KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Mikko Sairo

TIEDOSTOFORMAATTI KUVILLE JA ÄÄNELLE

Opinnäytetyö
Toukokuu 2014

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2014
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
+358 50 260 6800

Tekijä(t)
Mikko Sairo

Nimeke
Tiedostoformaatti kuville ja äänelle

Toimeksiantaja
Collapick Company Oy

Tiivistelmä

Opinnäytetyössä tavoitteena oli toteuttaa tiedostonmuodon määritelmä toimeksiantona. Tiedostomuodon tuli sisältää kuva- ja äänitiedostoista koostuva esitys pakattuna yhteen tiedostoon sekä metatietoa esityksestä, kuten esityksen otsikko ja kuvien ajastustietoa. Tiedostomuodon määritelmän lisäksi opinnäytetyössä kehitettiin HTML5-sovellus, jolla pystyy luomaan ja esittämään määritelmän mukaisia tiedostoja.

Määritelmän mukaisille tiedostoille asetettiin kriteeriksi, että niiden tuli olla luotavissa ja toistettavissa mobiili- ja tabletohjelmistoalustoilla sekä tietokoneilla, ja että tiedostot veisivät mahdollisimman vähän tilaa, mutta ei suorituskyvyn kustannuksella. Tiedostojen sisältämää metatietoa pitäisi pystyä laajentamaan helposti.

Määritelmää varten opinnäytetyössä tutkittiin ja vertailtiin eri kuva- ja äänitiedostoformaatteja sekä häviöttömiä pakkausmenetelmiä, joilla sidotaan erilliset tiedostot yhdeksi paketiksi. Pakkausmenetelmien syvemmän tuntemuksen saamiseksi opinnäytetyössä tutkittiin myös erilaisia pakkausalgoritmeja.

Äänitiedostoformaatiksi valittiin Ogg Vorbis ja kuvatiedostoformaatiksi JPEG niiden laajan levinneisyyden, ilmaisuuden sekä soveltuvuuden takia. Pakkausmenetelmäksi valittiin Tar tehokkuutensa ja yksinkertaisen rakenteensa vuoksi. Tiedostojen metatieto tallennetaan pakkaukseen omana tiedostonaan käyttäen JSON-notaatiota.

Opinnäytetyön tuloksena syntyi tiedostomuodon määritelmä kuvia ja ääntä sisältäville esityksille, sekä HTML5-sovellus, joka toteuttaa määritelmän vaatimukset.

Kieli
suomi

Sivuja 43
Liitteet 1
Liitesivumäärä 3

Asiasanat
tiedostomuodot, tiedontallennus, äänentallennus, kuvantallennus.

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2014
Degree Programme in
Information Technology
Karjalankatu 3
FI 80200 JOENSUU
FINLAND
+358 50 260 6800

Author(s)
Mikko Sairo

Title
File Format for Images and Audio

Commissioned by
Collapick Company Oy

Abstract

The purpose of this thesis was to develop a file format specification for presentations containing images and audio. The files should contain image and audio files and related metadata like the title of the presentation and timing information, packaged into one file. In addition to the specification, an HTML5 application that can create and present files that conform to the specification would be created during this thesis.

A set of criteria for the specification was given by the client. It should be possible to create and present the files on common mobile, tablet and desktop environments. The files should also take as little space as possible, but not at the cost of performance, and the included metadata should be easily extendable.

Various audio and image file formats and file packaging methods were studied for the specification during the thesis. To get a deeper understanding of the various packaging methods used in the file formats, it was necessary to study data compression algorithms as well.

As a result of the study, Ogg Vorbis was selected as the audio file format and JPEG as the image file format for the specification, due to their features and wide support. Tar was selected as the packaging method due to its superb performance and simple structure. Metadata was chosen to be saved as a separate file in the package using JSON to achieve extensibility.

The result of this thesis was a file format specification for presentations containing images and audio, and an HTML5 application that conforms to the specification.

Language	Pages 43
Finnish	Appendices 1
	Pages of Appendices 3

Keywords
file formats, data compression, audio compression, image compression.

Sisältö

Käsitteet

AES	Advanced Encryption Standard, kehittyneen enkryptauksen standardi, on laajassa käytössä oleva tiedon salaukseen käytetty menetelmä.
BWT	Burrows-Wheeler Transform, kaksisuuntainen muunnosmenetelmä, jolla tieto saadaan muunnettua helpommin pakattavaksi.
EOF	End-of-File, tiedoston loppu, käytetään kuvaamaan merkkiä, josta tiedetään, että tiedosto loppuu.
GIF	Graphic Interchange Format, kuvien häviötön tallennusmuoto.
HTML5	Hypertext Markup Language 5, verkkosivujen luomiseen ja esitykseen käytetyn standardin viides versio. Standardi sisältää JavaScript- ja CSS3-määritykset.
IANA	Internet Assigned Numbers Authority, järjestö, joka on vastuussa muun muassa IP-osoitteista sekä tiedostotyyppien tunnisteiden rekisteristä.
IETF	Internet Engineering Task Force, organisaatio, joka on vastuussa internetiin liittyvien asioiden standardoinnista.
JPEG	Joint Photographic Experts Group, yhdistynyt valokuvauksen eksperttien ryhmä, vastuussa samannimisen tiedostomuodon määritelmästä.
JSON	JavaScript Object Notation, menetelmä, jolla kuvataan rakenteellista tietoa.
Kibitavu	Erotuksena kilotavusta, joka on tuhat tavua SI-järjestelmän mukaan, kibitavu on 1024 tavua, mikä on tietotekniikassa yleisemmin käytetty kerrannaisyksikkö.
LZ77	Abraham Lempelin ja Jacob Zivin mukaan nimetty, heidän vuonna 1977 julkaisema häviötön pakkausalgoritmi.
LZ78	Abraham Lempelin ja Jacob Zivin vuonna 1978 julkaisema paranneltu pakkausalgoritmi.
LZMA	Lempel-Ziv-Markovin ketju -algoritmi, LZ77-algoritmia muistuttava pakkausalgoritmi, joka hyödyntää Markovin ketjua.
LZW	Lempel-Ziv-Welch, Abraham Lempelin, Jacob Zivin ja Terry Welchin vuonna 1984 julkaisema paranneltu versio LZ78-algoritmista.

MP3 MPEG-1 Audio Layer 3, MPEG-standardiin pohjautuva äänitiedostoformaatti.

MPEG Moving Picture Experts Group, ryhmä, jonka tavoitteena on kehittää ja standardoida videon pakkaukseen ja tallennukseen liittyviä teknologioita.

MTF Move-To-Front, menetelmä, jolla pakattava tieto muunnetaan paremmin pakkautuvaan muotoon.

PCM Pulse-code Modulation, pulssikoodimodulaatio, on menetelmä, jolla äänisignaali tallennetaan digitaaliseen muotoon.

PNG Portable Network Graphics, läpinäkyvyyttä tukeva, häviötön kuvien pakkausmenetelmä.

PPMD Prediction by Partial Matching, osittaisen tunnistuksen ennustaminen, menetelmä, jolla tietoa pakataan häviöttömästi perustuen osittaisten yhtäläisyyksien tunnistamiseen jo pakatusta tiedosta.

RAR Roschal Archive, venäläisen Eugene Roshalin kehittämä tiedon pakkausmenetelmä ja -muoto.

RGBA Red-Green-Blue-Alpha, värien esittämismuoto, jossa väristä erotetaan punainen, vihreä, sininen ja läpinäkyvyyskomponentti.

RIFF Resource Interchange File Format, tiedostojen tallennuksessa käytettävä yleinen muoto.

RLE Run-Length Encoding, jakson pituuden koodaus, yksinkertainen tiedonpakkausmenetelmä.

TAR Tape Archive, nauha-arkisto, on yksinkertainen tiedon arkistointimuoto, joka kehitettiin alunperin nauha-asemalle tallentamiseen.

WAV Waveform Audio File Format, aaltoäänitiedostomuoto, on PCM-koodatun äänen tallennusmuoto. Perustuu RIFF-tiedostomuotoon.

WebP Web Picture, verkkokuva, Googlen kehittämä avoin kuvatiedostomuoto, jolla saavutetaan aikaisempia muotoja tehokkaampi pakkaus.

1 Johdanto

Opinnäytetyön tavoitteena oli toteuttaa tiedostomuodon määritelmä sekä referenssitoteutus yritykselle Collapick Company Oy, joka toimi opinnäytetyön toimeksiantajana. Tiedostomuoto kehitettiin pääasiassa mobiili- ja tabletlaitteilla sekä tietokoneilla tehtävien ja esitettävien, kuvia ja ääntä sisältävien esitysten tallentamiseen. Tiedostomuoto on tarkoitettu paitsi esitysten helpompaan tallentamiseen, niin myös helppoon jakoon käyttäjältä toiselle sekä esitysten helppoon toistamiseen. Tiedostomuodon määritelmän lisäksi opinnäytetyön aikana kehitettiin HTML5-sovellus, jolla pystyy luomaan ja toistamaan määritelmän mukaisia tiedostoja. Sovelluksen oli tarkoitus toimia sekä todisteena määritelmän toimivuudesta että lähtöpisteenä jatkokehitykselle.

Erityisen tiedostomuodon tarve syntyi vuoden 2013 kesällä työharjoittelun aikana, kun todettiin, että silloin kehityksessä olleen sovelluksen tapa käsitellä dataa pakkaamattomana ja erillisinä tiedostoina ei soveltunut hyvin mobiili- ja HTML5-alustoilla käytettäväksi ja langattomien verkkojen yli siirrettäväksi. Erillisten kuva- ja äänitiedostojen käsittely ja tallentaminen oli myös työlästä, ja esitysten eheyden varmistus heikkoa. Näissä ympäristöissä toimiakseen data oli ehdottomasti saatava tallennettua pienempään tilaan tiedostojen siirron nopeuttamiseksi sekä myös pakattua yhdeksi tiedostoksi esitysten eheyden varmistamiseksi.

Varsinainen tutkimus ja kehitys tiedostomuotoa varten aloitettiin alkukeväästä 2014 tämän opinnäytetyön muodossa. Aikataulusuunnitelmana oli saada tiedostomuodon määritelmä valmiiksi viimeistään toukokuussa 2014. Määritelmää varten tässä työssä tutkittiin aluksi eri häviöttömiä tiedon pakkaus- ja arkistointimenetelmiä, josta siirryttiin kuvien ja äänen pakkausmenetelmiin. Tutkimuksen aikana toimeksiantajan edustaja määritteli kriteerit, joiden pohjalta valittaisiin käytettävät teknologiat. Teknologioiden tulisi olla ilmaisia käyttää, laajalle levinneitä, lopullisen tiedostokoon tulisi olla pieni, ja tiedostojen tekeminen ja purkaminen tulisi olla vähän suorituskykyä vaativaa.

2 Tiedon pakkaus

Tiedon pakkauksella tarkoitetaan tiedon tallentamiseen tarvittavan tilan pienentämistä ilman, että tallennettu tieto muuttuu merkittävästi tai ero alkuperäiseen tietoon on hyväksyttyjen rajojen sisällä. Tiedon luonne pääsääntöisesti määrää, miten sen voi ja miten se kannattaa pakata.

Kun tietoa pakataan siten, että se on palautettavissa myöhemmin täysin ennalleen, puhutaan häviöttömästä pakkauksesta. Häviötöntä pakkausta voi käyttää periaatteessa minkä tahansa tiedon pakkaamiseen, mutta suurin etu pakkauksesta saadaan esimerkiksi tekstin, kuvien, ohjelmakoodin tai luonnosta kerätyn mittausdatan pakkauksessa, jossa esiintyy paljon toisteisuutta.

Mikäli tiedon pakkauksen aikana sallitaan tiedon osittainen katoaminen, puhutaan häviöllisestä pakkauksesta. Häviöllistä pakkausta käytetään yleisesti esimerkiksi äänen, kuvan ja videon pakkauksessa, koska nykyaikainen laitteisto kykenee toistamaan mediaa ainoastaan rajoitetulla tarkkuudella, ja ihminen kykenee aistimaan toistetun median vielä rajoitetummin. Tällöin on tilan käytön kannalta parempi jättää ylimääräinen tarkkuus tiedosta pois, jolloin saavutetaan parempi pakkaustehokkuus ilman, että tiedon arvo tai havaittu laatu muuttuisi merkittävästi. Häviöllinen pakkaus ei sovellu esimerkiksi tekstin tai mittausdatan pakkaamiseen, koska tällöin mittausdatasta voi jäädä pois merkityksellisiä tietoja tai tieto voi vääristyä, tai tekstistä puuttuu luettavuudelle olennaisia osia.

2.1 Pakkaustehokkuus

Tiedon pakkauksesta puhuttaessa eräs merkittävin vertailukohde on pakkaustehokkuus. Pakkaustehokkuutta merkitään suhdelukuna alkuperäisen tiedon ja pakatun tiedon koon välillä. Joissain tapauksissa pakkaustehoa merkitään myös suhdeluvulla, joka kertoo, kuinka paljon pienempään tilaan pakattu tieto mahtuu.

