LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Avoimen lähdekoodin grafiikan- ja videontoistojärjestelmä

Case: CasparCG:n käyttöönotto ja tuotteistaminen pH kolme Oy:lle

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tieto- ja viestintätekniikan
koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2016
Ismo Vaittinen

Lahden ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutusohjelma

VAITTINEN, ISMO:

Avoimen lähdekoodin grafiikan- ja videontoistojärjestelmä
Case: CasparCG:n käyttöönotto ja tuotteistaminen pH kolme Oy:lle

Ohjelmistotekniikan opinnäytetyö, 47 sivua

Kevät 2016

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli tutkia avoimen lähdekoodin grafiikan- ja videontoistojärjestelmä CasparCG:n käyttöönottamista ja tuotteistamista lahtelaiselle pH kolme Oy:lle.

Teoriaosuudessa käsitellään grafiikan luomista suoran televisiolähetyksen tarpeisiin ja tässä toimintaympäristössä tarvittavia erikoistekniikoita, kuten avaintamista. Lisäksi teoriaosuus tutustuttaa Ruotsin yleisradioyhtiö SVT:n kehittämän CasparCG-ohjelmiston toimintaan, CasparCG-palvelimen rajapintoihin ja grafiikkapohjien luontiin.

CasparCG:n toimivuutta testattiin käytännössä toteuttamalla sillä kaikki opiskelijoille järjestetyn pelitapahtuman grafiikat. Tapahtumaa varten koottiin oma demolaite ja tapahtumassa todennettiin myös oman asiakassovelluksen käyttämistä.

Tapahtumassa toteutetun pilotoinnin jälkeen grafiikkapohjien luontia HTML pohjaisesti tutkittiin vielä lisää ja luotiin esimerkkipohja, joka sopisi pH kolme Oy:n työnkulkuun. Lisäksi pH kolme Oy:lle toteutettiin asiakasohjelma pelikellon hallintaan.

Asiasanat: televisio, grafiikka, CasparCG

Lahti University of Applied Sciences
Degree Programme in Information Technology

VAITTINEN, ISMO:

Open source graphics and video playout system
Case: Bringing CasparCG into use for pH kolme Ltd

Bachelor's Thesis in software engineering, 47 pages

Spring 2016

ABSTRACT

The goal of this thesis was to test an open source video and graphics playout system called CasparCG. The aim was to test both server and client software, which are provided by the Swedish broadcasting company SVT, so that pH kolme Ltd. could start using them.

The thesis deals with technologies like keying, which are needed to create graphics for a live television broadcast. Creating graphics templates and the protocols needed to communicate with CasparCG server were also studied.

Testing of the hardware and CasparCG server and client software in practice was done by creating graphics for a livestream of a Counter-Strike tournament, which was hosted by the campus radio LiMu Radio of Lahti University of Applied Sciences. A custom client was also created for the tournament.

After the test event, HTML-based templates were studied further and a sample template that suits the workflow of pH kolme was created. Additionally, a program for controlling the scoreboard was made.

Key words: television, graphics, CasparCG

SISÄLLYS

1 JOHDANTO

Digitalisaation edetessä perinteinen videotekniikka on muuttunut enemmän ja enemmän kohti tietotekniikkaa. Samalla tietotekniikan kehittyessä avoimen lähdekoodin ratkaisut ovat muodostuneet varteenotettaviksi vaihtoehdoiksi monella alalla.

Kaupallisille tuoteille löytyy avoimen lähdekoodin vaihtoehtoja aina toimisto-ohjelmistoista kuvankäsittelyyn ja 3D-mallinnukseen kuin kokonaisiin käyttöjärjestelmiinkin. Grafiikan tuottamiseksi suoriin televisiolähetyksiin on perinteisesti ollut tarjolla vain isojen monikansallisten yritysten tuotteita, kunnes Ruotsin yleisradio SVT muutti kaiken vuonna 2010.

pH kolme Oy on lahtelainen vuonna 2002 perustettu ohjelmistotuotantoa ja digitaalista mediaa tarjoava yritys. Yrityksen tärkein ohjelmistotuote on ollut kuvan- ja äänenhallintajärjestelmä ScreenManager. pH kolme on myös toteuttanut tv-grafiikkaan liittyviä kokonaisuuksia useille eri urheilulajeille sekä viihdetuotantoihin.

Vuonna 2011 koko pH kolmen henkilökunta oli siirtynyt työskentelemään ChyronHego Finlad Oy:lle, joka on suuren monikansallisen grafiikkalaitevalmistan Suomen-konttori, jonka asiakkaita ovat muun muassa MTV, Yle ja Nelonen. Tämä järjestely kuitenkin päättyi keväällä 2015 ja pH kolme jatkoi taas toimintaa itsenäisesti.

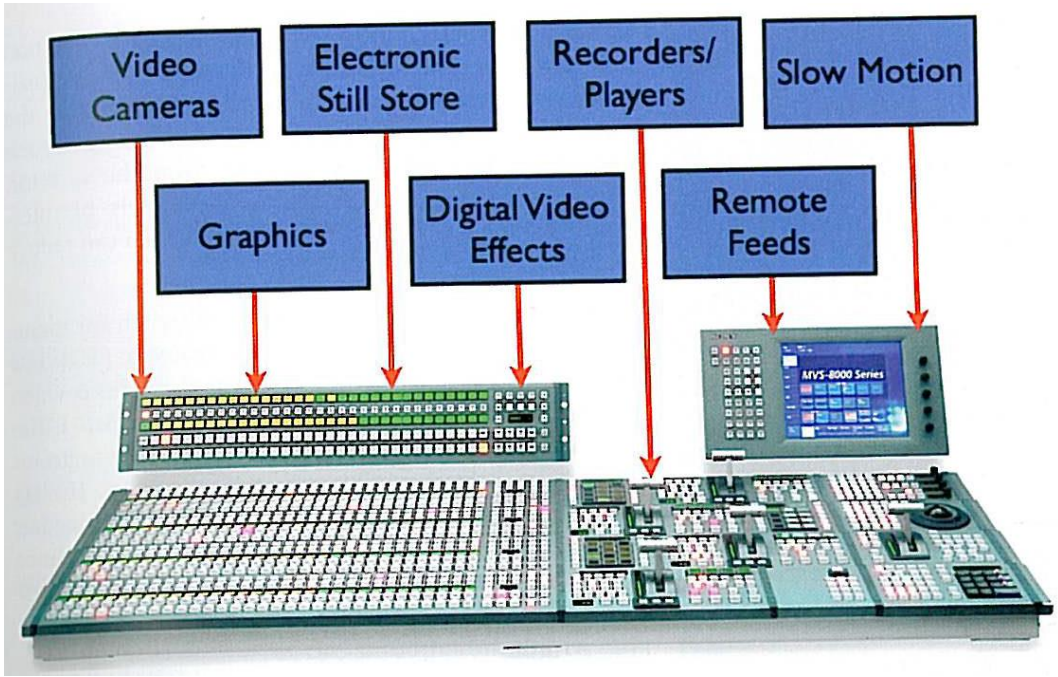
Opinnäytetyön tavoitteena oli tutusta avoimen lähdekoodin grafiikan- ja videontoistojärjestelmä CasparCG:hen ja tutkia, voisiko pH kolme hyödyntää tätä ohjelmistoa omissa teknisissä ratkaisuissaan. Lisäksi tavoitteena oli rakentaa oma grafiikkalaite ja testata järjestelmän toimivuus oikeassa tuotannossa.

Suora televisiolähetys koostuu monesta osasta. Opinnäytetyö keskittyy näistä osista reaaliaikaisen grafiikan tuottamiseen.

2 TOIMINTAYMPÄRISTÖ

Suoran televisiolähetyksen tuottaminen eroaa elokuvien tai mainosten tekemisestä siinä, että editointimahdollisuutta jälkituotannossa ei ole, vaan kaiken pitää tapahtua reaaliajassa. Tuotannoissa täytyy olla jokaiselle kuvakulmalle oma kameransa ja kuvalähteestä toiseen siirtymisen on oltava saumatonta. Suoraa lähetystä yleensä myös rikastetaan ennakkoon nauhoitetuilla inserteillä ja urheilutuotannossa usein tarkastellaan aiemmin lähetyksessä nähtyjä tapahtumia uudelleen. Lisäksi lähetyksissä on käytössä useita erilaisia graafisia elementtejä. Eikä kuva olisi mitään ilman ääntä.

