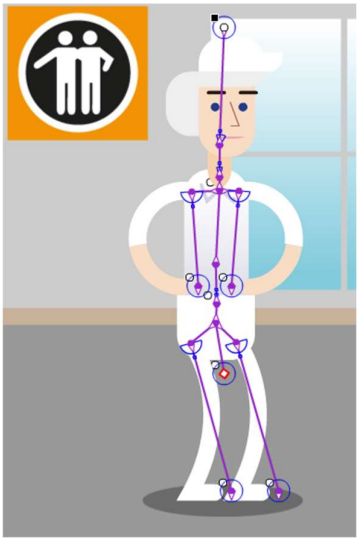
Kävelyliike tuotettiin frame per frame -animointimenetelmällä, jolla tarkoitetaan kirjaimellisesti jokaisen keyframe-pisteen animoimista. Tässä animointi tilanteessa käytiin jokainen keyframe-piste läpi ja siirrettiin hahmon jalkoja tarpeen mukaan oikeaan kohtaan, jotta kävelyliike tuli näyttämään realistiselta. Tämä animointimenetelmä oli kaikista hitain ja työläin animaation tuottamisen aikana ja sitä pohdittiin pitkään ja hartaasti, kuinka se olisi kannattavinta toteuttaa niin realistisesti kuin mahdollista.

Muut animaatiot usein tuotettiin tweening-animointimenetelmällä, jolla tarkoitetaan todellisuudessa sitä, että tietokone laskee keyframe-pisteiden välit ja tuottaa animaation sen mukaisesti. Esimerkiksi Käyttäytyminen-kategoria animaatiossa laitettiin keyframe-piste aikajanalla siihen kohtaan, josta tietyn objektin animointi aloitettiin ja sitten siirrettiin ajankohtaa eteenpäin parin keyframe-pisteen verran ja liikutettiin objektia johonkin muuhun kohtaan alkuperäisestä paikasta. Kuten kuvassa 19 kuvaillaan, tietokone määritteli animaation automaattisesti keyframe-pisteiden välissä.




Kuva 19: Esimerkki tweening-animointimenetelmästä

Animaatioissa on myös käytetty bone-työkalua, joka helpotti käsien ja jalkojen animointiprosessia merkittävästi. Bone-työkalulla voidaan määrittää se kohta, josta objekti kääntyy. Tällä tarkoitetaan sitä, että objektin kiinnekohta (anchor point) on siirretty siihen kohtaan, josta objekti halutaan kääntyvän. Tällä bone-menetelmällä saatiin objekti kääntymään halutulla tavalla animointivaiheessa ja se nopeutti animointiprosessia huomattavasti. Kuten kuvassa 20 kuvataan, bone-työkalun rakenne oli luotu kyseisellä tavalla. Lisäksi työn aikana pohdittiin vaihtoehtoa, jossa olisi pilkottu hahmojen kädet ja jalat osiin. Tämä olisi tarkoittanut sitä, että bone-menetelmällä olisi voitu liikutella hahmojen käsiä ja jalkoja enemmän. Tästä vaihtoehdosta luovuttiin siitä syystä, että animointikokemus ei riittänyt sen toteuttamiseen.