Häviöttömän pakkauksen tehokkuuteen vaikuttavat sekä pakattava tieto, että käytetty menetelmä. Mitä vaihtelevampaa ja satunnaisempaa pakattava tieto on, sen vaikeampi sitä on pakata häviöttömästi. Ominaisuutta kutsutaan entropiaksi, ja mitä suurempi entropia pakattavalla tiedolla on, sen vaikeampaa sen pakkaus on. Tämä johtuu häviöttömien pakkausmenetelmien luonteesta. Pakkausmenetelmät pyrkivät pääsääntöisesti vähentämään tiedon toisteisuutta, jota on luonnollisesti tuotetussa tiedossa, kuten tekstissä, kuvissa tai mittausdatassa, paljon. Tämän lisäksi menetelmät pyrkivät luomaan ennusteita tulevasta, pakkaamattomasta tiedosta, jolloin jo käsitellyssä tiedossa havaitut säännönmukaisuudet pyritään yleistämään myös tulevaan tietoon. Näiden menetelmien tukena ovat erilaiset muunnokset, kuten Move-to-Front- tai Burrows-Wheeler-muunnos, joiden ensisijainen tavoite on vähentää pakattavan tiedon entropiaa järjestämällä tieto uudestaan siten, että toistuvat elementit ovat vierekkäin ja siten helposti pakattavissa yhteen [1, s. 3; 2].

Häviöllisessä pakkauksessa menetelmien lähtökohdat ovat erilaiset. Koska häviöttömässä pakkauksessa tietoa ei saa kadota lainkaan, joudutaan pakkauksessa tyytymään menetelmiin, jotka ovat suoritettavissa käänteisesti, jolloin tuloksena on alkuperäinen data. Häviöllisessä pakkauksessa tiedon katoaminen pakkauksen edetessä on sallittua, mikä mahdollistaa tehokkaamman pakkauksen. Häviöllistä pakkausta ei kuitenkaan voida käyttää yhtä yleisluontoisesti kuin häviötöntä pakkausta, koska esimerkiksi tekstin tai ohjelmakoodin pakkauksessa ei ole suotavaa, että niistä puuttuu paloja, koska silloin niiden toiminta tai ymmärrettävyys muuttuu. Aistinvaraisemman tiedon, kuten äänen tai kuvien, pakkauksessa tiedon osittainen katoaminen on kuitenkin sallittua, koska ihmissilmä tai -korva ei kykene aistimaan ääntä tai kuvia ehdottoman tarkasti [3]. Tämä on myös häviöllisen pakkauksen pääasiallinen lähtökohta, ja suuri osa häviöllisistä pakkausmenetelmistä pyrkii poistamaan sellaista tietoa, jota ihminen ei pysty aistimaan. Esimerkiksi äänen pakkauksessa ei välttämättä ole tarvetta säilyttää taajuuksia, jotka ylittävät tai alittavat ihmisen kuuloalueen, mikäli ääni on tarkoitettu ihmisen kuunneltavaksi [3]. Tällä tavalla saadaan tietoa pakattua paljon tehokkaammin kuin pelkästään toisteisuutta vähentämällä. Liian tehokas häviöllinen pakkaus kuitenkin jättää pois myös sellaista tietoa, jonka ihminen kykenee aistimaan, ja tällöin myös

lopputuloksen laatu vaikuttaa huonolta. Häviöllisessä pakkauksessa pyritään löytämään tasapaino havaitun laadun ja pakkaustehokkuuden väliltä, ja optimaalinen lopputulos riippuu paljon myös siitä, mihin pakattavaa tietoa on tarkoitus käyttää.

2.2 Häviöttömät pakkausmenetelmät

Tässä luvussa esiteltyjen pakkausmenetelmien tarkoituksena on toimia johdatuksena tiedon pakkaukseen. Menetelmiä tarkastellaan aloittaen yksinkertaisemmasta jakson pituuden koodauksesta ja päätyen lopulta Lempel-Ziv-algoritmeihin, jotka toimivat pohjana nykyaikaisille tiedostojen pakkausformaateille. Osiosta on tarkoituksella jätetty pois Lempel-Ziv-algoritmeista eteenpäin kehitetyt menetelmät, koska niiden tarkempi käsittely oli opinnäytetyön aiheen kannalta epäolennaista. Kehittyneemmistä menetelmistä kerrotaan lisää luvussa kolme, jossa käsitellään eri tiedostojen pakkausformaatteja.

Opinnäytetyöstä on jätetty pois myös häviöllisten menetelmien tarkastelu, koska nämä eivät ole opinnäytetyön aiheen kannalta merkittävässä osassa. Opinnäytetyön kannalta merkittävää ovat eri kuva- ja äänitiedostoformaattien väliset erot eri käyttötarkoituksissa. Tiedostojen häviöllisessä pakkauksessa käytetyt menetelmät ovat yleistason kuvausta lukuunottamatta merkityksettömiä, koska tietyn menetelmän käytön vaikutus ihmisten havaitsemaan laatuun on pieni.

2.2.1 Jakson pituuden koodaus

Jakson pituuden koodaus (run-length encoding, RLE) on häviötön tiedon pakkausmenetelmä, jossa peräkkäiset toistuvat merkkijonot tiivistetään. Peräkkäisistä merkeistä tai merkkijonoista lasketaan toistojen määrä, ja koko ketju korvataan toistojen määrällä sekä yhdellä kappaleella toistettavaa merkkijonoa tai merkkiä. [4.]

Jakson pituuden koodaus soveltuu hyvin sellaisen tiedon tiivistämiseen, jossa on paljon toisteisuutta, kuten esimerkiksi kuvissa. Kuvissa voi olla paljon samanvärisiä kuvapisteitä peräkkäin, jolloin jakson pituuden koodauksella saavutetaan merkittävästi pienempi tiedon määrä. Tiivistysmenetelmä ei sovellu erityisen hyvin esimerkiksi luonnollisen tekstin tiivistykseen vähäisen toisteisuuden takia, ja menetelmän käyttö voi jopa kasvattaa tiedon tarvitsemaa tilaa riippuen siitä, miten paljon toisteisuutta esiintyy ja miten toistot merkitään. [4.]

Yksinkertaisimmassa ratkaisussa jokainen merkintä sisältää toistojen määrän sekä toistuvan merkin. Jos koodattava tieto sisältää yksittäisiä tai toistumattomia merkkejä, koodattu tieto vie tällöin näiden osalta kaksinkertaisen tilan, koska jokaisesta merkistä merkitään, ettei merkki toistu. Jakson pituuden koodauksesta on kuitenkin kehitetty menetelmiä, jotka parantavat tiivistystehokkuutta erilaisilla avainmerkeillä tai jättämällä tiivistämättä merkit, jotka eivät toistu. Tällöin tiivistyksestä ja sen purkamisesta tulee kuitenkin raskaampia. [4.]

2.2.2 Burrows-Wheeler-muunnos

Burrows-Wheeler-muunnosta (Burrow-Wheeler transform, BWT) käytetään tehostamaan varsinaisen pakkausalgoritmin tulosta järjestämällä tieto uudestaan. Muunnoksen voi tehdä molempiin suuntiin, jolloin tietoa ei hävitetä. [1, s. 1-3.]

Muunnos tapahtuu kolmessa vaiheessa. Ensimmäisessä vaiheessa merkkijonoa siirretään yksi merkki kerrallaan eteenpäin ja viimeinen kirjain siirretään ensimmäiseksi. Jokainen iteraatio otetaan talteen. Toisessa vaiheessa iteroidut merkkijonot laitetaan aakkosjärjestykseen, ja näiden merkkijonojen viimeiset kirjaimet ottamalla saadaan muunnoksen lopputulos. [5, s. 5-6.]

Muunnoksen purkaminen tapahtuu siten, että alkuperäisen muunnoksen tulos asetetaan taulukkoon ensimmäiseksi sarakkeeksi, taulukko järjestetään ensimmäisen sarakkeen mukaan aakkosjärjestykseen ja asetetaan muunnoksen tulos uudestaan ensimmäiseksi sarakkeeksi. Tätä toistetaan tuloksen merkkien verran, ja toistojen lopputuloksena saadaan alkuperäisessä muunnoksessa tehty aakkosjärjestykseen asetettu taulu. Näistä iteraatioista alkuperäinen tieto on se, jossa merkkijonon tai muunnetun tiedon lopetusmerkki (end-of-file, EOF) on viimeisenä. [5, s. 7 - 10.] Alkuperäisen arvon saa selville myös tallentamalla sen järjestysluvun järjestetyssä taulukossa muunnosta tehdessä, jolloin alkuperäinen merkkijono löytyy järjestysnumeron osoittamasta kohdasta.

2.2.3 Move-to-Front-muunnos

MTF- eli Move-to-Front-muunnos on tiedon pakkauksessa käytetty menetelmä, jonka tavoitteena on pienentää pakattavan tiedon entropiaa eli vaihtelevuutta ja siten parantaa tiedon pakkautuvuutta. MTF-muunnos itsessään ei välttämättä pienennä pakattavan tiedon tallennukseen tarvittavaa tilaa, mutta sen käyttö tehostaa muita pakkauksessa käytettyjä menetelmiä. [2.]

MTF-muunnoksessa pakattavan tiedon merkit korvataan viittauksilla johonkin ennalta tunnettuun tauluun, kuten esimerkiksi tauluun, joka sisältää kaikki aakkoset järjestyksessä. Jokainen viitattu merkki taulussa siirtää kyseisen merkin taulun ensimmäiseksi, jolloin usein toistuvien merkkien viittaukset saavat pienemmän arvon kuin harvoin toistuvat merkit, jotka löytyvät taulun perältä. Muunnoksen purkaminen tapahtuu lähes samalla tavalla, ottamalla viittausten osoittama merkki alkuperäisestä sanakirjasta ja siirtämällä otettu merkki sanakirjan alkuun. [2.]

2.2.4 Huffmanin koodaus

Huffmanin koodaus (Huffman coding) perustuu usein toistuvien merkkien korvaamiseen lyhyemmillä indekseillä. Huffmanin koodauksessa käytetään apuna laajennettua binääripuuta, jossa usein toistuvat merkit ovat lähempänä puun juurta ja harvinaisemmat merkit ovat syvemmällä puussa, ja usein toistuvien merkkien indekseihin vaaditaan vähemmän bittejä. [6.]

Merkkien esiintymistiheys antaa merkeille painoarvon, jonka mukaan puu järjestetään. Aluksi lähdemateriaalista lasketaan merkkien esiintymistiheys. Järjestäminen aloitetaan esiintymistiheydeltään eli painoarvoltaan kahdesta pienimmästä merkistä, joista muodostetaan binääripuu. Lehtiä eli merkkejä yhdistävän solmun arvoksi tulee kahden merkin painoarvon summa. Tätä toistetaan jäljellä olevilla merkeillä sekä muodostetuilla solmuilla, kunnes kaikki merkit ovat samassa binääripuussa ja puun juuri on löydetty. [6.]

Jokaiselle merkille annettava indeksi muodostuu puun juuresta kyseiseen merkkiin kulkevasta polusta. Jokaisen solmun vasemmalle haaralle annetaan binääriarvo yksi, ja oikealle haaralle nolla. Kuljetusta polusta syntyy täten binäärimuodossa esitetty luku, joka täten vastaa haettua merkkiä. Mitä useammin merkki toistuu, sitä korkeammalla se on puussa ja sitä lyhyempi polku siihen on, ja myös sitä vähemmän tarvitaan bittejä merkin esittämiseen. [6.]

2.2.5 Aritmeettinen koodaus

Aritmeettinen koodaus (Arithmetic coding) on pakkausalgoritmi, jossa merkit esitetään kahden nollasta yhteen ulottuvan alueen reaaliluvun välinä. Lukuväli on yläpäästä avoin, eli lukuvälin yläpäätä rajoittava luku ei kuulu kyseiseen lukuväliin [7, s. 2].

Aritmeettinen koodaus on Huffmanin koodauksen suora kilpailija, sillä molemmat algoritmit pyrkivät vähentämään tiedon esitykseen vaadittavien merkkien määrää merkkien esiintymistiheyden perusteella [7, s. 1].

Aritmeettisessa koodauksessa tarvitaan malli, joka määritellään pakattavassa tiedossa esiintyvien merkkien esiintymistodennäköisyyden perusteella. Kun aritmeettisessa koodauksessa suurin väli on nollasta yhteen, tämä väli jaetaan merkeille niiden esiintymistodennäköisyyden mukaan. Malli voidaan laskea suoraan pakattavasta tiedosta, tai mikäli tiedetään pakattavan tiedon olevan esimerkiksi englanninkielistä tekstiä, voidaan käyttää valmista mallia, joka on optimoitu englanninkielisen tekstin pakkaukseen. [7, s. 2.]

Aritmeettinen koodaus etenee lukemalla lähteestä merkki, katsomalla mallista sen esiintymistodennäköisyys, jakamalla senhetkinen lukuväli merkkien esiintymistodennäköisyyksien mukaan ja valitsemalla uudeksi lukuväliksi se, johon luettu merkki sisältyy. Tätä toistetaan, kunnes kaikki merkit on luettu mukaan, ja tuloksena saadaan lukuväli, josta valitaan yksi reaaliluku esittämään pakattua tietoa. [8.]

```
1 Lukuvälin pituus = lukuvälin yläraja - lukuvälin alaraja
2 Lukuvälin yläraja = lukuvälin alaraja + lukuvälin pituus * merkin lukuvälin yläraja
3 Lukuvälin alaraja = lukuvälin alaraja + lukuvälin pituus * merkin lukuvälin alaraja
```

Kuva 1. Aritmeettisen koodauksen toistettava laskentakaava [8].