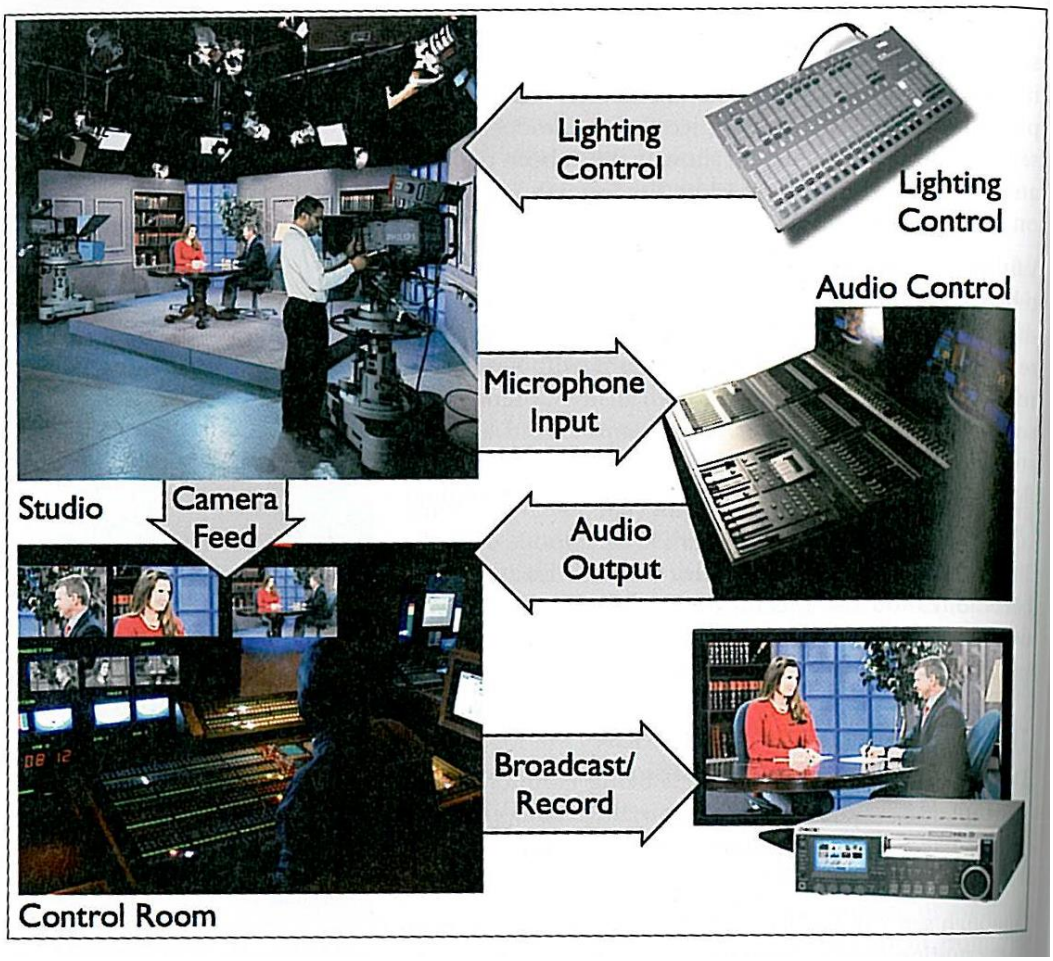
Tuotannon eri kuvalähteet yhdistetään toisiinsa kuvamikserin avulla. Kuviossa 1 on kuvattu erilaisia yleisimpiä mikseriin syötettäviä kuvalähteitä. Kuvassa jokaisen sinisen laatikon sisällä voi olla yksi tai useampi lähde. Eri äänilähteitä varten tuotannoissa on äänimikseri.



KUVIO 1. Videomikseri (Millerson & Owens 2009, 33)

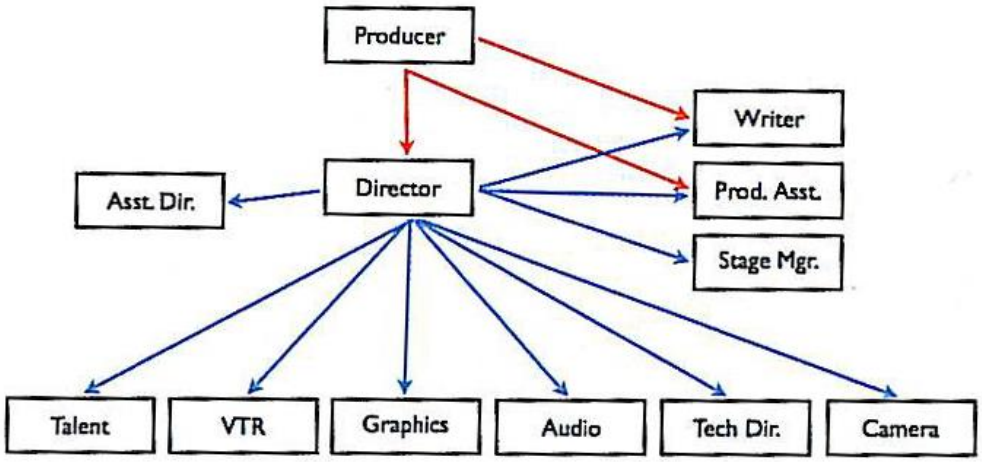
Kuvamikseri sijaitsee usein omassa erillisessä tarkkaamossa (Control Room). Tarkkaamo voi sijaita studiotilan vieressä, tai se voi olla rakennettuna ulkotuotantoautoon. Eri signaalien kulkua tarkkaamoon on

havainnollistettu kuviossa 2.



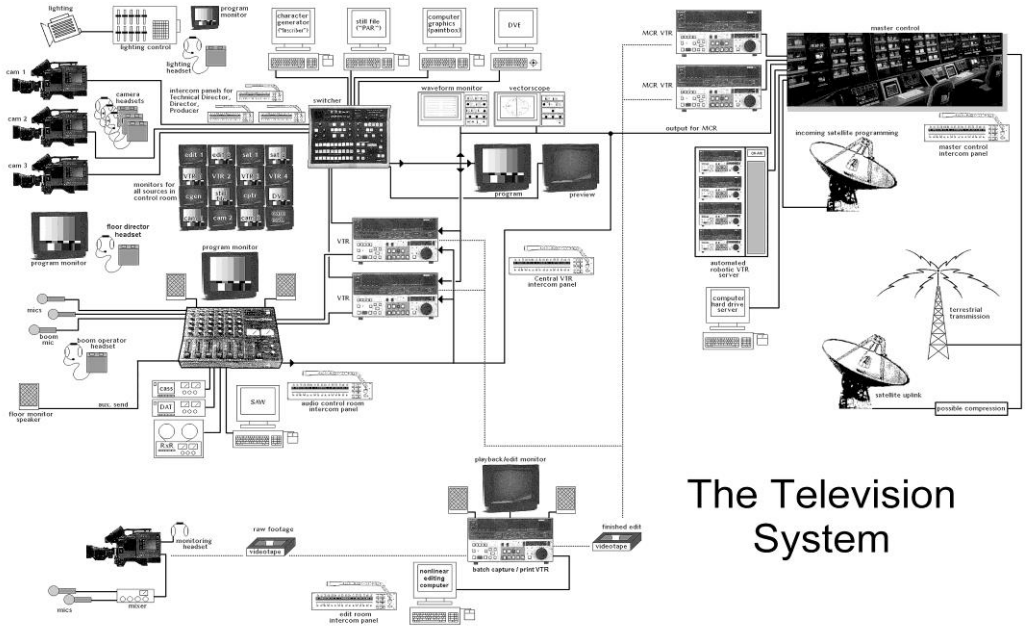
KUVIO 2. Signaalien kulku tarkkaamoon (Millerson & Owens 2009, 32)

Tarkkaamossa kuvamikserillä on usein oma käyttäjänsä (Tech Dir.), mutta pienemmissä tuotannoissa ohjaaja saattaa hoitaa miksaamisen myös itse. Ohjaaja (Director) ohjaa juontajien (Talent), kameramiesten (Camera) ja äänimiesten (Audio) lisäksi VTR-operaattoria, joka on vastuussa erilaisista videoinserteistä, sekä grafiikan ulosajajaa, joka operoi grafiikkalaitetta. Ohjaaja saa toiminnalleen raamit tuottajalta (Producer). Nämä tuotannon ammattilaiset on kuvattu kuviossa 3.



KUVIO 3. Tuotannon ammattilaiset (Millerson & Owens 2009, 19)

Kuvasignaali kulkee tuotannosta tavallisesti vielä lähetyskeskuksen kautta, ennen kuin se jaellaan maailmalle. Lähetyskeskuksessa kuvaan lisätään vielä lisää grafiikkaa, kuten kanavalogo tai mainoksia tulevista ohjelmista. Tämä kokonaisuus on kuvattu kuviossa 4.