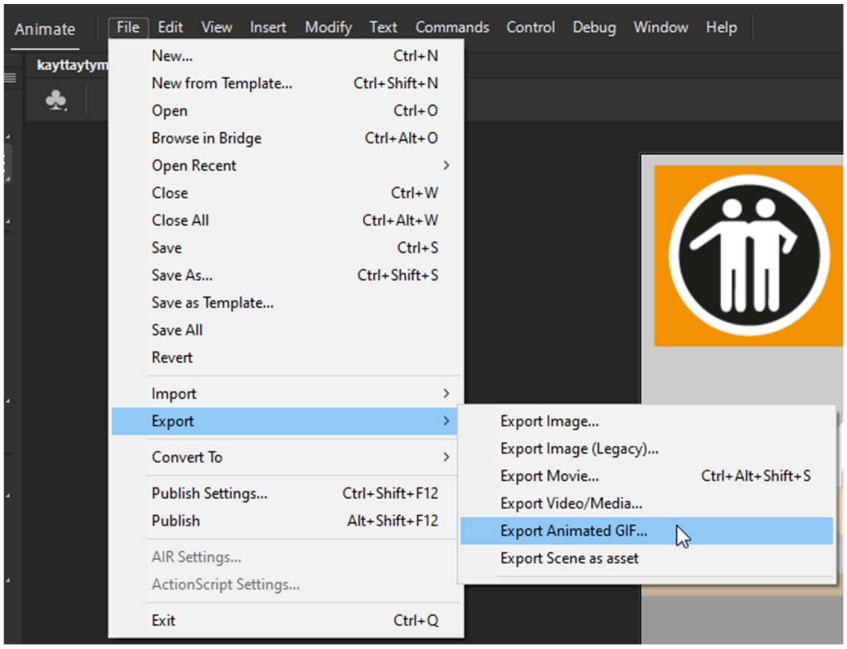
Kuva 20: Bone-työkalun rakenne

Animaatiossa hahmojen yksityiskohtia myös animoitiin. Niiden animointi tapahtui hahmojen kasvo-objektien sisällä. Tällä tarkoitetaan sitä, että hahmon kasvo-objektin sisälle on toteutettu animaatioita. Näitä animaatioita toteutettiin tweening-animointimenetelmällä, sillä se menetelmä oli kaikista loogisin tapa tuottaa mm. silmien ja suun liike. Lopullisessa animaation tuotoksessa suun liike ei kuitenkaan tapahtunut, sillä GIF-animaatio ei jostain syystä ajanut animaatioita objektin sisältä. Lisäksi huomioitavana tekijä todettiin, että silmien

avautuminen ja sulkeutuminen ei ollut tasaista liikettä animaation aikana. Näihin ongelmiin ei löydetty projektin aikana mahdollista ratkaisua, jolla ne olisi voinut korjata.

Projektin animaatio tuotos oli suunniteltu toteutuvan HTML Canvas -animointimenetelmällä, mutta ohjelmoinnin kokemus ei riittänyt Adobe Animate -tuotoksen sisällyttämiseen jo olemassa olevan koodin sekaan. Tästä ongelmatilanteesta seurasi, että Adobe Animate -tuotoksen animaatioformaattia mietittiin jonkin aikaan ja se päädyttiin vaihtamaan GIF-animaatioon loppujen lopuksi. Lopullinen animaatio toteutettiin siis GIF-muodossa.

Projektin animaatiot muutettiin GIF-animaatioiksi Adobe Animate -ohjelman sisällä. Tämä muutos saatiin aikaan tulostamalla animaatiotiedosto GIF-formaatissa. Kuten kuvassa 21 näkyy, valittiin animaatiotiedoston vienti GIF-animaatio muotoon. Tämän toiminnon jälkeen avautui valintaikkuna, jossa määriteltiin GIF-animaation asetukset ja sen jälkeen Adobe Animate -ohjelma tulosti animaatiotuotoksen ulos projektitiedostosta.




Kuva 21: Animaation vienti GIF-animaatio muotoon

Viimeisteltyjä animaatioita testattiin lopullisessa työturvallisuuspelissä. Niiden laadussa ei ollut merkittäviä eroja alkuperäisiin tuotoksiin verrattuna. Ainoana huomionottavana tekijä todettiin, että animaatioiden tiedostokoko oli kasvanut yli 20 kertaiseksi alkuperäiseen Adobe Animate -projektitiedostoon verrattuna. Kuten taulukossa 3 esitetään, tiedostojen koot ovat kasvaneet huomattavasti projektitiedostomuodosta GIF-animaatio muotoon.

| Animaatio | Adobe Animate -projektitiedosto (.fla) | GIF-animaatio (.gif) |
|---|---|---|
| Käyttäytyminen | 95,3 kilotavua | 2,51 megatavua |
| Liikkuminen | 327 kilotavua | 5,86 megatavua |
| Nostaminen | 185 kilotavua | 2,37 megatavua |
| Ohjeet | 294 kilotavua | 6,83 megatavua |
| Poistuminen | 203 kilotavua | 2,56 megatavua |
| Suojavarusteet | 108 kilotavua | 8,44 megatavua |
| Trukkiliikenne | 145 kilotavua | 2,60 megatavua |
| Tuoteturvallisuus | 142 kilotavua | 6,32 megatavua |
| Ennakoiva turvallisuus | 301 kilotavua | 11,7 megatavua |
| Työhyvinvointi | 157 kilotavua | 1,92 megatavua |
| Työympäristö | 218 kilotavua | 6,53 megatavua |