Aritmeettisen koodauksen purussa tarvitaan koodauksen tuloksena saatu reaaliluku, koodauksessa käytetty malli sekä koodattujen merkkien määrä. Koodauksen purku tapahtuu vertailemalla lukua mallin määrittelemiin lukuväleihin. Ensimmäinen merkki on se, jonka lukuvälille tulos sijoittuu. Tämän jälkeen kyseisen merkin lukuväli jaetaan mallin mukaan uudestaan eri merkeille ja katsotaan taas, minkä merkin kohdalle tulos sijoittuu. Tätä jatketaan, kunnes riittävä määrä merkkejä on selvitetty. [8.]

2.2.6 Lempel-Ziv-algoritmit

Abraham Lempelin ja Jacob Zivin vuosina 1977 ja 1978 julkaisemat pakkausalgoritmit perustuvat sanakirjapohjaisiin toteutuksiin. Algoritmit käyttävät adaptiivista eli mukautuvaa sanakirjatoteutusta, jossa sanakirjaa täytetään samalla, kun tietoa pakataan. [9, s. 5.] Lempelin ja Zivin vuonna 1977 julkaisema algoritmi LZ77 rakentaa sanakirjan, jossa viittaukset kohdistuvat suoraan pakattavaan tietoon [9, s. 5], kun taas heidän vuonna 1978 julkaisema LZ78 toteuttaa pakattavasta tiedosta erillisen sanakirjan, johon pakatusta tiedosta tehdään viittauksia [9, s. 10].

LZ77 lukee pakattavaa tietoa eteenpäin, ja kun se huomaa vastaavuuden aikaisemmin luettuun tietoon, se korvaa mahdollisimman pitkän vastaavan merkkijonon tiedolla siitä, mistä alkuperäinen havainto alkaa, kuinka pitkä se on ja mikä on ensimmäinen merkki korvatun tiedon jälkeen. Jotta vastaavuuden sijainnin luku pysyy järkevissä mittasuhteissa, LZ77:ssä käytetään niin kutsuttua liukuvaa ikkunaa, joka rajoittaa tarkasteltavan alueen pituutta ja siten korvaavan merkinnän pituus pysyy järkevänä [9, s. 6]. Algoritmi tarkastelee tietoa merkki kerrallaan, etsien vastaavuuksia ikkunan sisällä, ja jos merkin kohdalta ei löydy vastaavuutta, se tulostaa merkinnän, jossa sijainti ja pituus on merkitty nollaksi, ja seuraavaksi merkiksi juuri tarkastellun merkin.

LZ77:lla pakatun tiedon purku perustuu siihen, että pakkauksessa käytetyn ikkunan koko tiedetään, sillä tämä määrittelee myös sen, kuinka pitkiä merkinnät ovat. Pakkauksen purussa dataa luetaan merkintä kerrallaan aloittaen alusta, ja jos merkintä ei viittaa vastaavuuteen (sijainti ja pituus ovat nolla), tulostetaan merkintään liitetty merkki. Muussa tapauksessa liukuvan ikkunan alusta lasketaan vastaavan merkkijonon sijainti, tulostetaan merkinnän pituuden verran merkkejä sekä vastaavuuden jälkeen tuleva merkki, ja siirrytään seuraavaan merkintään.

Toisin kuin LZ77, LZ78 rakentaa erillistä sanakirjaa pakatun tiedon oheen [9, s. 10]. Sanakirjaan tallennetaan viittaus aikaisempaan sanakirjamerkintään, mikäli vastaavuus löytyy, sekä vastaavuutta seuraava merkki. Jos sanakirjasta ei löydy vastaavuutta, tallennetaan sanakirjaan silti uusi merkintä merkistä, ja viittaukseksi annetaan nolla. Ainoastaan sanakirjamerkinnät tulostetaan.

Purettaessa LZ78:lla pakattua tietoa, purkaja saa ainoastaan pakkaajan tulostaman tiedon. Purkaminen tapahtuu samalla tavalla kuin pakkaus, eli purkaja rakentaa syötteen mukaan sanakirjaa ja purkaa sanakirjaviittaukset. [9, s. 12.]

3 Tiedostomuodot

Tässä kappaleessa esitellyt tiedostomuodot ovat kehitettävän tiedostomuodon määritelmän kannalta olennaisia tiedostomuototyyppejä, jakautuen kuva- ja äänitiedostoihin sekä tiedostojen pakkausmuotoihin.Tiedostojen valinnassa käytettiin alakappaleissa esitettyjä valintakriteereitä, joilla tutkittavien tiedostomuotojen määrää saatiin rajattua opinnäytetyön aiheen kannalta järkevään määrään.

3.1 Tiedostojen pakkausmuodot

Tässä opinnäytetyössä tutkittaviksi pakkausmuodoiksi valikoitui nykypäivänä paljon käytettyjä ja hyvin tuettuja formaatteja. Esimerkiksi Zip- ja RAR-tiedostomuodot ovat olleet suosittuja tiedostojen jaossa jo pitkään, ja näiden suosiotaan kasvattava suora kilpailija 7z otettiin myös mukaan ilmaisuutensa sekä tehokkaan pakkauksensa takia.

Vaikka Zip-tiedostomuoto on hyvin vakiintunut pakkausmuoto Windows-ympäristössä, Unix-käyttöjärjestelmien käyttäjien keskuudessa se ei ole yhtä itsestään selvä valinta tiedostojen pakkaukseen. Suurempaa suosiota Unix-

käyttöjärjestelmissä nauttivatkin Tar ja Gzip, jotka ovat täysin ilmaisia formaatteja käyttää. Gzipin ohella vaihtoehtoinen pakkausmenetelmä on Bzip2, joka on myös ilmainen, mutta ei niin paljoa käytetty pakkausmuoto.

Tarkasteltavaksi valitut pakkausmenetelmät ovat kaikki hyvin yleiskäyttöisiä tiedostoformaatteja, joiden pääasiallinen käyttötarkoitus on tiedostojen yhteensitominen ja pakkaus pienempään tilaan. Tarkastelusta jätettiin pois monia pakkausformaatteja, jotka ovat olleet yleisessä käytössä aikaisemmin, mutta joiden tuki on hiipunut uusista tiedonpakkaussovelluksista. Tarkastelusta jätettiin pois myös johonkin erityiseen käyttötarkoitukseen tehdyt pakkausmuodot, kuten Matroska tai Ogg, jotka on tehty pääasiassa videon tai äänen pakkaukseen.

3.1.1 7z

7z on avoimen lähdekoodin GNU Lesser General Public (GNU LGPL) -lisenssiä käyttävä tiedonpakkausmuoto [10], joka julkaistiin ensimmäisen kerran 7-Zip -tiedonpakkausohjelman kanssa. Tiedostomuoto on Igor Pavlovin kehittämä. 7z pystyy arkistoimaan tiedostoja eli sitomaan useita tiedostoja yhteen tiedostojen pakkauksen ohella.

7z käyttää oletuksena LZMA (Lempel-Ziv-Markov) -pakkausalgoritmia, joka on paranneltu versio LZ77-algoritmista, mutta se tukee myös muita pakkausmenetelmiä, kuten LZMA2-, PPMD- (Prediction by Partial Matching -menetelmään pohjautuva algoritmi), BZip2- (Burrows-Wheeler -muunnokseen pohjautuva pakkausmenetelmä) sekä Deflate -algoritmeja. [10.]

Tiedonpakkausmuodon muita ominaisuuksia on tuki 256-bittiselle AES-salaukselle, jota käytetään myös esimerkiksi WLAN-verkkojen liikenteen salaukseen. 7z tukee unicode-merkistöä käyttäviä tiedostonimiä sekä jopa 16 eksatavun (16 miljardin gigatavun) kokoisia tiedostoja. [10.]

7z-tiedostomuodon uutuudesta johtuen se ei ole levinnyt yhtä laajalle kuin esimerkiksi zip. Sen etuna kuitenkin on usein tehokkaampi pakkaus kuin muissa pakkausmuodoissa [10].

3.1.2 Zip

Zip on Phillip Katzin vuonna 1989 kehittämä tiedonpakkausmuoto. Zip on hyvin laajalle levinnyt tiedostomuoto, jota käytetään yhden tai useamman tiedoston pakkaukseen. [11.]

Zip tukee useita eri pakkausalgoritmeja, joista yleisimmin käytetty on Deflate. Tämän lisäksi zip tukee muun muassa LZMA- ja PPMD-algoritmeja sekä ilman pakkausta toteutettavaa tiedostojen arkistointia. [12.]

Zip-tiedostomuoto tukee yksinkertaista, salasanaan pohjautuvaa symmetristä salausta, joka on murrettu useasti ja sen ongelmat ja aukot ovat laajasti tunnettuja. Zipin otsakekentässä on varattu tietoa kolmannen osapuolen tiedoille, joten tiedostomuotoa voi helposti laajentaa uusiin käyttötarkoituksiin. Zip tukee ainoastaan neljän gigatavun kokoisia tiedostoja ja noin 65 000 merkintää sisältäviä arkistoja, mutta zipistä tehty uudistettu versio zip64 poistaa nämä rajoitukset, tukien jopa 16 eksatavun tiedostoja. Zip64-muodon tuki ei kuitenkaan ole niin laaja kuin alkuperäisen zip-muodon. [12.]

3.1.3 RAR

Rar (Roshal Archive) on venäläisen Eugene Roshalin vuonna 1993 julkaisema [13], osittain suljetun lähdekoodin tiedonpakkausmuoto. Tiedostomuodosta on julkistettu ainoastaan rar-tiedostojen purkamiseen tarkoitettu menetelmä. Ainoastaan kaupallisella WinRAR-ohjelmistolla sekä sovelluksilla, jotka ovat saaneet kirjallisen luvan Alexander Roshalilta, Eugene Roshalin veljeltä ja rar-tiedostomuodon tekijänoikeuksien omistajalta, tai jotka on julkaistu häneltä saadun lisenssin alaisena, voidaan luoda rar-tiedostoja. [14.]

Ominaisuuksiltaan rar-pakkausmuoto on verrattavissa avoimen lähdekoodin 7z:iin. Rar tukee suuria, korkeintaan kahdeksan eksatavun kokoisia tiedostoja, unicode-merkistöä, 256-bittistä AES-salausta sekä pakattujen arkistojen pilkkomista useampaan osaan [15]. Pakkausteholtaan rar-muoto ei aivan yllä 7z:n tasolle [16].

3.1.4 Tar

Tar-tiedostomuoto, toisin kuin esimerkiksi zip tai 7z, ei ole varsinaisesti tiedon pakkaamiseen käytetty muoto, vaan sitä käytetään erillisten tiedostojen yhteen sitomiseen. Tar-lyhenne tulee sanoista “tape archive” ja sitä käytettiin alunperin tiedostojen tallentamiseen nauha-asemalle. [17.] Nykyään käyttö on yleistä lähdekoodin paketoinnissa, jolloin käytetään lisäksi vielä gzip-menetelmää tiedon pakkaukseen. Tar- ja gzip-menetelmien yhteiskäytöstä syntyy yleinen .tar.gz-tiedostopääte.

Tar-tiedosto koostuu sarjasta tiedostomerkintöjä, joiden koko on pyöristetty 512 tavun tarkkuuteen. Ylimääräinen tila täytetään nollilla. Tiedostomerkinnän alkuun lisätään tietoa tiedostosta, kuten tiedoston nimi ja sen koko. [17.] Koska Tar-pakkausmenetelmä ei varsinaisesti pakkaa tietoa vaan vain sitoo tiedostoja yhteen, se on hyvin yksinkertainen ja nopea toteuttaa ja käyttää.

3.1.5 Gzip

GZip- eli GNU Zip -tiedostomuotoa käytetään yksittäisten tiedostojen pakkaukseen. Gzip-muotoa käytetään lähes pääsääntöisesti ainoastaan tar-arkistojen kanssa, sillä vaikka gzip kykenee pakkaamaan yhteen liitettyjä tiedostoja, pakkausta purettaessa tiedostoja ei pystytä erottelemaan toisistaan, koska gzip käsittelee yhteen liittyneitä tiedostoja kuten mitä tahansa muuta yksittäistä tiedostoa. Käyttämällä gzip-muotoa tar-arkistoinnin kanssa yhdessä saavutetaan useamman tiedoston tehokas pakkaus. [18.]

Gzip-muoto sisältää otsakkeen, Deflate-menetelmällä pakatun tiedon sekä alatunnisteen, johon kirjoitetaan CRC32-tarkistussumma ja alkuperäisen pakkaamattoman tiedoston koko. Gzipin otsake sisältää gzipin käyttämän ns. maagisen luvun, jolla tunnistetaan tiedoston olevan gzip-muotoinen, tiedostomuodon versionumeron sekä pakkauksen päivämäärän. [19.]

3.1.6 Bzip2

Bzip2, kuten gzip, on pakkausmenetelmä, joka pakkaa yksittäisiä tiedostoja. Tiedostomuodon julkaisi Julian Steward vuonna 1996 käyttäen BSD:n kaltaista lisenssiä. [20.]

Toisin kuin gzip, bzip2 käyttää Burrows-Wheeler- ja Move-to-Front-muunnoksia tiedon järjestelyyn ennen kuin tieto pakataan käyttäen Huffmanin koodausta [20]. Tällä saavutetaan parempi pakkausteho kuin gzipin ja zipin käyttämällä Deflate-menetelmällä, mutta sen heikkoutena on hitaus ja suuri muistin käyttö. [21.]