KUVIO 4. Kokonainen lähetysjärjestelmä (Lee 2004)

3 GRAFIIKKAKONE

CG eli Character generator on yleisnimitys, jota käytetään laitteista, joilla luodaan grafiikkaa televisiolähetyksiin. Erilliset tätä tarkoitusta varten olevat laitteet hallitsivat aikaisemmin markkinaa, mutta tietotekniikan kehityttyä nämä laitteet ovat korvaantuneet ohjelmistopohjaisilla ratkaisuilla. (Millerson & Owens 2009, 182 - 183.) Ohjelmistojen toimittajat usein kuitenkin tarjoavat sekä tietokoneen että ohjelmiston pakettiratkaisuna ja laskuttavat näiden käytöstä lisenssimaksua.

Siirryttäessä ohjelmistopohjaisiin ratkaisuihin on myös grafiikkalaitteiden luonne pelkästään tekstiä luovina koneina muuttunut. Grafiikkalaitteilla onkin nykyään mahdollista myös esimerkiksi toistaa kuvaa, videota, ääntä ja 3D-grafiikkaa.

Tavallisten PC-komponenttien, kuten riittävän tehokkaan näytönohjaimen lisäksi grafiikkakoneesta on löydyttävä erillinen videokortti, sillä ammattimaisessa videonsiirrossa HDMI ei ole enää riittävä ratkaisu.

Tommy Scully (2013) onkin listannut 7 hyvää syytä, miksi HD-SDI on parempi, kuin kuluttajille suunnattu HDMI:

1. HD-SDI on tv-alalle suunniteltu formaatti, jolla on pitkä historia.
2. HDMI:ssä yhteyden nopeutta hidastaa kopiosuojaus, jollaista SDI-HD-signaalissa ei tarvita.
3. HDMI-liittimiä ei voi lukita paikoilleen, mutta HD-SDI:ssä käytössä olevat BNC-liittimet lukittuvat ja takaavat sen, etteivät johdot voi irrota vahingossa.
4. HD-SDI-signaalia voidaan kuljettaa tavallisen koaksiaalikaapelin avulla, joka on huomattavasti halvempaa kuin HDMI-kaapeli.
5. HMDI-signaalia voi kuljettaa vain noin 10 metriä ilman erillistä signaalin vahvistinta, kun taas HD-SDI-signaalia voi siirtää 300 metriä ilman ongelmia.
6. HD-SDI-signaali on helpompi skaalata alaspäin, kun taas HDMI ei ole suoraan yhteensopiva vanhemman VGA-signaalin kanssa

7. HD-SDI-signaalissa pystytään kuljettamaan videon lisäksi myös aikakoodi, kun taas HDMI-signaalissa tätä ominaisuutta ei ole. (Scully 2013.)

4 AVAINNUSTEKNIIKAT

Kun kaksi videosignaalia halutaan yhdistää yhdeksi, tulee ymmärtää tekniikka, jota kutsutaan avaintamiseksi (eng. keying). Brad Weston (2009) vertaa artikkelissaan *The basics of video keying* avaintamista maalaamiseen sapluunoiden avulla.

Sapluunoiden avulla maalatessa käytettävissä olisi pahvi, johon on leikattu kirjaimille reikä, spray-maalia sekä kyltti, jolle maalataan. Avaintaminen on tätä vielä hieman hienostuneempaa, sillä maalin määrää voidaan säädellä. Sapluunavertauksen maali ja sen määrä vastaavat täyttösignaalia ja sen läpinäkyvyyttä. Sapluuna sen sijaan kuvaa mustavalkoista avainnussignaalia. (Weston 2009, 24.)

Tässä mustavalkoisessa signaalissa valkoinen väri päästää kaiken täyttösignaalissa olevan näkyviin ja musta ei päästä mitään läpi. Valkoista osaa kuvassa kutsutaan usein leikkaavaksi osaksi. Eri harmaan sävyt mustan ja valkoisen välissä paljastavat kuvaa aina sitä enemmän, mitä lähempänä väri on valkoista. Vertauksen kyltti sen sijaan kuvaa taustalla olevaa toista videosignaalia, johon kuvaa ollaan yhdistämässä (Weston 2009, 24).

Kun puhutaan kolmesta yleisimmästä avainnusteniikasta (chroma, luma, linear), niin käytännössä puhutaan siitä, miten avainnussignaalin mustavalkokuva luodaan. Chroma- ja luminance-avainnuksissa avainnussignaali luodaan dynaamisesti videolähteestä, kun taas linear-avainnuksessa käytetään erillistä alphakanavaa. (Weston 2009, 24.)

4.1 Chroma key

Chroma key on tekniikka, jota on perinteisesti käytetty säätiedotuksissa. Avainnus perustuu väriin ja tapahtuu yleisimmin joko vihreästä tai sinisestä väristä.

Avainnussignaaliksi muodostuu tällöin taustan edessä olevan ihmisen valkoinen siluetti. Täyttö-signaalina toimii samasta lähteestä peräisin oleva

ihminen ja taustana toimii sääkartta, jonka päälle säämiehen kuva yhdistetään. Lähes jokaisesta videomikseristä löytyy mahdollisuus käyttää chroma keytä. (Weston 2009, 24.)

Chroma keytä käytettäessä joudutaan kuitenkin luopumaan yhdestä väristä täyttö-signaalissa (Weston 2009, 24). Irrottaessa ihmishahmoa taustasta tämä ei haittaa niin paljon, mutta kun käsitellään grafiikkaa, tämä rajoite on merkittävämpi.

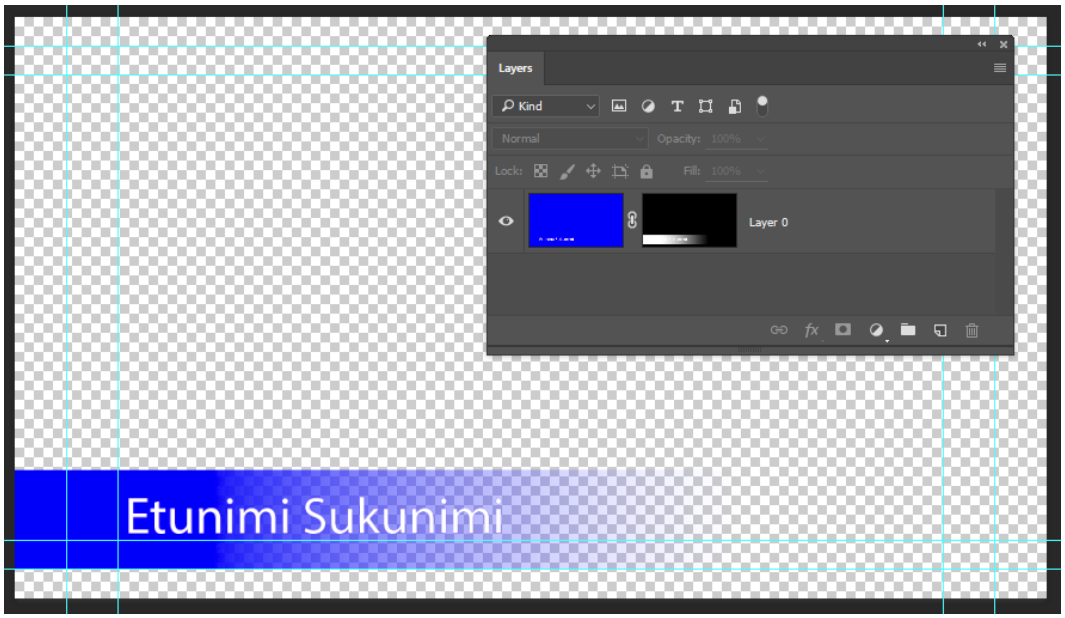
Chroma keytä käytettäessä syntyy myös eräänlainen haamu-efekti, kun kuvia häivytetään. Tuolloin kuva muuttuu aina vain tummemmaksi, samalla kun sen läpinäkyvyytä lisätään. (Weston 2009, 25.) Näin ollen valkoinen teksti muuttuisi mustaksi samalla kun sitä häivytetään.

4.2 Luma key

Luma keyssä avainnussignaalin luomisen perusteena toimii kuvan kirkkaus. Mahdollisuus käyttää Luma keytä on yleensä käytettävissä vain kalliimman hintaluokan miksereissä. Tämä tekniikka kärsii myös haamuefektistä, tosin tätä ei tapahdu olenkaan, jos käytetään erillistä täyttösignaalia. Tällä tekniikalla ei ole mahdollista käyttää tummia värejä osana näkyvää kuvaa, sillä mitä tummempi kuva on, sitä enemmän sillä on myös läpinäkyvyyttä. (Weston 2009, 25.)