Taulukko 3: Adobe Animate -projektitiedostojen ja GIF-animaatioiden kokoerot

Lopuksi vielä vertailtiin tiedostokokojen mahdollisia eroja riippuen siitä mihin muotoon projektitiedosto julkaistaan. Taulukossa 4 vertaillaan tiedostokokojen eroja keskenään. Tämän vertailun perusteella voidaan päätellä, että GIF-formaatti on suurin kuvaformaatti tiedostokooltaan verrattuna JPG- ja PNG-formaattiin.

| Animaatio | JPG | PNG | GIF |
|---|---|---|---|
| Käyttäytyminen | 37,7 kilotavua | 15,9 kilotavua | 2,51 megatavua |
| Liikkuminen | 62,5 kilotavua | 19,6 kilotavua | 5,86 megatavua |
| Nostaminen | 25,9 kilotavua | 5,60 kilotavua | 2,37 megatavua |
| Ohjeet | 39,2 kilotavua | 14,8 kilotavua | 6,83 megatavua |
| Poistuminen | 25,6 kilotavua | 10,2 kilotavua | 2,56 megatavua |
| Suojavarusteet | 40,3 kilotavua | 13,9 kilotavua | 8,44 megatavua |
| Trukkiliikenne | 30,5 kilotavua | 10,6 kilotavua | 2,60 megatavua |
| Tuoteturvallisuus | 44,1 kilotavua | 14,9 kilotavua | 6,32 megatavua |
| Ennakoiva turvallisuus | 33,3 kilotavua | 15,1 kilotavua | 11,7 megatavua |
| Työhyvinvointi | 31,7 kilotavua | 12,7 kilotavua | 1,92 megatavua |
| Työympäristö | 31,3 kilotavua | 14,3 kilotavua | 6,53 megatavua |

Taulukko 4: Tiedostokokojen vertailu

# 7 Yhteenveto

Tavoitteena oli tutkia ja selvittää web-animaatioiden kehitys vuosien aikana. Työssä saatiin selville, että web-animointitavat ovat muuttuneet hiukan vuosien aikana ja mitä ohjelmistoja tai työkaluja käytetään niiden toteutukseen. Opinnäytetyössä saatiin myös selville, mitä tämän päivän animointitavoista käytetään missäkin tilanteessa ja mitä tapoja käytetään muita enemmän.

Animaatiosovelluksien kohdalla suositellaan käyttämään mm. Adobe After Effect -ja Adobe Animate -ohjelmia animoinnissa. Toisaalta Adobe Photoshop -ja Adobe Illustrator -ohjelmia voidaan käyttää hyödyksi animaation tuottamista varten, mutta niillä tuotetaan enemmän grafiikkaa sen toteuttamiseen. Animaation tekeminen eroaa myös sovelluksien välillä hieman ja niitä käytetään animaation eri vaiheissa. Lisäksi saatiin selville, että kaikki ohjelmat eivät sovi samaan tarkoitukseen ja kuinka tietyt ohjelmat on suunniteltu eri käyttötapoihin ja mitä animointimetodeja ohjelmat käyttävät. Tutkitut ohjelmat myös käyttävät vain tiettyjä animointitapoja animaation tuotoksessa.

Laadun ja suorituskyvyn kohdalla käytiin läpi niihin liittyviä realistisia ongelmatilanteita ja miten niihin saadaan ratkaisu. Otettiin myös esille, kuinka suorituskyky ja laatu laskee mm. get- ja set-elementtien osalta. Lisäksi saatiin selville, että layout thrashing -virhettä suositellaan välttämään.

Tiedostoformaateista saatiin selville, että SVG ja GIF eroavat huomattavasti kokoeron perusteella toisistaan. Tästä todettiin, että tiedostotyyppi vaikutti merkittävästi, ja saatiin selville, että sovelluksen valinta vaikutti huomattavasti web-animaation tuotokseen. Lisäksi niissä havaittiin huomioitavia eroja, ja animaation kokoerot vaikuttavat latausaikoihin verkkoselaimessa.