3.2 Äänen pakkausmuodot

Tässä kappaleessa esitellyt ääniformaatit ovat vain kapea leikkaus kaikista saatavilla ja käytössä olevista formaateista. Aaltoäänimuoto opinnäytetyöhön valittiin kehitettävän tiedostomuodon lähtökohdan pohjalta, sillä aikaisemmin esitysten ääniraita on tallennettu aaltoäänenä.

Valituista teknologioista MP3-formaatti edustaa laajalle levinnyttä ja hyvin yleisesti tunnettua formaattia, jonka laatu on hyvä. MP3:lla on monia kilpailijoita, joista osa on lisensoitavia teknologioita MP3:n tapaan, kuten AAC-formaatti, ja osa täysin ilmaisia, kuten tässä opinnäytetyössä esitelty Ogg Vorbis. Koska MP3 ja AAC ovat hyvin samankaltaisia formaatteja, päädyttiin opinnäytetyöhön

valitsemaan vain toinen, historiallisesti tunnetumpi MP3. Ogg Vorbis valittiin opinnäytetyöhön paitsi sen MP3:a vastaavan laadun, niin myös sen ilmaisuuden takia, joka on tärkeä kriteeri määritelmään valittaville teknologioille.

Opinnäytetyöhön valittiin tutkittavaksi myös uusi Opus-formaatti samoista syistä kuin WebP-kuvaformaatti. Formaatti on uusi ja teknisesti parempi äänen tallennusmuoto, ja sen ollessa ilmainen ja standardoitu oli se mielenkiintoinen lisä muiden valittujen teknologioiden rinnalla.

3.2.1 Aaltoääni

Ääniaaltotiedostomuoto wav (Waveform Audio File Format) on häviötön digitaalisen äänen tallennusmuoto. Tiedostomuoto on IBM:n ja Microsoftin vuonna 1991 julkaisema. Tiedostomuotoa käytetään yleensä raa'an ja pakkaamattoman audion tallentamiseen Windows-ympäristöissä. [22.] Wav-muodon yhteydessä käytetään usein pulssikoodimodulaatiota (Pulse-Code Modulation, PCM) [23] audiodatan koodaamiseksi.

Wav-tiedostoilla on hyvin laaja tuki sen yksinkertaisen rakenteen vuoksi. Koska tiedoston data on pakattu ainoastaan yksinkertaisella häviöttömällä menetelmällä, wav-tiedostot ovat yleensä suuria ja eivät siten sovellu erityisen hyvin äänen varastointiin tai levitykseen. [22.]

Tiedostomuoto pohjautuu Microsoftin ja IBM:n aikaisemmin kehittämään RIFF (Resource Interchange File Format) -määritelmään, jossa määritellään tiedostojen rakenne. RIFF-määritykseen perustuvat tiedot on jaettu paloihin. Jokainen pala, ensimmäinen poislukien, sisältää palaa kuvaavan tunnisteen sekä tiedon siitä, kuinka suuri kyseinen pala on. Tiedoston alussa on nelikirjaiminen tunniste siitä, millainen tiedosto on kyseessä, ja kuinka iso tiedosto on. Tämän jälkeen voidaan lisätä valinnaisia paloja, joissa voi määritellä lisämääreitä tiedostolle, wav-tiedoston tapauksessa esimerkiksi

äänikanavien määrä, mikä on äänen näytteistystaajuus ja kuinka monen bitin pituinen jokainen näyte on. Viimeisenä tiedostossa tulee pala, joka sisältää varsinaisen audiodatan. [23.]

3.2.2 MP3

MP3 (MPEG-1 Audio Layer 3) on MPEG-1 ja MPEG-2 (Moving Pictures Experts Group) -standardiin perustuva audiotiedostomuoto. MP3-muoto julkaistiin vuonna 1993 ja nykyään se on yksi suosituimmista tiedostomuodoista audion levitykseen ja säilöntään. [24.] Sen suurin ongelma vapaassa audion levityksessä on tiedostomuodon maksullisuus. Toisin kuin esimerkiksi Ogg Vorbis tai Opus, MP3:a ei saa käyttää vapaasti, vaan sen käytöstä ohjelmistossa tulee maksaa lisenssimaksuja. [25.] Tästä syystä esimerkiksi useat avoimen lähdekoodin käyttöjärjestelmät, kuten Debian Linux, eivät sisällä oletuksena ohjelmistoja tai ohjelmakirjastoja MP3-tiedostojen käyttöön [26].

Tiedostomuodon suosio johtuu sen hyvästä äänenlaadusta suhteessa tiedostokokoon. Audiodata voidaan pakata murto-osaan alkuperäisen, pakkaamattoman audiodatan koosta ilman, että ihmiskorva havaitsisi merkittävää eroa. [24.]

3.2.3 Ogg Vorbis

Vorbis on vuonna 2000 julkaistu patentiton äänenpakkausmenetelmä [27]. Vorbis ei ole yhtä laajalle levinnyt muoto audiotiedostojen levityksessä kuluttajille kuin esimerkiksi mp3, mutta sitä käytetään laajasti muun muassa peliteollisuudessa, kun halutaan välttää mp3:n aiheuttamia kustannuksia.

Vorbisista ei ole tehty yhtä kattavaa vertailua sen laadusta verrattuna muihin audion pakkausteknologioihin kuten mp3:een, mutta niissä testeissä, joita on tehty, on todettu Vorbisin yltävän parempaan laatuun kuin mp3 pienemmillä

tiedostoilla. Tämä tulos pätee erityisesti alhaisilla kaistanleveyksillä. Siirryttäessä korkeampiin kaistanleveyksiin tiedostomuotojen ero pienenee ja mp3 kuulostaa useammin paremmalta kuin Vorbis. [28.]

Toisin kuin esimerkiksi wav- tai mp3-tiedostoilla, Vorbiksella ei ole omaa tiedostopäätettään vaan Vorbiksella koodattu audiodata pakataan yleisimmin Ogg-tiedostomuotoon [29, s. 5; 29, s. 68]. Tästä yhdistelmästä tulee Vorbiksen yleisemmin tunnettu nimitys Ogg Vorbis. Oggin lisäksi Vorbis-audiota voi pakata myös esimerkiksi Matroska-tiedostoon [30]. Molemmat multimediatiedosto-paketeiksi kutsutut tiedostomuodot voivat sisältää paitsi ääntä, myös videota ja esimerkiksi tekstityksiä.

3.2.4 Opus

Opus on internetin standaroinnista vastaavan IETF:n (Internet Engineering Task Force) kehittämä audion koodausmenetelmä, jonka perimmäinen käyttö-tarkoitus on audiodatan reaaliaikainen välitys internetissä, mutta jota voidaan käyttää myös audion tallennukseen. Opus on täysin avoin audiodatan koodaus-muoto, ja kaikki siihen liittyvät patentit ovat rojaltivapaita. [31.] Opuksen ensimmäinen versio julkaistiin vuonna 2012 [32, s. 1], joten verrattuna muihin, jo 1990-luvulla julkaistuihin koodausmuotoihin, sen levinneisyys ja tuki on rajoittunutta.

Opus-muodon suurin etu verrattuna mp3:een ja Ogg Vorbisiin on sen mahdollisuus muuntautua moniin eri käyttökohteisiin. Opus-koodatulla audiolla on lyhyt toistoviive, jopa murto-osa mp3:een tai Vorbisiin verrattuna. Lyhyt viive on erityisen tärkeä reaaliaikaisessa äänen välityksessä, kuten internetin yli tapahtuvassa puheluissa. Opus käyttää sisäisesti kahta eri koodausmenetel-mää, joiden avulla Opus pystyy optimoimaan sekä puheen että musiikin laadun. Sekä mp3 että Vorbis tukee muuttuvan kaistanleveyden koodausta, mutta Opus vie tekniikan vielä pidemmälle, mahdollistaen pienemmät muutokset kaistan-leveydessä ja tarjoten laajemman mahdollisen kaistanleveyden kuin mp3 tai Vorbis. [31.]

3.3 Kuvan pakkausmuodot

Opinnäytetyössä tutkittaviksi kuvatiedostomuodoiksi valikoituivat yleiset JPEG-, GIF- ja PNG-formaatit niiden laajan levinneisyyden vuoksi. Näiden lisäksi tarkasteluun otettiin mukaan Googlen kehittämä WebP-kuvaformaatti, joka on kehitetty erityisesti kuvien levittämiseen ja esittämiseen internetissä [33].

Tarkasteltavat kuvaformaatit eivät kuitenkaan ole ainoat yleisesti käytössä olevat. Esimerkiksi RAW- ja TIFF-kuvaformaatit ovat valokuvaajien ja media-alan ammattilaisten käytössä niiden sisältämän ylimääräisen tarkkuuden ja muun tiedon takia, mikä mahdollistaa kuvien muokkaamisen ja uudelleen tallentamisen ilman häviötä. Tiedostoja ei kuitenkaan ole tarkoitettu kuvien levitykseen tai esitykseen, mistä kertoo myös niiden suuri tiedostokoko. Suurin osa markkinoilla olevista internet-selaimista ei kykene avaamaan tiedostoja lainkaan. [34.]

RAW:n ja TIFF:n kaltaisten kuvaformaattien lisäksi on olemassa Windows-ympäristöissä paljon käytetty BMP-kuvaformaatti. BMP on häviötön kuva-formaatti, mutta koska se ei sisällä lainkaan pakkausta, tiedostot ovat usein isoja. Sen suosio levitettävänä kuvamuotona onkin heikko, ja nykyiset internet-selaimet ovat jättäneet sen tukemisen pois pienemmän tiedostokoon tarjoavien muiden kuvaformaattien takia. [34.]

3.3.1 GIF

GIF (Graphic Interchange Format) on vuonna 1987 CompuServen esittelemä kuvaformaatti [35, s. 1]. GIF-kuvamuoto käyttää häviötöntä Lempel-Ziv-Welch-tiedonpakkausalgoritmia (LZW) kuvatiedoston koon pienentämiseksi [35, s. 7]. GIF-kuvamuodolla on laaja tuki verkkoselaimissa.

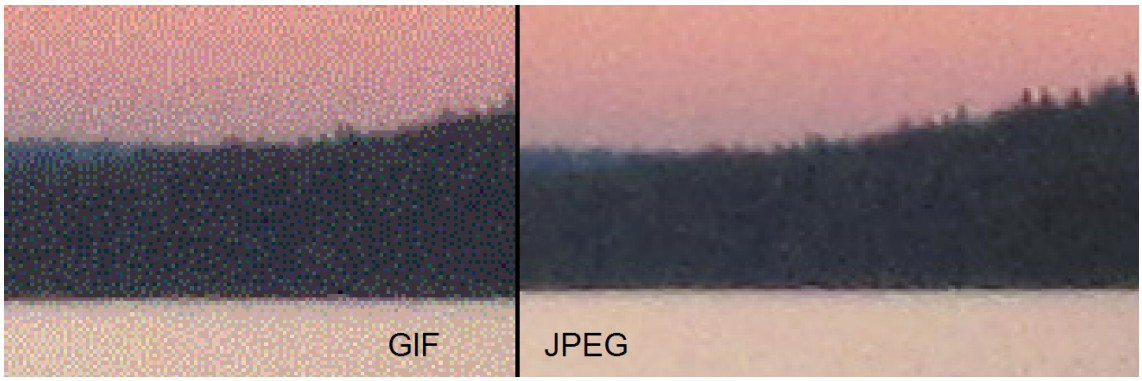
GIF-kuvien erityinen tunnuspiirre on sen tuki animaatioille, jolloin yksi GIF-tiedosto sisältää useamman eri kuvan [35, s. 5]. GIF-muotoiset kuvat tukevat ainoastaan kahdeksan bitin väriavaruutta [35, s. 4], mikä tarkoittaa, että GIF-

kuva pystyy toistamaan ainoastaan 256 eri väriä kerralla. Kuvassa toistuvat värit voidaan kuitenkin valita 24-bittisestä väriavaruudesta [35, s. 5], joten kuvamuoto sopii hyvin yksinkertaisten kuvioiden ja kaavioiden esittämiseen. GIF-tiedostossa jokainen kuva voi sisältää erilaisen väriavaruuden [35, s. 3]. GIF-kuvamuoto tukee myös läpinäkyvyyttä sillä rajoituksella, että kuvapiste voi olla ainoastaan kiinteä tai täysin läpinäkyvä [36, s. 16]. Asteittaista läpinäkyvyyttä tai läpikuultavuutta ei ole.

3.3.2 JPEG

JPEG-kuvamuoto (Joint Photographic Experts Group) on samannimisen ryhmän kehittämä [37, s. 4], häviöllistä pakkausta hyödyntävä kuvamuoto. Erityisesti valokuvaajat suosivat kuvamuotoa, ja sitä käytetään paljon myös kuvien jakamiseen internetissä. Kuvamuodon suosio johtuu pitkälti sen hyvästä kuvanlaadusta valokuvia esitettäessä, sekä pienestä tiedostokoosta.

JPEG:n pakkaus perustuu vierekkäisten, lähes saman väristen kuvapisteiden yhdistämiseen ja ihmissilmän näkemättömien värivaihteluiden poistamiseen, jolloin osa kuvan absoluuttisesta laadusta jätetään pois. Koska ihmissilmä ei ole absoluuttisen tarkka, ei se aina myöskään huomaa puuttuvia yksityiskohtia, mutta pakkaustehoa kasvattamalla kuvan laatu laskee myös siinä määrin, että ihminen kykenee havaitsemaan virheet kuvassa. [38.]