4.3 Linear key

Linear key on käytettävissä olevista tekniikoista paras mahdollinen, sillä se mahdollistaa täydellisen värien ja läpinäkyvyyksien hallinnan. Tässä tekniikassa avainnus- ja täyttösignaalit ovat erilliset. (Weston 2009, 26.) Täydellinen kuvien yhdistäminen aiheuttaa sen, että videomikserin rajallisista sisääntuloista kuluu yhden sijaan kaksi. Adoben Photoshop-ohjelman tasomaskit toimivat samalla tavoin, kuten kuviosta 5 voi nähdä.



KUVIO 5. Maski Adobe Photoshop-ohjelmassa

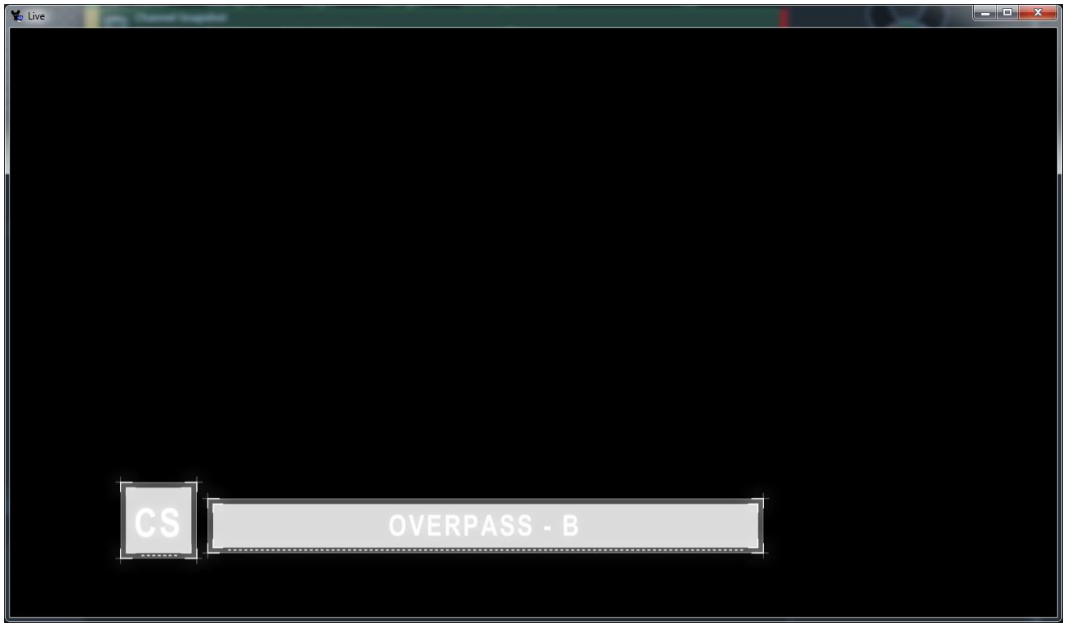
4.4 Alpha-kanava

Videosignaalin alpha-kanavan esittämiseen on olemassa kaksi erilaista tapaa: straight ja premultiplied. Lopputulos on molemmilla tekniikoilla sama, kunhan signaalia tulkitaan oikein. Avainnussignaali on molemmissa samanlainen mustavalkokuva, mutta kun kyseessä on premultiplied alpha, niin täyttösignaaliin yhdistetään myös läpinäkyvyystietoa kertomalla jokainen RGB-väriavaruuden värikanava arvolla, joka vastaa läpinäkyvyyttä. Tällöin täyttösignaali yhdistetään matte väriin, joka on yleensä musta, mutta voi olla mikä tahansa muukin. (CG Director 2011.) Käytettäessä premultiplied alphaa tulee myös mikserin asetukset säätää vastaamaan grafiikkalaitteen asetuksia.

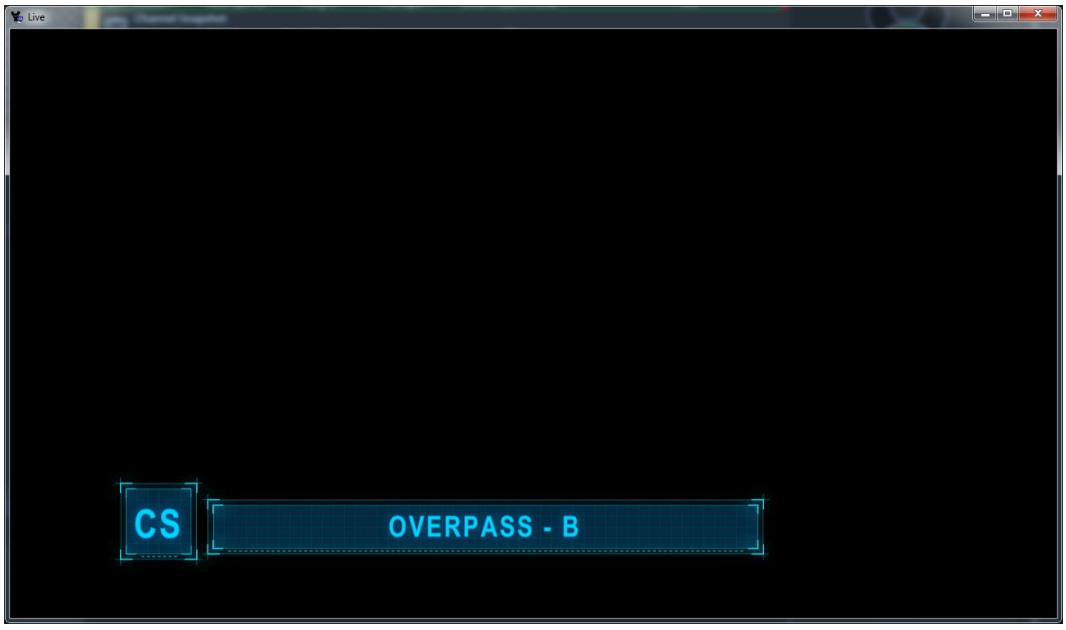
CasparCG-palvelimen 2.0-versio olettaa kaiken toistettavan sisällön olevan luotu premultiplied-tekniikalla. Versio 2.0.4 lisäsi tuen straight alpha -tekniikalle ulostulovaiheessa, mutta jokaisen moduulin sisällön tulee edelleen olla luotu premultiplied-tekniikalla. (CasparCG Wiki 2016c.)

Kuviossa 6 on nähtävissä esimerkki mustavalkoisesta avainnussignaalista. Kun tämän signaalin lisäksi kuvamikserille lähetetään joko kuvion 7 tai

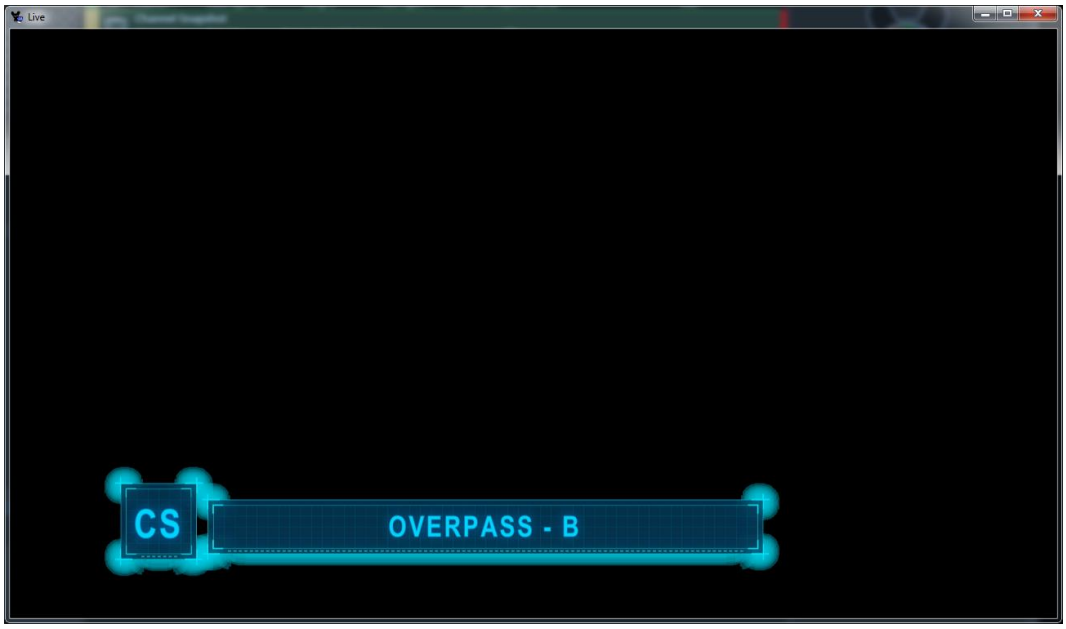
kuvion 8 mukainen täyttösignaali, on lopputulos kuvion 9 mukainen, kunhan mikserin asetukset on asetettu oikein.