Case-osuudessa avattiin havaintoja, joita ilmaantui Fazer Leipomot -projektin animaation aikana. Animaatioprojektissa käytettiin Adobe Animate -ohjelmaa ja tuotos toteutettiin GIF-animaatio muodossa. Osuudessa myös huomattiin, että laadulla ja suorituskyvyllä oli vaikutusta animaation tuotokseen. Tiedoston koko ei vaikuttanut lopputulokseen, mutta sen koko oli merkittävän suuri, ja animaatiot toimivat sujuvasti selainpohjaisessa työturvallisuuspelissä. Lisäksi animaation tekoon ei tullut huomioitavia muutoksia, mutta animaation tuotoksen animaatioformaatti muuttui HTML Canvas -koodista GIF-muotoon. Lisäksi animaation työstämisvaiheessa useaan ongelmatilanteeseen saatiin ratkaisu. Ainoa ongelmatilanne, johon ei saatu ratkaisua oli HTML Canvas -koodin ujuttaminen jo olemassa olevan ohjelmointikoodin sekaan.

## Lähteet

Dayley, L. D. & Dayley, B. 2013. Photo CC Bible. John Wiley & Sons. Viitattu 22.4.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=1527433

Eppink, J., Portword-Stacer, L. & Nooney, L. 2014. A Bief History of the GIF (so far). Journal of Visual Culture. Viitattu 11.4.2021. Saatavissa https://journals-sagepub-com.ezproxy.saimia.fi/doi/pdf/10.1177/1470412914553365

Gasston, P. 2014. CSS: A Developers Guide to the Future of Web Design. No Starch Press. Viitattu 15.4.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=1842165

Jerron, S. & AGI Creative Team. 2014. After Effects CC Digital Classroom. John Wiley & Sons. Viitattu 17.4.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=1629165

Maclees, N. 2014. jQuery for Designers Beginner's Guide Second Edition. Birmingham: Packt Publishing. Viitattu 12.5.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=1644004&query=

Makzan. 2011. HTML5 Games Development by Example Beginner's Guide: Create Six Fun Games Using the Latest HTML 5, Canvas, CSS and JavaScript Techiniques. Packt Publishing. Viitattu 20.4.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=948541&query=

MDN Web Docs & Mozilla. 2021. Basic animations – Web APIs | MDN. Viitattu 29.5.2021. Saatavissa https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations

Miltner, K. M. & Highfield T. 2017. Never Gonna GIF You Up: Analyzing the Cultural Significance of the Animated GIF. Social Media + Society. Viitattu 13.4.2021. Saatavissa https://journals.sagepub.com/doi/pdf/10.1177/2056305117725223

Nicholson, S. 1998. GIF versus JPEG: Choosing a graphics compression format for Web publications. American Library Association. Viitattu 23.5.2021. Saatavissa https://www-proquest-com.ezproxy.saimia.fi/docview/215830088/fulltext/219757D86E594997PQ/1?accountid=202350

PlotDB Ltd. 2019. Format for Animations. loading.io. Viitattu 23.5.2021. Saatavissa https://loading.io/articles/format-for-animation/

Refsnes Data. 2021a. CSS Tutorial. W3Schools. Viitattu 10.5.2021. Saatavissa https://www.w3schools.com/css/default.asp

Refsnes Data. 2021b. HTML Canvas. W3Schools. Viitattu 16.5.2021. Saatavissa https://www.w3schools.com/graphics/canvas_intro.asp

Refsnes Data. 2021c. jQuery Tutorial. W3Schools. Viitattu 11.5.2021. Saatavissa https://www.w3schools.com/jquery/default.asp

Refsnes Data. 2021d. Tryit Editor v3.6. W3Schools. Viitattu 30.5.2021. Saatavissa https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_eff_fadeout_fadein

Russell, C. 2019. Adobe Animate CC Classroom in a Book. Adobe Press. Viitattu 28.4.2021. Saatavissa https://books.google.fi/books?id=M1mEDwAAQBAJ&lpg=PT16&ots=wwDMMvXoXX&dq=adobe%20animate%20cc&lr&hl=fi&pg=PT29#v=onepage&q=adobe%20animate%20cc&f=false

Schmidt, C. W. 2001. Presentation accuracy of Web animation methods. Behavior Research Methods, Instruments & Computers. Viitattu 23.5.2021. Saatavissa https://link.springer.com/content/pdf/10.3758/BF03195365.pdf

Shapiro, J. 2015. Web Animation using JavaScript: Develop and Design. USA: Peachpit Press.

Shenoy, A. & Guarini, G. 2013. HTML 5 and CSS3 Transition, Transformation and Animation. Olton: Packt Publishing. Viitattu 15.4.2021. Saatavissa https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/reader.action?docID=1389401&query=