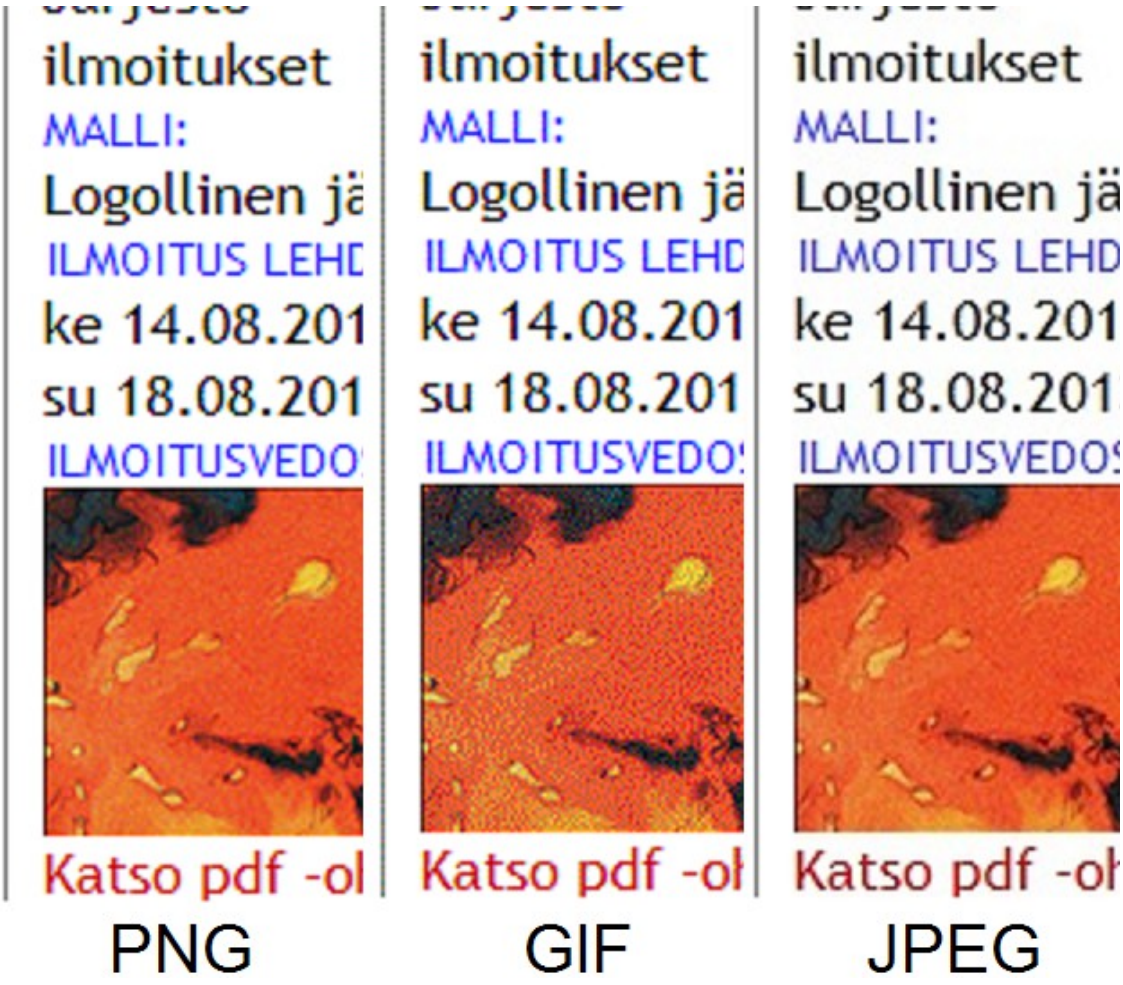


Kuva 2. GIF- ja JPEG-muotojen ero on helposti havaittavissa valokuvista.

3.3.3 PNG

PNG- eli Portable Network Graphics -kuvamuodon kehitys alkoi vuonna 1995, kun huomattiin, että GIF-kuvamuodon käyttämä LZW-algoritmi oli patentointu, ja ettei kuvamuotoa siksi voisi käyttää ilman lisensointia [39]. PNG-kuvamuodon ensimmäinen versio julkaistiin vuonna 1996 [39] ja siitä tehtiin ISO-standardi vuonna 2004 [40].

PNG on häviötön kuvamuoto, joka tukee 24-bittistä RGB-väriavaruutta sekä 32-bittistä RGBA-väriavaruutta, toisin kuin GIF, joka tukee ainoastaan kahdeksan bitin väriavaruutta. GIF-kuvamuotoon verratessa myös läpinäkyvyyden tuki on parantunut, ja PNG-muoto sisältää ns. alfa-kanavan eli mahdollisuuden määritellä asteittainen läpinäkyvyys kuvapisteelle värin lisäksi. [41, s. 4.]



Kuva 3. Tekstiä ja kuvan maalauksesta sisältävä kuva tallennettuna PNG-, GIF- ja JPEG-muodoissa.

Koska PNG on häviötön kuvamuoto, sen tiedostokoko on usein suurempi kuin vastaavan JPEG-muotoisen kuvan. Tästä syystä PNG ei sovellu yhtä hyvin valokuvien esittämiseen. PNG:ssä ei kuitenkaan esiinny samankaltaisia epätarkkuuksia kuin häviöllisesti pakatussa JPEG-kuvassa, joten esimerkiksi tekstiä sisältävät kuvat näyttävät paremmilta PNG-muodossa kuin JPEG:ssä. Jos tekstiä sisältävä kuva ei kuitenkaan sisällä valokuvan kaltaista sisältöä, voi GIF-muotoinen kuva olla tehokkain ratkaisu, joka on myös häviötön kuvamuoto.

3.3.4 WebP

WebP-kuvamuoto (Web Picture) on Googlen kehittämä, erityisesti verkkosivuilla käytettäväksi tarkoitettu formaatti. Toisin kuin PNG- tai JPEG-formaatit, jotka tukevat ainoastaan häviötöntä tai häviöllistä pakkausta, WebP-muodossa voidaan valita, pakataanko kuva häviöllisesti, käyttäen ennustavaa pakkausmenetelmää, vai häviöttömästi, jolloin pakattavat kuvapisteet koodataan perustuen jo pakattuihin kuvapisteisiin sekä pakkauksen ohessa toteutettavaan ja päivitettävään väripalettiin. Kummallakin tavalla pakattuna WebP tukee myös asteittaista läpinäkyvyyttä. [33.]

Lisäetuna JPEG:hen ja PNG:hen verrattuna WebP saavuttaa jopa kolmasosan pienempiä tiedostoja, kun havaittu laatu säilytetään samana. Läpinäkyvyyttä käytettäessä tarvitaan noin viidesosa enemmän tilaa, mutta tällöinkin tiedostojen koko pysyy pienempänä kuin vastaavan PNG-muotoisen kuvan. [33.]

WebP:n heikkous JPEG:hen ja PNG:hen verrattuna on sen heikompi tuki sovelluksissa. Googlen Chrome-selaimessa kuvamuodolle on sisäänrakennettu tuki, mutta esimerkiksi Internet Explorerissa tuki on toteutettu erikseen asennettavana lisäosana. [33.]

4 Tiedostomuodon kriteerit

Opinnäytetyössä kehitettävänä oleva tiedostomuoto on tarkoitettu ensisijaisesti pakkaamaan kuvia ja ääntä yhteen, ja muodostamaan näistä toistettava esitys. Toisin kuin videoesitys, jossa kuva päivittyy tasaisena virtana ja useita kertoja sekunnissa, tavoitellussa esityksessä esitetty kuva ei vaihdu määritellyin väliajoin, vaan määritettynä hetkenä, ollen luonteeltaan verrattavissa nauhoitettuun PowerPoint-esitykseen. Tiedoston tuli sisältää paitsi kuvia ja ääntä, myös tieto siitä, milloin kuvaa tulee vaihtaa.

Tiedostomuodon ominaisuuksia valitessa toimeksiantajan kuvailema pääasiallinen käyttötapaus asetti tavoitteen, johon teknologiavalinnoilla pyrittiin. Määritelmän mukaisia tiedostoja luodaan pääasiassa kameran ja mikrofonin sisältävillä mobiililaitteilla, minkä jälkeen tiedosto jaetaan internet-yhteyden yli muille, esimerkiksi julkiselle palvelimelle katsottavaksi internet-selaimella tai sähköpostin välityksellä suoraan toiselle käyttäjälle.

Käyttötapauksen lisäksi toimeksiantaja on asettanut teknologioille kriteeriksi, että käytetyillä teknologioilla täytyy olla mahdollisimman laaja laite- ja ohjelmistoalustatuki tarkastellessa noin vuoden mittaista ajanjaksoa. Teknologioiden tulisi myös olla ilmaisia, jotta tiedostomuotoa voitaisiin levittää laajan käyttöön ongelmitta. Valitun pakkaustekniikan tulisi tämän lisäksi tukea laajennettavaa metadataa, jolloin tiedostomuodon määrittelyyn kirjattujen metatietojen lisäksi määritelmän mukaisia tiedostoja tekevät ohjelmat voisivat lisätä myös muita metatietoja.

5 Teknologioiden valinta

5.1 Kuvaformaatin valinta

Tiedostomuodolle asetettujen kriteerien ja tehdyn taustatutkimuksen pohjalta tiedostomuodon käyttämän kuvaformaatin valinta on lähes yksiselitteinen. Käytännössä jokainen nykyaikainen digitaalikamera, mukaanlukien

älypuhelimet, käyttävät JPEG-kuvamuotoa otettujen kuvien tallentamiseen sen hyvän soveltuvuuden luonnollisten kuvien tallentamiseen sekä suhteellisesti pienen tiedostokoon takia. Tästä syystä JPEG-formaatti mahdollistaa suoraviivaisemman, tässä työssä määriteltävän tiedostomuodon määritelmän mukaisten tiedostojen luomisen, kun kameran ottamia tiedostoja ei tarvitse muuntaa tiedostomuodosta toiseen.

JPEG:n valintaa puoltavat myös kehitettävälle tiedostomuodolle asetetut kriteerit. Luotavien tiedostojen tulee olla mahdollisimman pieniä ja nopeasti käsiteltäviä, ja JPEG:n häviöllinen pakkaus tuottaa pienempiä kuvia kuin esimerkiksi PNG-muodon häviötön pakkaus. Koska tiedostomuodon pääasiallisessa käyttötapauksessa hyödynnettiin luonnollisia valokuvia, eikä esimerkiksi PowerPoint-tyyppisiä, tekstiä sisältäviä kuvia tai dioja, ei tiedostomuodon ole tarvetta näyttää kuvissa olevaa tekstiä erityisen tarkasti. Myöskään PNG:n tai GIF:n tukemalle läpinäkyvyydelle ei ole tarvetta, joten JPEG on tältäkin osin selkeä valinta.

Mikäli tiedostomuotoon haluaisi todella pienikokoisia kuvia, jopa laadun kärsiessä, GIF olisi ratkaisu tähän skenaarioon. GIF-kuvaformaatin huono soveltuvuus luonnollisten valokuvien esittämiseen rajatun väriavaruuden takia kuitenkin on ristiriidassa ensisijaisen käyttötapauksen kanssa, koska luonnolliset valokuvat tulevat olemaan tiedostomuodon olennainen osa.

Uutena kuvamuotona markkinoille nouseva WebP olisi laadullisesti ja tiedostojen koon perusteella selvä valinta tuetuksi tekniikaksi, sillä suuri osa tähän mennessä esitetyistä perusteista JPEG:n valinnalle pätevät myös WebP:lle. WebP tarjoaisi myös JPEG:tä paremman laadun tai pienemmän tiedostokoon samantasoisella kuvanlaadulla. WebP:n uutuus on kuitenkin myös ongelma. WebP:llä ei ole yhtä laajaa tukea kuin JPEG:llä, jolla on lähes standardinomainen asema kuvia esittävissä ja käsittelevissä sovelluksissa. Tekniikan tuen leviäminen vuoden sisällä laaja-alaisesti hyödynnettäväksi on kyseenalaista, ja JPEG on tässä mielessä turvallisempi valinta. Tulevaisuudessa tämä voi kuitenkin olla merkittävä kehitysidea tiedostomuodolle, sillä sen tekniset edut verrattuna muihin laajasti käytettäviin kuvamuotoihin ovat selvät.

5.2 Ääniformaatin valinta

Kuten kuvien formaatin valinta, on myös äänen tallennukseen käytettävän formaatin valinnassa selkeä, kriteerit täyttävä tiedostomuoto, mutta myös teknisesti parempi tiedostomuoto, jonka levinneisyys ei kuitenkaan yllä vaatimusten tasolle.

Mp3 olisi lähes ehdoton valinta tiedostomuodolle, mikäli sen käyttö olisi ilmaista. Sen sijaan teknisesti ja laadullisesti samalla tasolla oleva Ogg Vorbis on ilmainen, jonka lisäksi sille on vuosien saatossa syntynyt vakaa laitteisto- ja sovellustuki. Ogg Vorbiksen etu mp3:een nähden on sen parempi laatu alhaisemmilla kaistanleveyksillä ja siten pienemmillä tiedostoilla, joka on kriittistä kehitettävän tiedostomuodon käytössä.

Tulevaisuudessa valinnan kanssa kilpailemaan tulee nousemaan Opus, joka on ominaisuuksiltaan parempi aiotussa käytössä. Opuksen puheen ja yleisen äänen tallennukseen optimoitu pakkausmenetelmä sopisi toteutettavaan tiedostomuotoon erittäin hyvin, mutta Opuksen suurin ongelma on sen vielä heikko laite- ja ohjelmistotuki.