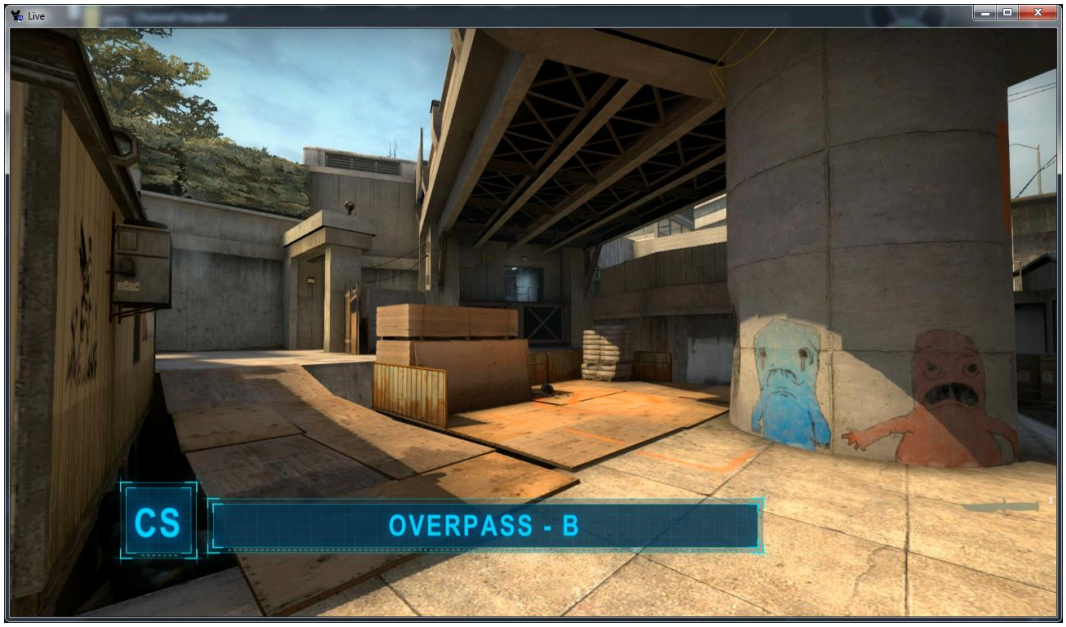
KUVIO 6. Avainnussignaali



KUVIO 7. Täyttösignaali premultiplied alpha -tekniikalla



KUVIO 8. Täyttösignaali straight alpha -tekniikalla



KUVIO 9. Avainnus- ja täyttösignaalit yhdistettynä taustaan

4.5 Videosignaalien synkronointi

Kun käytössä on kaksi tai useampi kuvalähdettä, täytyy ne saada samaan tahtiin keskenään. Epätahti voi ilmetä esimerkiksi grafiikoiden välkkymisenä tai seilaamisena ruudulla.

Tahdistukseen on aikojen saatossa syntynyt useita eri tapoja, joista aikaisin on genlock, joka tahdistaa kuvalähteet ulkoisen signaalin avulla. Tällaiseen tahdistamiseen voidaan käyttää esimerkiksi Tri-Sync- tai Blackburst-signaaleja. Tämä tekniikka toimii niin kauan kuin ulkoista tahtia ei häiritä. Ulkoisen tahdin häiriintyessä kaikki paikalliset laitteet menettäisivät tahtinsa. (Millerson & Owens 2009, 368.) Tätä tekniikkaa voikin verrata marssimiseen rummun tahdissa.

Reaaliaikaisten signaalien, kuten kameroiden, tahdistukseen käytetään yleisimmin frame sync -tekniikkaa ja esivalmistellut materiaalit tavallisimmin tahdistetaan time-base corrector (TBC) -tekniikan avulla (Millerson & Owens 2009, 368). Videotekniikan digitalisoitumisen myötä videolähteiden tahdistaminen on helpottunut huomattavasti, sillä laitteet, joilla on digitaalisia sisääntuloja, voivat myös tahdistaa ne. Asiat kuitenkin monimutkaistuvat, mikäli tuotannossa on mukana analogisia lähteitä. Analogiset mikserit täytyy tahdistaa nanosekuntien tarkkuudella, mutta digitaalisille miksereille riittää tahdistukseen yhden vaakaviivan tarkkuus. (Tektronix Inc. 1997, 21.)

5 CASPARCG

Ruotsin yleisradioyhtiö SVT lähti kehittämään omia tarpeitaan varten omaa CG-ohjelmistoa, CasparCG:tä, vuoden 2006 vaalilähetyksiä varten. Taustalla oli turhautuminen vuoden 2002 lähes täysin epäonnistuneeseen lähetykseen, jolloin tunnettu grafiikkalaitteiden toimittaja laskutti liikaa ja toimitti liian vähän. 2002 ja 2006 vaalien välillä tietokoneiden luonne oli muuttunut pelkistä kirjoituskoneista graafisiksi työasemiksi ja samaan aikaan web-teknologiat olivat kasvaneet räjähdysmäisesti. Tämä mahdollisti grafiikalaitteen uudelleen keksimisen. (Hummelstrand 2012, 1.)

Onnistuneen vaalilähetyksen jälkeen SVT:n uutisohjelmat korvasivat vanhat vizRT- ja Chyron-laitteensa vuonna 2008 ja urheilu ja ulkotuotannot seurasivat pian perässä. Vuonna 2010 myös kanavan kanavailme siirryttiin toteuttamaan CasparCG:n avulla. Samana vuonna SVT julkaisi CasparCG:n yleisölle ilmaiseksi. SVT ei ollut ensimmäinen yleisradioyhtiö, joka julkaisee avoimen lähdekoodin työkaluja, mutta SVT:lle kyseessä oli iso askel. (Hummelstrand 2012, 3.)

Tämä askel tarkoitti sitä, että ohjelmiston vapaaseen käyttöön julkaisemisen lisäksi myös lähdekoodista tehtiin julkista. Näin kuka tahansa voi muokata ohjelmistosta itselleen sopivan version tai osallistua kehitystyöhön tekemällä parannusehdotuksia tai vikaraportteja.

CasparCG toimii asiakas-palvelin-mallin mukaisesti, ja SVT tarjoaa CasparCG.com-sivuilla ladattavaksi sekä palvelin- että asiakassovelluksen. Sovellusten välistä kommunikaatiota käsitellään tarkemmin luvussa 6.

CasparCG:llä on myös hyvin aktiivinen yhteisö, joka on toteuttanut lukuisia erilaisia asiakassovelluksia. Yhteisö on luonut iPad-käyttöliittymiä vaalilähetyksiin, selainkäyttöliittymän baseballille, jääkiekkokäyttöliittymän Android-puhelimille sekä MIDI-ohjattuja fyysisiä nappeja. Yksinkertainen verkkoprotokolla mahdollistaa palvelimen ohjaamisen lähes millä tahansa alustalla tai laitteella. (Hummelstrand 2012, 4.)

Tästä aktiivisesta yhteisöstä on helppo hakea tukea CasparCG:n kotisivuilla olevan foorumin kautta. Myös SVT:n kehitystiimi osallistuu keskusteluihin aktiivisesti. Kehitystiimi on mahdollista tavoittaa myös CasparCG:n IRC-kanavalta.

5.1 Palvelin

CasparCG-palvelimella on mahdollista käsitellä useita sisään- ja ulostuloja samalla koneella. Kaikki sisällöt, kuten kuvat, videot ja dynaamiset grafiikkapohjat, voidaan pinota omiksi tasoikseen. Näiden tasojen ominaisuuksia, kuten sijaintia, kokoa ja läpinäkyvyyttä, voidaan vapaasti animoida. Lisäksi tasoille voidaan käyttää erilaisia sekoitustiloja Adoben Photoshopin tapaan. CasparCG osaa käyttää tiedostoihin sisällytettyjä alpha-kanavia automaattisesti, mutta tasoille voidaan myös luoda erillinen alpha-kanava palvelimen toimesta. Tämä on hyödyllistä silloin, jos videoihin halutaan leikata reikiä dynaamisella datalla, kuten tekstillä, tai jos käytettävissä oleva videomateriaali on pakattu sellaisella kodeekilla, jossa ei ole alpha-kanavaa, kuten H.264. (Hummelstrand 2012, 2.)

CasparCG-palvelimen vaatimukset on listattu taulukossa 1. Lisäksi erillisen näytönohjaimen käyttämistä suositellaan prosessoriin integroidun näytönohjaimen sijaan. Windowsin Aero-teema tulisi myös ottaa pois päältä, samoin ClearType-fontin pehmennys (CasparCG Wiki 2016b.)

TAULUKKO 1. CasparCG-palvelimen vaatimukset (CasparCG Wiki 2016b)

Intel processor capable of using SSSE3 instructions.
A graphics card (GPU) capable of OpenGL 3.0 is required.
Windows 7 (64-bit) strongly recommended.
Microsoft Visual C++ 2010 Redistributable Package must be installed.
Microsoft .NET Framework" (version 4.0 or later) must be installed.

Käyttöjärjestelmäksi suositellaan 64-bittistä versiota Windows 7:stä. CasparCG toimii myös 32-bittisessä Windows 7:ssä sekä Windows XP:n

SP2-versioissa. Sen sijaan Windows 8, Windows 2003 Server tai Windows Vista eivät ole tuettuina. (CasparCG Wiki 2016b.)

TAULUKKO 2. SVT:n käyttämä kokoonpano grafiikkakoneelle (CasparCG Wiki 2016b)

1 pc.	HP Z420 4-core Xeon E5-1620 @ 3.6GHz with 8 GB ECC RAM. HP part number is B2B93UT or WM445ET#ABU.
1 pc.	HP NVIDIA Quadro 2000 (1 GB) GPU (WS094AA)
2 pcs.	OCZ VERTEX 3 2.5" 240GB SSD SATA/600 MLC (VTX3-25SAT3-240G)
2 pcs.	BlackMagic Design DeckLink 4K Extreme SDI video cards for a two channel System, otherwise one card for fill and key is sufficent.
1 pc.	Windows 7 Professional SP1 64-bit

5.1.1 Asetukset

Kehittäjien fokus on ollut enemmän palvelimen suorituskyvyssä kuin *out-of-the-box*-elämyksen aikaansaamisessa. CasparCG ei ole aloittelijaystävällisin ohjelmisto, mutta monimutkaisempiin työnkulkuihin tottuneille sen käyttöönotto on helppoa. (Hummelstrand 2012, 4.) CasparCG:n asetuksille ei esimerkiksi ole käyttöliittymää, vaan asetukset tehdään muokkaamalla xml-tiedostoa. Kuviossa 10 on esimerkki tästä asetustiedostosta.

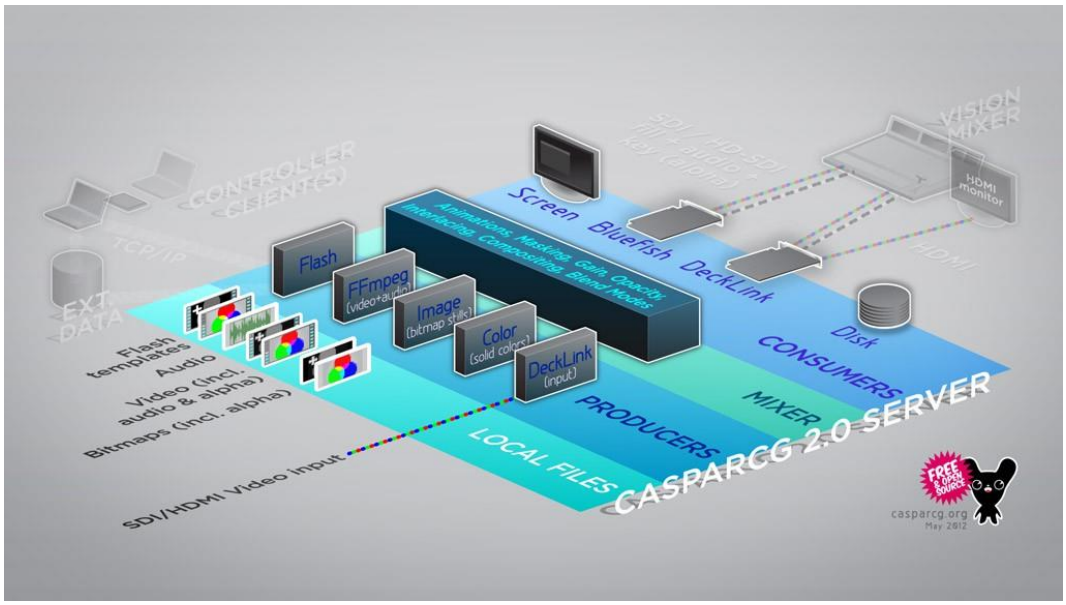
Asetusten asettamiseksi oli aiemmin yhteisön luoma käyttöliittymä, mutta tuolla projektilla ei ole tällä hetkellä ylläpitäjää. Tuo käyttöliittymä ei ole siten enää yhteensopiva uusimman palvelinversion kanssa. (Didikunz 2014.)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <paths>
    <media-path>media\</media-path>
    <log-path>log\</log-path>
    <data-path>data\</data-path>
    <template-path>templates\</template-path>
    <thumbnails-path>thumbnails\</thumbnails-path>
  </paths>
  <channels>
    <channel>
      <video-mode>1080i5000</video-mode>
      <consumers>
        <decklink>
          <device>1</device>
          <key-device>2</key-device>
          <embedded-audio>false</embedded-audio>
          <channel-layout>stereo</channel-layout>
          <latency>normal</latency>
          <keyer>external</keyer>
          <key-only>false</key-only>
          <buffer-depth>3</buffer-depth>
          <custom-allocator>true</custom-allocator>
        </decklink>
        <system-audio />
      </consumers>
    </channel>
  </channels>
  <controllers>
    <tcp>
        <port>5250</port>
        <protocol>AMCP</protocol>
    </tcp>
  </controllers>
  <diagnostics>
 <graphs>true</graphs>
</diagnostics>
</configuration>
```

KUVIO 10. CasparCG-asetustiedosto

5.1.2 Moduulit

CasparCG-palvelimen toiminnallisuus jakautuu kuvion 11 mukaisiin osiin: paikallisiin toistettaviin tiedostoihin, sisääntulomoduuleihin (producers), mikseriin ja ulostulomoduuleihin (consumers).



KUVIO 11. CasparCG moduulit (CasparWiki 2015a)

Videoiden toistaminen tapahtuu CasparCG:ssä ffmpeg-sovelluskehyksen (framework) avulla. Ffmpeg:n ansiosta CasparCG:llä on mahdollista toistaa kaikkia yleisimpiä videoformaatteja (About Ffmpeg 2016).

Dynaamisen tekstin ja grafiikkapohjien luontiin CasparCG käyttää Flash-moduulia. Ainoa vaihtoehto grafiikkapohjien luontiin oli aiemmin maksullinen Adoben Flash Professional, mutta nykyään flash-pohjia on mahdollista luoda myös ilmaisen Flash Develop -ohjelman avulla. CasparCG:n tunnistamien pohjien luomista varten tarvitaan lisäksi erillinen Template Generator -lisäosa. (CasparCG Wiki 2016c).