Aaltoäänimuodon käyttö kehitettävässä tiedostomuodossa olisi yksinkertaista, sillä se on usein valittu oletustallennusmuodoksi ääntä nauhoitettaessa ja siten erillistä muunnosta muodosta toiseen ei tarvittaisi. Aaltoäänellä on myös laaja tuki ohjelmistoissa ja laitteissa sen yksinkertaisen formaatin takia. Aaltoäänimuodon karsii kuitenkin pois sen hyvin suurikokoiset tiedostot, jotka eivät sovellu hyvin internet-yhteyden yli siirrettäviksi. Aaltoäänimuodon tarjoama hyvä laatu menee hukkaan myös siksi, että mobiililaitteissa olevat mikrofonit ovat laadultaan usein heikkoja.

5.3 Pakkausmuodon valinta

Pakkausmuodon valintaa varten täytyy ottaa huomioon paitsi muotojen lisäominaisuudet, myös lopullisen tiedoston koko, pakkaukseen kulunut aika, pakkauksen aikana käytetty prosessointiteho sekä varattu muisti.

Pakkausmenetelmien tuottamista tuloksista on tehty vertailuja laaja-alaisesti [42; 43], joista pystyisi tekemään johtopäätöksiä, millaisia eroja eri pakkausmenetelmissä on. Tarkimman tuloksen kuitenkin saa vertailemalla tiedostokokoja sekä pakkausprosessissa käytettyjä resursseja, kun pakataan määritelmään valituilla tekniikoilla pakattuja tiedostoja. Tällöin vertailun tulokset ovat suoraan vertailtavissa tilanteisiin, joissa luodaan toteutettavan määritelmän mukaisia tiedostoja.

Pakkausmenetelmien tehokkuuden vertailun lisäksi valinnassa tulee ottaa huomioon menetelmien ominaisuudet ja rajoitukset. Pakkausvertailuun valittavia menetelmiä voidaan karsia jo kyseisten rajoitusten perusteella, jonka lisäksi lopullinen valinta tullaan tekemään menetelmien ominaisuuksien perusteella. Pakkaustehokkuudelle sekä pakkausprosessin aikana varattujen resurssien käytölle annetaan valinnassa yhtäläinen painoarvo muihin ominaisuuksiin verrattuna.

5.3.1 Vertailumenetelmä

Vertailtavia pakkausmenetelmiä verrataan toisiinsa paitsi lopullisen tiedostokoon kautta, niin myös pakkaukseen kuluneen ajan, prosessin varaaman prosessointitehon sekä varatun muistin kautta.

Vertailu suoritetaan Samsung NP900X4D-A02SE -tietokoneella, jossa on 64-bittinen Microsoft Windows 8 -käyttöjärjestelmä. Vertailussa käytetään 7-Zip -pakkausohjelmaa tiedostojen pakkaamiseen sekä pakkaukseen kuluneen ajan laskentaan, sekä käyttöjärjestelmään sisäänrakennettua Performance Monitor -ohjelmaa käytetyn prosessointitehon ja muistin määrän mittaamiseen. Lopullisen tiedoston koko selvitetään käyttöjärjestelmän tiedostoselaimesta. Mittaukset suoritetaan peräkkäin samalla kertaa, jotta vältetään vaihtelut päällä olevissa taustaohjelmissa tai muissa tietokoneen resursseja varaavissa toiminnoissa. Testin aikana pakataan satunnaisesti valittuja Ogg Vorbis- ja JPEG-tiedostoja.

Koska jo häviöllisesti pakattujen kuva- ja äänitiedostojen pakkaus ei oletettavasti merkittävästi pienennä tiedostojen kokoa, tärkeimmäksi tehokkuuskriteeriksi nousee pakkaukseen kuluva aika, sekä pakkauksen aikana varatut resurssit.

5.3.2 Vertailuun valitut pakkausmenetelmät

Koska pakkausmenetelmän tehokkuus on vain yksi valintakriteeri, voidaan vertailtavien menetelmien määrää vähentää jättämällä pois sellaiset menetelmät, jotka eivät täytä muita tiedostomuodolle asetettuja ehtoja.

RAR-pakkausta ei voida sen oletetusta tehokkuudesta huolimatta valita tiedostomuodon pakkausmenetelmäksi, koska tiedostomuotoon valittavien teknologioiden tulee olla lisensoitu ilmaisella lisenssillä. RAR-pakkausmenetelmää lähes vastaava, ilmainen 7z-muoto taas aiheuttaa ongelmia ensisijaisten ohjelmistoalustojen kanssa, sillä sille ei ole toteutettu mm. HTML5-kirjastoa. Verkkoympäristössä 7z-pakkauksia voisi kuitenkin purkaa esimerkiksi Javalla, jolle on käännetty oma kirjasto, sekä PHP:llä, jolla on mahdollista suorittaa komentorivikomentoja palvelimella. PHP-toteutus on kuitenkin tietoturvariski, sillä tällöin verkkosivun kävijälle annetaan epäsuora oikeus suorittaa komentorivikomentoja verkkopalvelimella, ja usein tämänkaltaisen ratkaisun toteutuksen estää verkkopalvelimia tarjoavat tahot. Vaikka Java-sovelluksia on mahdollista suorittaa verkkosivulla, se ei kuitenkaan ole käyttäjälle helppo tai nopea ratkaisu, koska Java vaatii käyttäjältä Java-ohjelmistoa asennettuna tietokoneelleen.

Kun 7z- ja RAR-menetelmät jätetään vertailusta pois, vertailuun jäävät vielä zip-, tar-, gzip- ja bzip2 -menetelmät. Kaikille näille menetelmille löytyy myös JavaScript-ohjelmakirjasto natiivien ohjelmointikielien kirjastojen lisäksi, joten menetelmien käyttö onnistuu riippumatta alustasta. Menetelmät ovat myös ilmaisia ja vapaita käyttää kaupallisessa käytössä, joten mikään lisenssi ei estä niiden käyttöä.

Koska gzip- ja bzip2 -menetelmät eivät kykene sitomaan useita tiedostoja yhteen, käytetään niitä tar-menetelmän kanssa yhdessä, jolloin tiedostot sidotaan tar-menetelmällä ensin yhteen ja tämän jälkeen pakataan kyseisillä menetelmillä.

5.3.3 Vertailun tulokset

Vertailun tuloksista, jotka on esitetty liitteessä 1, voidaan helposti todeta, että suorituskyvyltään tehokkain menetelmä Ogg Vorbiksella pakatun äänen sekä JPEG-kuvien pakkaamiseen yhdeksi tiedostoksi on Tar. Koska Tar ei pyri pienentämään tiedostojen viemää tilaa, vaan sitoo tiedostot vain yhteen ja lisää muuta tietoa tiedostojen oheen, sen ei tarvitse käsitellä tiedostojen sisältöä ja säästää näin resursseja sekä aikaa. Ero vaaditussa suorituskyvyssä tiedostojen pakkaamiseen Tar-menetelmän ja muiden menetelmien välillä on huima. Tar-menetelmää käytettäessä pakkaukseen kului alle sekunti, kun taas esimerkiksi Zip-pakkaus vei viisi sekuntia ja hitain menetelmä Bzip2 tarvitsi jopa 24 sekuntia pakkaukseen.

Menetelmästä riippumatta tiedostojen tilan tarve muuttui alle 1000 kibitavua, mikä on reilun prosentin muutos tilan tarpeessa. Tar-menetelmä oli ainoa, jossa tiedostojen tallentamiseen vaadittavan tilan määrä kasvoi, mutta tällöinkin vain 12 kibitavun verran. Verratessa pakatun tiedoston kokoa pakkaukseen käytettyyn aikaan, Tar on tehokkuudeltaan ehdottomasti paras tiedostojen paketointimenetelmä tiedostomuodolle.

Koska muilta ominaisuuksiltaan Tar on vastaava kuin muut vertaillut menetelmät, eivätkä kehitettävälle tiedostomuodolle asetetut kriteerit sulje pois Tar:n käyttöä, valikoitui se käyttötarkoitukseen parhaiten soveltuvaksi pakkausmenetelmäksi.

6 Tulokset

6.1 Tiedostomuodon määritelmä

Tiedostomuoto pohjautuu aikaisemmissa kappaleissa esiteltyjen teknologioiden ja tehtyjen valintojen pohjalta Tar-pakkaukseen sekä Ogg Vorbis -muotoiseen ääneen ja JPEG-kuviin. Kuva- ja äänitiedostot tulee pakata Tar-pakkauksen määrittelemällä tavalla siten, että äänitiedostot sijaitsevat tiedoston alussa toistojärjestyksessä, ja kuvatiedostot äänitiedostojen jälkeen ensimmäisen esiintymiskerran mukaisessa järjestyksessä. Esityksen ja erillisten kuva- ja äänitiedostojen metatieto tallennetaan pakkauksen alkuun erillisenä tiedostona käyttäen JSON (JavaScript Object Notation) -merkintätapaa. [44.]

Kuvatun kaltaiseen tiedostojen järjestämiseen päädyttiin Tar-pakkauksen sisällön indeksoinnin puutteen takia. Koska Tar-pakkaukseen ei kirjata, mitä tiedostoja pakkaus sisältää tai missä järjestyksessä tiedostot ovat, määrittelemällä metatietotiedoston olevan aina ensimmäinen tiedosto pakkauksessa voidaan pakkauksen lopuista tiedostoista sekä esityksestä saada tietoa ilman, että koko pakkausta on tarve käsitellä ensin.

JSON-notaation käyttöön päädyttiin, koska toimeksiantajalla oli tarve pystyä laajentamaan tallennettavaa metatietoa sovelluskohtaisesti. JSON-notaatio mahdollistaa uusien tietueiden lisäämisen pakettiin ilman, että se hajoittaisi yhteensopivuutta muiden tiedostomuotoa käyttävien ohjelmien kanssa, kunhan määritelmään kirjatut tietueet ovat saatavilla.

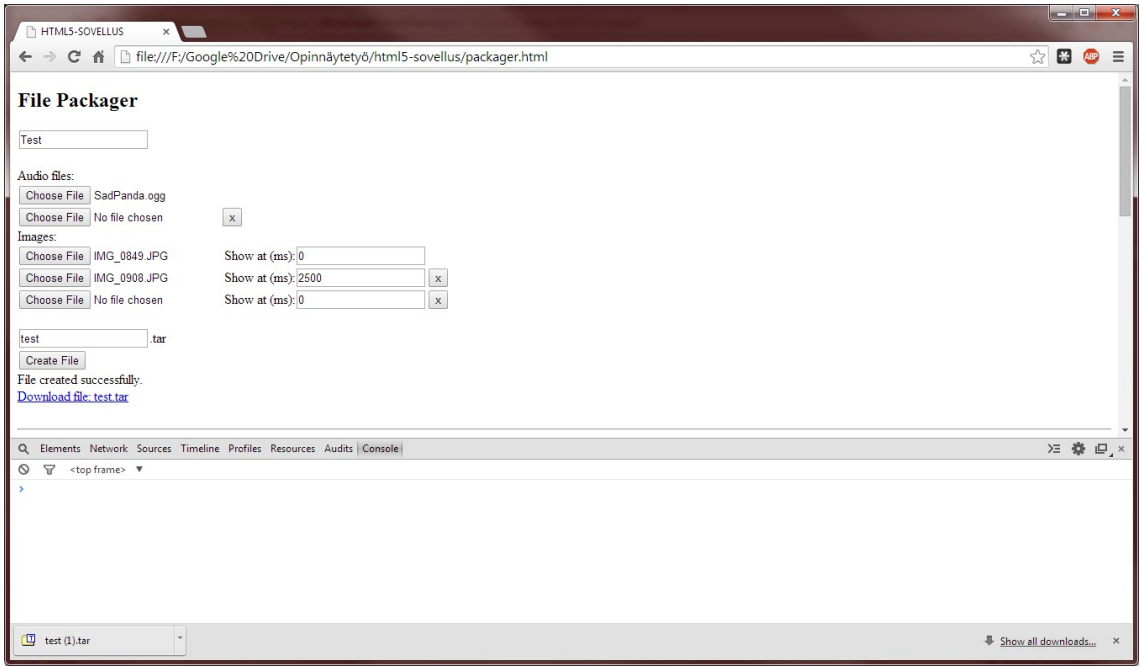
Määritelmään kirjatut metatietotietueet kattavat vähimmäisvaatimukset, jotka tiedostomuodolle on asetettu, sekä jotka vaaditaan, jotta esityksen voisi toistaa onnistuneesti. Tietueet kattavat muun muassa esityksessä olevien tiedostojen määrän, esityksen nimen ja pituuden, yksittäisten äänitiedostojen pituuden ja toistojärjestyksen sekä kuvien toistojärjestyksen ja -ajan. Määritelmä sallii myös saman kuvan käyttämisen useaan kertaan saman esityksen aikana, jolloin riittää, että kuvatiedosto lisätään pakkaukseen kerran. [44.]

6.2 HTML5-sovellus

Opinnäytetyön aikana toteutetun sovelluksen tavoitteena oli toimia sekä todisteena tiedostomääritelmän toimivuudesta että lähtöpisteenä jatkokehitykselle. Sovelluksen toteutuksessa päädyttiin HTML5-ratkaisuun, koska se on nykyisillä mobiili- ja työpöytäympäristöillä hyvin laajalle levinnyt tekniikka. Erityisesti tiedostojen esittämiseen HTML5-toteutus sopii hyvin, koska tällöin tiedostoja pystytään esittämään samoilla laitteilla kuin esimerkiksi Youtuben videoita.