CasparCG:n versiosta 2.0.7 beta 1 lähtien grafiikkapohjia on ollut mahdollista luoda myös HTML:n avulla. Aikaisemmin CasparCG:n html-moduuli perustui Berkelium-kirjastoon, mutta uusimman version toteutuksessa on käytetty Googlen kehittämää CEF-selainta. Tämä uusi selain avaa mahdollisuudet käyttää HTML 5:n ja CSS 3:n tuomia animaatiomahdollisuuksia. Samalla on auennut myös mahdollisuus luoda 3D-grafiikkaa WebGL-rajapinnan kautta. (Karlsson 2014; Hummelstrand 2014.)

Mikseri-moduuli on vastuussa tasojen animoinneista, maskeista ja sekoitustiloista. Mikseri-moduulin avulla on mahdollista myös hyödyntää lähdemaeriaaleihin chroma key -tekniikkaa.

Ulostulomoduuleista löytyy mahdollisuus striimata suoraan verkkoon, tallentaa levylle, toistaa grafiikka tietokoneen näytöllä tai käyttää ulostuloon ammattimaista videokorttia, jolla saadaan aikaiseksi SDI-signaali.

5.2 SVT:n asiakassovellus

SVT:n kehittäjätiimi on keskittynyt enemmän palvelimen toimintoihin kuin geneerisen asiakassovelluksen luomiseen ja sen käytettävyyteen. SVT:n kehittäjät yleensä luovat oman käyttöliittymän eri ohjelmien erikoistarpeille. (Hummelstrand 2012, 3.)

CasparCG:n kotisivuilla tarjoama asiakassovellus on kuitenkin erittäin monipuolinen, ja sillä on muun muassa mahdollista luoda kuva- ja videotasoja ja toteuttaa niille erilaisia siirtymiä. Lisäksi sovelluksen avulla voi hyödyntää grafiikkapohjia ja välittää niille dataa avain-arvopari muodossa.

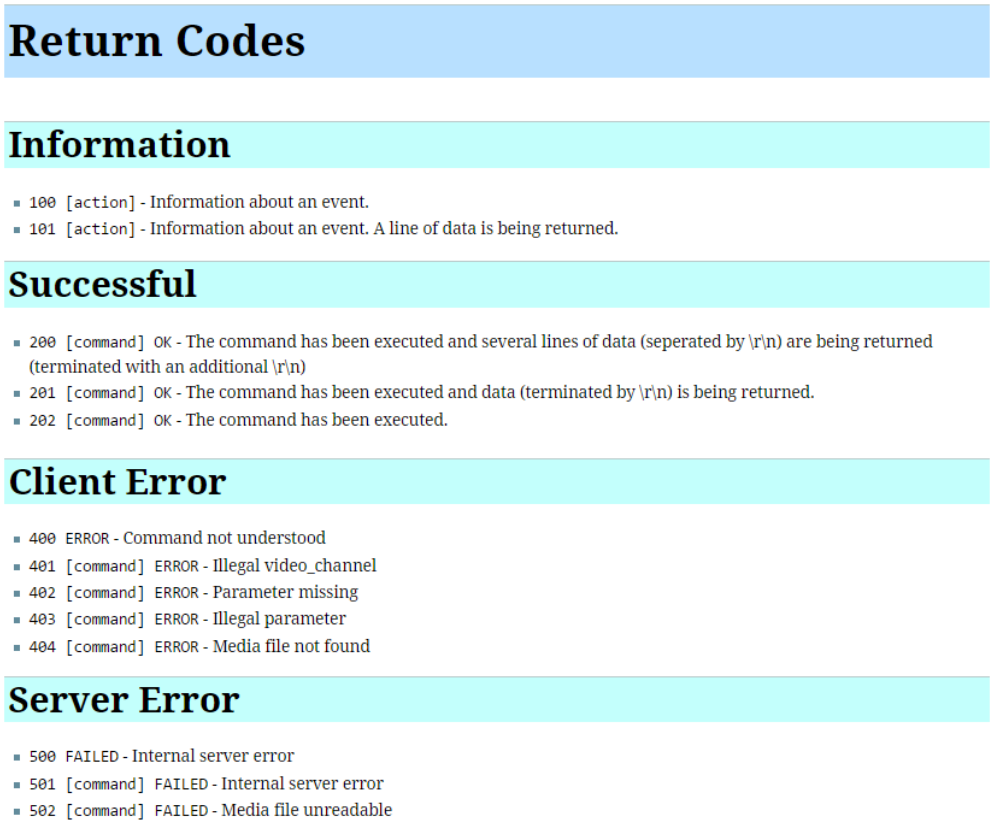
SVT:n asiakassovellus laajentaa järjestelmän toimintaa niin, että CasparCG-palvelimen lisäksi sillä on mahdollista ohjata esimerkiksi Black Magicin, Panasonicin ja Tricasterin kuvamiksereitä. Tätä asiakassovellusta voi myös etähallita OSC- ja GPIO-protokollien avulla. Tämä tarjoaa hyvän lisän erilaisten automaatioiden rakentamiselle.

6 RAJAPINNAT

CasparCG-palvelinta voidaan ohjata AMCP (Advanced Media Control Protocol) -protokollan avulla. Lisäksi palvelin lähettää itsestään tilatietoja asiakasohjelmille OSC-protokollan avulla. SVT:n asiakasohjelmisto tuo mukaan myös mahdollisuuden GPI-ohjaukselle. HTML- ja Flash-grafiikkapohjat taas antavat käyttöön erilaisia WEB-tekniikoita.

6.1 AMCP

AMCP-protokollaa ei ole mitenkään standardoitu vaan se on SVT:n vain tähän käyttötarkoitukseen kehittämä protokolla. Protokollan komennot ovat UTF-8 koodattua selkotekstiä, joka kulkee palvelimelle tcp/ip:n yli hyödyntäen telnet-protokollaa. Palvelin palauttaa tiedon komentojen onnistuneesta tai epäonnistuneesta vastaanottamisesta samaa tcp/ip sockettia hyödyntäen. (CasparWiki 2015a). Mahdolliset paluukoodit on esitetty kuviossa 12.



Return Codes

Information

- `100 [action]` - Information about an event.
- `101 [action]` - Information about an event. A line of data is being returned.

Successful

- `200 [command] OK` - The command has been executed and several lines of data (seperated by \r\n) are being returned (terminated with an additional \r\n)
- `201 [command] OK` - The command has been executed and data (terminated by \r\n) is being returned.
- `202 [command] OK` - The command has been executed.

Client Error

- `400 ERROR` - Command not understood
- `401 [command] ERROR` - Illegal video_channel
- `402 [command] ERROR` - Parameter missing
- `403 [command] ERROR` - Illegal parameter
- `404 [command] ERROR` - Media file not found

Server Error

- `500 FAILED` - Internal server error
- `501 [command] FAILED` - Internal server error
- `502 [command] FAILED` - Media file unreadable

KUVIO 12. Paluukoodit (CasparWiki 2015a)

Suurin osa komennoista koostuu aina itse komennosta, kanava- ja tasotiedosta, sekä komennon vaatimasta datasta. Kuviossa 13 on esimerkki MY_FILE-nimisen videon lisäämisestä ensimmäisen kanavan tasolle numero 1. Tämä video voitaisiin lataamisen jälkeen toistaa PLAY-komennolla.