Sovelluksen toteutuksessa käytettiin HTML- ja JavaScript-ohjelmointikieliä. HTML5-standardiin sisältyvistä ominaisuuksista sovelluksessa hyödynnettiin audio-elementtiä [45] äänitiedostojen metatietojen lukemiseen ja tiedostorajapintoja [46] tiedostojen sisältöjen lukemiseen JavaScriptilla. Näiden lisäksi sovelluksessa käytettiin Tar-tiedostojen luomiseen ja purkamiseen Beatgammit-nimimerkin julkaisemaa Tar.js -JavaScript-kirjastoa [47], josta poistettiin riippuvuus Pakmanager-paketinhallintajärjestelmään. Sovellus toimii täysin käyttäjän selaimessa, eikä hyödynnä esimerkiksi PHP:ta sivujen tai toimintojen luomiseen. Sovelluksen käyttöön ei siis tarvita verkkopalvelinta.

Toteutunut sovellus koostuu kahdesta osasta. Ensimmäisenä toteutettu osa on pakkaaja, jonka avulla voidaan luoda käyttäjän valitsemista tiedostoista tiedostomääritelmän mukaisia pakkauksia. Sovellus pitää huolen, että kaikki tiedostomääritelmän vaatimat tiedot ovat saatavilla, ja että pakkaukseen valitut tiedostot ovat määritelmän mukaisia. Käyttäjältä vaaditaan paitsi esityksen tekemiseen käytetyt tiedostot, niin myös esityksen ja lopullisen tiedoston nimi, sekä kuvien vaihtojen ajat. Sovellus pitää huolen siitä, että kuva- ja äänitiedostojen tiedot tallentuvat oikein, ja että esityksen pituus on äänitiedostojen kestojen summa.



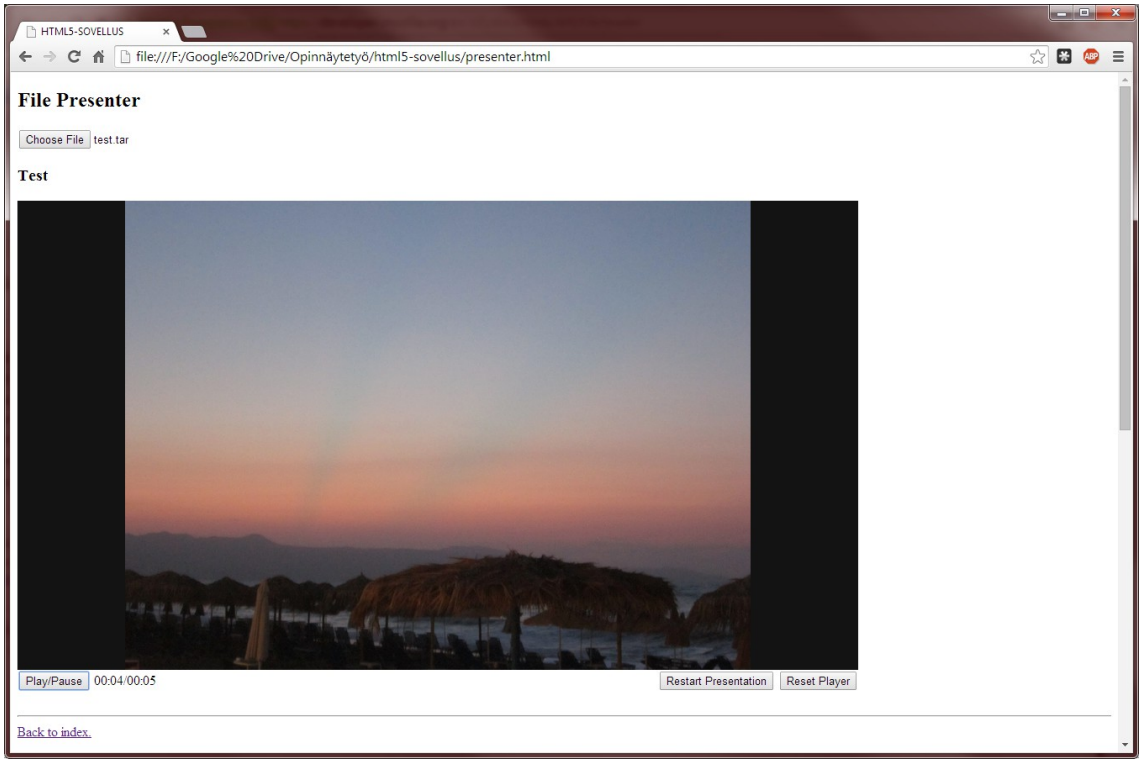
Kuva 4. Tiedostomääritelmän mukaisten tiedostojen pakkaaja.

Pakkaaja ei yksinkertaisen rakenteensa vuoksi toteuta kaikkia tiedostomääritelmän mahdollistamia ominaisuuksia, kuten yhden kuvan käyttämistä esityksessä useaan kertaan, tai ylimääräisen metatiedon lisäämistä tiedostoihin. Sovellus toteuttaa siis kaikki tiedostomääritelmän vaatimukset, mutta ei määritelmän mahdollistamia, erilaisia toteutustapoja ja lisäominaisuuksia.

Sovelluksen toinen osa on tiedostojen esittäjä. Tällä osalla käyttäjä voi katsoa tiedostomääritelmän mukaisia tiedostoja. Tiedostot valitaan käyttäjän tietokoneelta, ja puretaan selaimen muistiin. Esitys toistetaan hyödyntäen JavaScriptin ajastustoimintoja, intervallia (eng. interval) sekä aikakatkaisua (eng. timeout), sekä HTML5-audioelementtiä. Koska audioelementti pystyy toistamaan vain yhden ääniraidan, päätettiin audioelementti piilottaa käyttäjältä täysin, ja toteuttaa ajastus käyttäen aikakatkaisuja kuvien ja äänitiedostojen vaihtamiseen ja esityksen lopettamiseen, sekä intervallia käyttöliittymässä näytettävän ajan päivittämiseen.

Koska sovellus ei vaadi verkkopalvelinta tai internetyhteyttä toimintaansa, sovellusta voi käyttää myös kuten mitä tahansa muuta työpöytäsovellusta. Sovelluksen käyttöä varten ainoa vaatimus onkin nykyaikainen verkkoselain, joka tukee HTML5:n audioelementtiä ja osaa toistaa Ogg Vorbiksella koodattua

ääntä, sekä tukee FileReader-osiota HTML5:n tiedostorajapinnasta. Mozilla Developer Networkin mukaan nämä ehdot täyttyvät työpöytäselaimista ainakin Google Chromen versiolla 7, Mozilla Firefoxin versiolla 3.6, Operan versiolla 12.02 ja Safarin versiolla 6.2. Microsoft Internet Explorerilla sovellus ei toimi, koska siinä ei ole tukea Ogg Vorbis -äänitiedostoille. [46; 48.]



Kuva 5. Tiedostomääritelmän mukaisten esitysten toistin.

7 Pohdinta

Kokonaisuudessaan opinnäytetyö sujui hyvin, ja toimeksiantajana toiminut Collapick Company Oy on ollut tyytyväinen työn tuloksiin. Opinnäytetyön suunnitelmassa esitetty aikatauluarvio ei pitänyt paikkaansa muuten kuin aloituspäivämäärän osalta, mutta työ ei kuitenkaan viivästynyt yli suunnitelmassa asetetun takarajan yli. Suurin syy tähän on tehtävän taustatutkimuksen määrän aliarvioiminen, ja toisaalta liian syvällisen menetelmän valitseminen taustatutkimukseen, mikä aiheutti sen viivästymisen. Kun taustatutkimus oli saatu valmiiksi, toteutusvaihe sujui paljon suunniteltua nopeammin, ja se mahdollisti opinnäytetyön valmistumisen lähes aikataulussa.

Opinnäytetyön tekeminen opetti paljon, paitsi ammatillisesti, niin myös henkilökohtaisesti. Opinnäytetyön tekeminen vaati hyvää tuntemusta monista eri teknologioista, jotka ovat nykypäivänä merkittävissä rooleissa, kun puhutaan tietojenkäsittelystä, tai yleisestikin sähköisen median käytöstä. Jopa niin merkittävässä, että niiden käyttö ja toiminta koetaan itsestäänselvyytenä, ja niiden tarkkaa toimintamallia ei edes ajatella. Tämä oli opinnäytetyön luultavasti merkittävin hidaste, sillä myös dokumentaation laatu on heikko, johtuen mahdollisesti juuri yleisesti käytettyjen menetelmien arkipäiväisyydestä, mikä näkyi opinnäytetyöprosessissa hitaana taustatutkimuksena. Opinnäytetyön aikana oppi paljon paitsi laajasta kirjosta erilaisia tiedon pakkausmenetelmiä, niin myös tiedon hausta ja lähdekriittisyydestä.

Henkilökohtaisesti suurin kasvu opinnäytetyön aikana tapahtui myös taustatutkimuksen aikana. Taustatutkimuksesta olisi voinut tehdä huomattavasti laajemman ja syvällisemmän, mutta aikataulusyistä se jätettiin kevyemmäksi. Opinnäytetyö opetti erityisesti vaikeiden päätösten tekemistä työn laadun ja aikataulun välillä, ja tässä opinnäytetyössä päädyttiin karsimaan taustatutkimuksen laatua, jotta työ saataisiin valmiiksi. Työn aikana tutkittiin pääasiallisesti niitä teknologioita, joille on jo valmiiksi laaja tuki, joten eri teknologioiden skaala oli rajattu. Tällä valinnalla kuitenkin nopeutettiin paitsi tämän

opinnäytetyön toteutusta, niin myös tulevaisuudessa tämän työn pohjalta tehtävää sovelluskehitystä, koska ei ole tarvetta toteuttaa tukea valituille teknologioille tyhjästä.

Opinnäytetyöhön liittyvää taustatutkimusta tehdessä huomattiin selvä ero vanhempien, jo 70- ja 80-luvuilla julkaistujen teknologioiden, sekä uudempien teknologioiden dokumentaation tasossa. Vanhempien teknologioiden ja menetelmien dokumentaatio oli tarkempaa, ja se oli helpommin saatavilla luotettavista lähteistä, kun taas uusien teknologioiden dokumentaatio on siirtynyt yhä enemmän yhteisöjen ja nimimerkillä toimivien henkilöiden ylläpitämäksi, jolloin dokumentaation oikeellisuuden varmistus on vaikeaa. Yhä useammin teknologioiden ainoa saatavilla oleva dokumentaatio löytyy vain Wikipedia-tyyppisiltä sivustoilta, joihin kuka tahansa rekisteröityvä käyttäjä voi tehdä muokkauksia. Vaikka dokumentaation näennäinen laatu olisi hyvä, sen oikeellisuudesta ei voida olla varmoja, koska sen ainoa oikeellisuuden tae perustuu yhteisön luottamukseen, ja yhteisön sisällä siihen, että nimimerkkien takana toimivat henkilöt ovat luotettavia.

Tämän opinnäytetyön aikana toteutettu tiedostomuodon määritelmä, sekä sen toteuttava sovellus, ovat toimeksiantajan asettamien kriteerien mukaisia, mutta silti vasta lähtökuopissa. Opinnäytetyötä varten tiedostomuodosta jätettiin pois sellaiset ominaisuudet, jotka eivät olleet opinnäytetyön tai määritelmän kannalta merkittäviä. Tällaisia ominaisuuksia ovat esimerkiksi tiedostojen salaus ja esityksen kuvien nimeäminen. Tiedostomuotoa ei myöskään ole vielä virallisesti nimetty, ja se on tiedostojen levityksen kannalta suurin ongelma tällä hetkellä. Tulevaisuudessa tiedostomuodolla on kuitenkin mahdollisuud levitä yleiseen käyttöön, ja muutaman vuoden sisällä saattaa olla mahdollista siirtyä käyttämään uusia teknologioita, kuten Opus ja WebP, jotka ovat valittuja teknologioita huomattavasti kehittyneempiä, mutta joilla ei vielä ole riittävän laajaa tukea. Tässä vaiheessa tiedostomuotoon tulee luultavasti liittää tuki usealle vaihtoehtoiselle ääni- ja kuvatiedostomuodolle, jotta tukea saataisiin sekä laajennettua uusiin ympäristöihin, että pidettyä yllä vanhemmille tiedostoille.