```
>> LOAD 1-1 MY_FILE
```

KUVIO 13. Esimerkki LOAD-komennosta (CasparCG Wiki 2016a)

Kuviossa 14 oleva komento taas muokkaa ensimmäisen kanavan alimman tason kokoa niin, että sen leveys on 50 % ja korkeus 50 % ja etäisyys vasemmasta reunasta ja yläreunasta on 25 %. Lisäksi toiminnolle on asetettu 25 ruutua kestävä siirtymä.

```
>> MIXER 1-0 FILL 0.25 0.25 0.5 0.5 25 easeinsine
```

KUVIO 14. Esimerkki MIXER-komennosta (CasparCG Wiki 2016a)

Grafiikkapohjien lisäämisessä ja päivittämisessä komennon mukana kulkee paljon enemmän dataa. Tämä data lähetetään palvelimelle joko XML- tai JSON-muodossa. Kuviossa 15 on esimerkki XML-muotoisesta datasta.

```
<templateData>
        <componentData id="f0">
                <data id="text" value="Niklas P Andersson" /> </componentData>
        <componentData id="f1">
                <data id="text" value="Developer" />
        </componentData>
        <componentData id="f2">
                <data id="text" value="Providing an example" />
        </componentData>
</templateData>
```

KUVIO 15. Esimerkki grafiikkapohjalle välitettävästä xml-datasta (CasparCG Wiki 2016b)

6.2 OSC

Open Sound Control (OSC) on avoin, siirtotavasta riippumaton, viestipohjainen protokolla. OSC on kehitetty tietokoneiden, syntetisaattoreiden ja muiden multimedialaitteiden väliseen kommunikointiin. Verkossa, jossa lähetetään OSC-paketteja, täytyy lähettää vastaanottaville sovelluksille sekä paketin sisältö että sen koko. OSC-paketteja on mahdollista lähettää verkkoprotokollilla, kuten UDP:lla tai TCP:lla. (Wright 2002.)

CasparCG-palvelin lähettää myös tietoa omasta tilastaan, kuten missä kohtaa videotiedoston toistaminen on meneillään, OSC-protokollan avulla. SVT:n asiakassovellusta on myös mahdollista etähallita tämän saman protokollan avulla.

6.3 GPIO

CasparCG:n virallinen asiakasohjelmisto sisältää tuen myös GPIO (General purpose input/output) -protokollalle, jonka avulla asiakassovelluksen on mahdollista kommunikoida suoraan kuvamikserin kanssa. Tämä vaatii kuitenkin erillisten laitteiden lisäämistä koneeseen.

SVT on käyttänyt tähän tarkoitukseen IoCorea ja Arduinoa. SVT on myös kokeillut komentojen välittämistä PBus2-väylän avulla mikserille, mutta asiakassovellusta tämän hyödyntämiseksi ei ole julkaistu. (CasparCG Wiki 2016b.)

6.4 WEB-tekniikat

Html- ja Flash-grafiikkapohjat laajentavat CasparCG:n kommunikointiprotokollien määrää entisestään, sillä grafiikkapohjille voi välittää tietoa suoraan erilaisilla web-tekniikoilla ilman, että viestit kulkevat palvelimen rajapintojen lävitse. Tällaisissa ratkaisuissa on kuitenkin syytä harkita mahdollisuutta kierrättää data oman asiakas-sovelluksen läpi. Varsinkin, jos datan oikeellisuutta tai saatavuutta halutaan tarkastella

operaattorin toimesta. Useimmiten grafiikoita ei haluta ajaa ruutuun sokkona.

Mahdollisuus käyttää web-tekniikoita on kuitenkin suuri etu, sillä useimmat tulospalvelujärjestelmät on rakennettu selaimia silmällä pitäen, eikä esimerkiksi eri urheilulajien lajiliitoilla välttämättä ole resursseja luoda erillisiä palveluja pelkästään tv-grafiikan tarpeisiin.

Web-tekniikoilla on helppoa esimerkiksi kytkeytyä tietokantoihin, hakea tietoa REST- tai SOAP-palveluista. Myös XML- ja JSON-muotoisen datan hyödyntäminen muuttuu vaivattomaksi.

Datan ajastetusti tai latauksen yhteydessä hakemisen rinnalle on syntynyt myös toinen vaihtoehto. Websocket-tekniikan avulla on mahdollista saada esimerkiksi tulospalvelun tarjoajilta ilmoituksia heti, kun uutta dataa on saatavilla. Websocket synnyttää myös mielenkiintoisen mahdollisuuden välittää omaa dataa suoraan grafiikkapohjille häiritsemättä palvelinta.

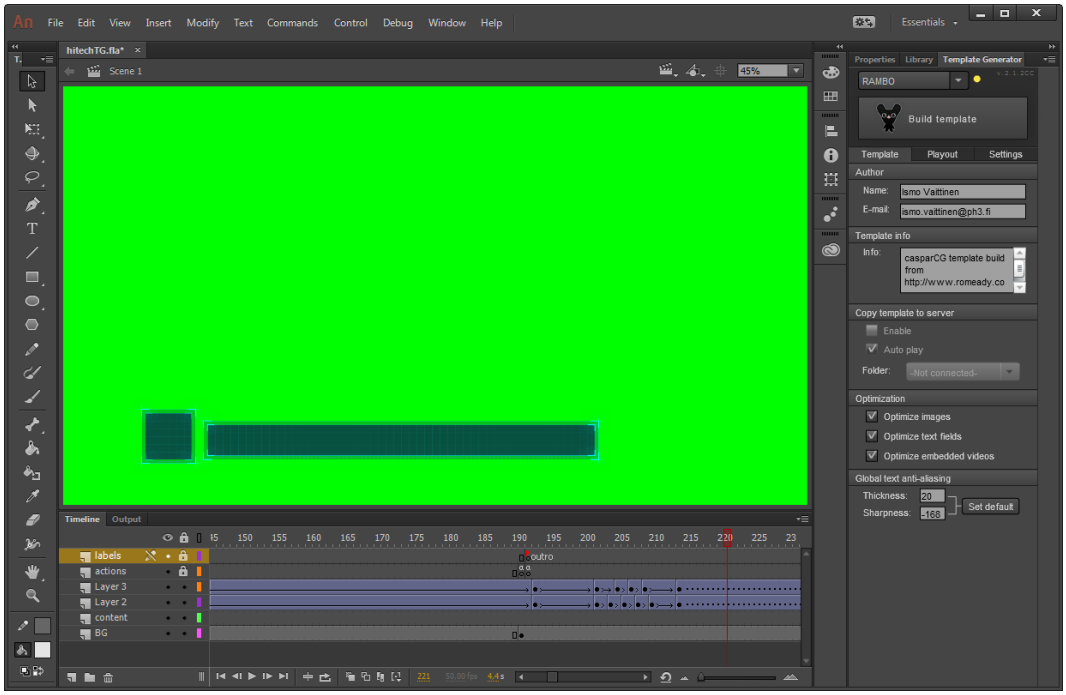
7 GRAFIIKKAPOHJAT

Staattisten videoiden ja kuvatiedostojen toistaminen onnistuu muillakin laitteilla kuin CG-grafiikkakoneella. Mahdollisuus dynaamisen sisällön ja grafiikkapohjien hyödyntämiseen on kuitenkin perusteena erillisen CG-koneen käyttämiseen tuotannoissa. Grafiikkapohjien avulla on mahdollista esimerkiksi näyttää urheilulähetyksissä tilastotietoja sekä esitellä sellaisia haastateltavia, jotka eivät ole olleet ennakkoon tiedossa.

7.1 Flash

Dynaamisten grafiikkapohjien luontiin SVT:n kehittäjät ovat valinneet Flashin. Flash-pohjia on mahdollista luoda Adoben Animate CC -tuotteella (aiemmin Flash professional).

Grafiikkapohjien luontiin Animate CC:llä on CasparCG:n kotisivuilta ladattavissa lisäosa, joka generoi Flash-projektista CasparCG:n ymmärtämän ft-tiedoston. Kuvio 16 on tämän lisäosan käytöstä. Tarjolla on myös valmiita projektipohjia eri televisiostandardeille. Grafiikkapohjiin on mahdollista lisätä edistyneempiä toimintoja ActionScriptin avulla.



KUVIO 16. Adobe animate CC ja CasparCG template generator

Animate CC:llä on helppoa hyödyntää materiaalia, joka on toimitettu kuvasekvenssinä, joka on tavallinen tapa toimittaa materiaali. Dynaamisten elementtien animoinnin tahdistaminen tähän materiaaliin on helppoa, sillä jokaista ruutua pääsee tarkastelemaan Animate CC:ssä erikseen.

Materiaalista pitää kuitenkin tehdä movieclip, jonka ruutuja ei enää pääsekään tarkastelemaan tahdissa pääaikajanan kanssa. Tämä on pakko tehdä, jotta pääsee asettamaan blur filtterille arvon 0. Ilman tätä asetusta virhe ohjelmistossa ei jätä movieclipin taustaa läpinäkyväksi, vaan tekee siitä mustan. Flashin looppaava luonne asettaa myös haasteita ja lisätyötä, kun jokaiselle movieclipille täytyy muistaa asettaa stop()-komento.

Asettamalla animaatiolle ruudun, jonka *label* on outro, saadaan toimintoja rytmitettyä niin, että asiakasohjelmistolta tuleva play-komento toistaa animaatiot tuohon ruutuun asti ja stop-komento sitten jatkaa siitä.

Vaikka Flashin suosio internetsivustoilla onkin hiipumassa, on se tässä käytössä kuitenkin edelleen toimiva ratkaisu.

7.2 Html

CasparCG-palvelimen versiossa 2.0.7 on toteutettuna myös tuki HTML-pohjaisille grafiikkapohjille. Tuki on toteutettu hyödyntäen aktiivisesti ylläpidettyä Chromium Embedded Framework (CEF 3) -sovelluskehystä. Tätä kehystä on lisäksi muokattu niin, että window.requestAnimationFrame-toiminnosta on luotu uusi versio, joka mahdollistaa animaatioiden tahdistamisen CasparCG:n kanavien tahtiin. Näin saadaan aikaan täydellisen sulavia animaatioita (Karlsson 2014).

Ilman visuaalista työkalua animointi voi olla haastavaa, joten työskentelyn helpottamiseksi ja suorituskyvyn parantamiseksi apuna voi käyttää erilaisia animaatiokirjastoja. Yksi tällainen animaatiokirjasto on Greensock Animation Platform (GSAP).

GreenSock Animation Platform (GSAP) -kirjaston luoja Jack Doyle on verrannut CSS:n, jQueryn ja GSAP-kirjaston suorituskykyä toisiinsa. Tässä testissä CSS oli jQueryä nopeampi, kuten oli odotettua, mutta GSAP oli vielä CSS:ää suorituskykyisempi. (Doyle 2014.)

CSS:n avulla on mahdollista pysäyttää ja jatkaa animaatioita, mutta GSAP JavaScript-kirjaston avulla animaatioita voi myös toistaa takaperin. Myös toistonopeutta ja animaation aloituskohtaa voi hallita. Lisäksi animaation eri vaiheille voi luoda callback-funktioita, joiden avulla pystytään luomaan rikkaampia animaatioita. (Doyle 2014.)

Animaatioita halutaan yleensä säätää sekunneissa, ei prosenteissa. Tämä onnistuu JavaScriptillä, mutta CSS:ssä voi asettaa vain koko animaation keston, ei yksittäisten arvojen muutosten kestoja. CSS:