Lähteet

1. Langmead, Ben. Introduction to the Burrows-Wheeler Transform and FM Index. Johns Hopkins University. 2013. http://www.cs.jhu.edu/~langmea/resources/bwt_fm.pdf. 29.4.2014.
2. Campos, Arturo. Move to Front. 1999. http://www.arturocampos.com/ac_mtf.html. 29.4.2014.
3. Montgomery, Monty. 24/192 Music Downloads ...and why they make no sense. Xiph.Org. 2012. http://people.xiph.org/~xiphmont/demo/neil-young.html. 7.4.2014.
4. Dipperstein, Michael. Run Length Encoding (RLE) Disucssion and Implementation. 2010. http://michael.dipperstein.com/rle/. 25.3.2014.
5. Corrada, Héctor. Burrows-Wheeler Transform. University of Maryland. 2013. http://www.cbcb.umd.edu/~hcorrada/CMSC423/lectures/exactMatching/bwt.pdf. 29.4.2014.
6. Weisstein, Eric W. Huffman Coding. MathWorld. 2014. http://mathworld.wolfram.com/HuffmanCoding.html. 25.3.2014.
7. Witten, Ian H., Neal, Radford M., Cleary, John G.. Arithmetic Coding for Data Compression. Stanford University. 1987. http://www.stanford.edu/class/ee398a/handouts/papers/WittenACM87ArithmCoding.pdf. 28.3.2014.
8. Dipperstein, Michael. Arithmetic Code Disucssion and Implementation. 2014. http://michael.dipperstein.com/arithmetic/. 29.4.2014.
9. Zeeh, Christina. The Lempel Ziv Algorithm. Cape Institute of Technology. 2003. http://www.itglitz.in/ICT/unit2/LempelZiv.pdf. 25.3.2014.
10. Pavlov, Igor. 7-Zip. 2013. http://www.7-zip.org/. 29.4.2014.
11. Phillip Katz, Computer Software Pioneer, 37. The New York Times. 2000. http://www.nytimes.com/2000/05/01/us/phillip-katz-computer-software-pioneer-37.html. 29.4.2014.
12. PKWare Inc.. ZIP File Format Specification. 2012. http://www.pkware.com/documents/casestudies/APPNOTE.TXT. 29.4.2014.
13. Voloshin, Cyril. Interv'yu po perepiske. 2011. http://www.compression.ru/arctest/descript/roshal.htm. 29.4.2014.

14. Win.rar GmbH. License Agreement. 2014. http://www.buyrar.com/content .asp?Type=License. 29.4.2014.
15. RARLab. WinRAR – What's new in the latest version. 2014. http://www.rar lab.com/rarnew.htm. 29.4.2014.
16. Ramakanth. WinRAR 5 Final Released. Techno360. 2013. http://www.tech no360.in/winrar-5/. 29.4.2014.
17. Kientzle, Tim. Tar(5). FreeBSD. 2011. http://www.freebsd.org/cgi/man.cgi? query=tar&sektion=5&manpath=FreeBSD+8-current. 29.4.2014.
18. Gailly, Jean-loup. Gzip – The data compression program. 1996. http://www.math.utah.edu/docs/info/gzip_5.html#SEC8. 29.4.2014.
19. Deutsch, Peter L.. GZIP file format specification version 4.3. 1996. http://www.gzip.org/zlib/rfc-gzip.html. 29.4.2014.
20. Seward, Julian. 2007. http://www.bzip.org/1.0.5/bzip2-manual-1.0.5.html. 29.4.2014.
21. Hicks, Alan, Lumens, Chris, Cantrell, David, Johnson, Logan. Bzip2. Slackware Linux Inc.. 2005. http://www.slackbook.org/html/archive-files-bzip2.html. 29.4.2014.
22. Meadows, Chris. What Is WAV Format?. Answers. 2014. http://pc.answers. com/file-types/what-is-wav-format. 4.5.2014.
23. Kabal, Peter. Wave File Specifications. McGill University. 2011. http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html. 4.5.2014.
24. Sellars, Paul. Perceptual Coding: How Mp3 Compression Works. Sound On Sound. 2000. http://www.soundonsound.com/sos/may00/articles/ mp3.htm. 4.5.2014.
25. Technicolor. Mp3licensing.com – FAQ. Technicolor. 2009. http://www.mp3licensing.com/help/index.html. 5.4.2014.
26. Sa'ad, Fauzan. Install Non-Free Multimedia Codecs in Debian. 2012. http://www.fandigital.com/2012/08/install-non-free-codecs-in-debian.html. 4.5.2014.
27. Vorbis Audio Compression. Xiph.Org. 2013. https://xiph.org/vorbis/. 4.5.2014.
28. Fearnley, Michael. Ogg Vorbis vs. MP3 – Audio Quality Test at 64kb/s. 2011. www.youtube.com/watch?v=xbw3ItwCrv8. 4.5.2014.

29. Vorbis I specification. Xiph.Org Foundation. 2012. https://xiph.org/vorbis/doc/Vorbis_I_spec.pdf. 4.5.2014.
30. Matroska FAQ. Matroska Non-profit Organisation. 2013. http://www.matroska.org/technical/guides/faq/index.html. 4.5.2014.
31. Opus Interactive Audio Codec. Xiph.Org Foundation. 2013. http://www.opus-codec.org/. 17.3.2014.
32. Valin, Jean-Marc, Vos, Koen, Terriberry, Timothy B.. RFC 6716 – Definition of the Opus Audio Codec. IETF. 2012. http://tools.ietf.org/html/rfc6716. 4.5.2014.
33. WebP – A new image format for the Web. Google. 2014. https://developers.google.com/speed/webp/. 17.3.2014.
34. Hoffman, Billy. Unsuitable Image Formats for Websites. 2012. http://zoompf.com/blog/2012/04/unsuitable-image-formats-for-websites. 12.5.2014.
35. Graphics Interchange Format (GIF) Specification. CompuServe Incorporated. 1987. http://www.w3.org/Graphics/GIF/spec-gif87.txt. 4.5.2014.
36. Graphics Interchange Format, Version 89a. 1990. http://www.w3.org/Graphics/GIF/spec-gif89a.txt. 4.5.2014.
37. Digital Compression And Coding of Continuous-Tone Still Images. ITU. 1993. http://www.w3.org/Graphics/JPEG/itu-t81.pdf. 4.5.2014.
38. Blazé, Kevin. Understanding JPEG Compression. LearningSpark. 2009. http://www.learningspark.com.au/kevin/issues/jpeg_compression.htm. 4.5.2014.
39. Roelofs, Greg. History of the Portable Network Graphics (PNG) Format. 2009. http://www.libpng.org/pub/png/pnghist.html. 4.5.2014.
40. Roelofs, Greg. Portable Network Graphics (PNG) Specification and Extensions. 2011. http://www.libpng.org/pub/png/spec/. 4.5.2014.
41. Boutell, T., et. al.. PNG (Portable Network Graphics) Specification Version 1.0. 1997. https://tools.ietf.org/html/rfc2083. 4.5.2014.
42. Gilchrist, Jeff. ACT Graphic (TIF) Compression Test. 2002. http://compression.ca/act/act-graphics.html. 4.5.2014.
43. Nieminen, Juha. Archiver Comparison. 2004. http://warp.povusers.org/ArchiverComparison/. 4.5.2014.

44. Sairo, Mikko. Tiedostomuodon määritelmä kuville ja äänelle. Collapick Company Oy. 2014.
45. Tomorris. <audio>. Mozilla Developer Network. 2014. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio. 11.5.2014.
46. Scimonster. FileReader. Mozilla Developer Network. 2014. https://developer.mozilla.org/en-US/docs/Web/API/FileReader. 11.5.2014.
47. Beatgammit. Tar-js. 2011. https://github.com/beatgammit/tar-js. 11.5.2014.
48. Silverwind. Media formats supported by the HTML audio and video elements. Mozilla Developer Network. 2014. https://developer.mozilla.org/en-US/docs/HTML/Supported_media_formats. 11.5.2014.

Pakkausmenetelmien vertailun mittaustulokset

Pakattavat tiedostot

Audiotiedosto	**Koko (kibitavua)**	**Pituus**
281 And I Did.ogg	335	00:37
Lähde: http://archive.org/details/281AndIDid		
apples.ogg	1 226	02:16
Lähde: http://archive.org/details/apples		
Avsnitt 1 Helena Roth.ogg	4 473	09:40
Lähde: http://archive.org/details/Avsnitt1HelenaRoth		
BusinessWord.ogg	253	00:25
Lähde: http://archive.org/details/BusinessWord_941		
My beauty will not capture you.ogg	2 131	03:47
Lähde: http://archive.org/details/MyBeautyWillNotCaptureYou		
PrettyLittleBird.ogg	176	00:16
Lähde: http://archive.org/details/PrettyLittleBird_598		
SadPanda.ogg	82	00:05
Lähde: http://archive.org/details/SadPanda		
Secret Life of Josef Mengele.ogg	1 307	01:50
Lähde: http://archive.org/details/SecretLifeOfJosefMengele		
Sett_I_ing_after_The_Day_That_Was.c	241	00:28
Lähde: http://archive.org/details/Sett_I_ing		
SteppingBackAndMovingForward.ogg	1 936	03:36
Lähde: http://archive.org/details/SteppingBackAndMovingForward		
ThePast.ogg	1 982	02:30
Lähde: http://archive.org/details/ThePast		
Up-Close.ogg	306	00:30
Lähde: http://archive.org/details/Up-Close_204		

Kuvatiedosto	**Koko (kibitavua)**	**Koko (px)**	**Lähde**
IMG_0047.JPG	2 810	3240 * 4320	Mikko Sairo
IMG_0072.JPG	2 979	4320 * 3240	Mikko Sairo
IMG_0141.JPG	3 346	4321 * 3240	Mikko Sairo
IMG_0157.JPG	2 548	4322 * 3240	Mikko Sairo
IMG_0206.JPG	2 598	4323 * 3240	Mikko Sairo
IMG_0373.JPG	3 408	3240 * 4320	Mikko Sairo
IMG_0512.JPG	3 632	4320 * 3240	Mikko Sairo
IMG_0513.JPG	3 254	4320 * 3240	Mikko Sairo
IMG_0837.JPG	4 456	4320 * 3240	Mikko Sairo
IMG_0842.JPG	2 402	3240 * 4320	Mikko Sairo
IMG_0849.JPG	2 984	4320 * 3240	Mikko Sairo
IMG_0863.JPG	3 118	4320 * 3240	Mikko Sairo
IMG_0875.JPG	2 608	4320 * 3240	Mikko Sairo
IMG_0888.JPG	3 343	4320 * 3240	Mikko Sairo
IMG_0897.JPG	2 694	4320 * 3240	Mikko Sairo
IMG_0903.JPG	3 111	4320 * 3240	Mikko Sairo
IMG_0908.JPG	2 758	4320 * 3240	Mikko Sairo
IMG_0913.JPG	2 815	4320 * 3240	Mikko Sairo
IMG_0927.JPG	3 518	4320 * 3240	Mikko Sairo
IMG_0962.JPG	3 272	4320 * 3240	Mikko Sairo
IMG_1366.JPG	3 316	4320 * 3240	Mikko Sairo
IMG_20140417_223615.jpg	2 479	4160 * 3120	Mikko Sairo
IMG_20140419_230035.jpg	2 552	4099 * 2823	Mikko Sairo
IMG_20140423_100302.jpg	2 561	4160 * 3120	Mikko Sairo

Pakkausmenetelmien vertailun mittaustulokset

Koko yhteensä (kibitavua): 87 010

Testi 1: ZIP

Pakkausalgoritmi:	DEFLATE		
Sanakirjan koko:	32 kibitavua		
Sanan pituus:	32		
Käytettyjä säikeitä:	1		
	Minimi	**Maksimi**	**Keskiarvo**
Suoritinaika:	1,56%	101,50%	81,92%
Varattu työmuisti:	2 453 504 tavua	9 711 616 tavua	6 776 539 tavua
IO-operaatiota sekunnissa:	0 iops	66,964 iops	47,922 iops
Kulunut aika:	5 sekuntia		
Alkuperäinen tiedostokoko:	87 010 kibitavua		
Pakattu tiedostokoko:	86 497 kibitavua		

Testi 2: TAR

Pakkausalgoritmi:	Store (Ei pakkausta, vain tiedostojen yhteensitominen)		
Käytettyjä säikeitä:	1		
	Minimi	**Maksimi**	**Keskiarvo**
Suoritinaika:	Tulosta ei saatu lyhyen suoritusajan takia.		
Varattu työmuisti:	Tulosta ei saatu lyhyen suoritusajan takia.		
IO-operaatiota sekunnissa:	Tulosta ei saatu lyhyen suoritusajan takia.		
Kulunut aika:	Alle 1 sekunti		
Alkuperäinen tiedostokoko:	87 010 kibitavua		
Pakattu tiedostokoko:	87 022 kibitavua		

Testi 3: TAR + GZIP

Pakkausalgoritmi:	DEFLATE		
Sanakirjan koko:	32 kibitavua		
Sanan pituus:	32		
Käytettyjä säikeitä:	1		
	Minimi	**Maksimi**	**Keskiarvo**
Suoritinaika:	34,35%	99,93%	86,82%
Varattu työmuisti:	3 047 424 tavua	5 496 832 tavua	5 059 925 tavua
IO-operaatiota sekunnissa:	13,992 iops	51,965 iops	42,972 iops
Kulunut aika:	6 sekuntia		
Alkuperäinen tiedostokoko:	87 022 kibitavua		
Pakattu tiedostokoko:	86 496 kibitavua		

Pakkausmenetelmien vertailun mittaustulokset

Testi 4: TAR + BZIP2

Pakkausalgoritmi:	Bzip2		
Sanakirjan koko:	900 kibitavua		
Käytettyjä säikeitä:	1		
	Minimi	**Maksimi**	**Keskiarvo**
Suoritinaika:	1,56%	101,50%	95,83%
Varattu työmuisti:	3 043 328 tavua	13 635 584 tavua	12 828 180 tavua
IO-operaatiota sekunnissa:	6,999 iops	68,954 iops	55,582 iops
Kulunut aika:	24 sekuntia		
Alkuperäinen tiedostokoko:	87 022 kibitavua		
Pakattu tiedostokoko:	86 021 kibitavua		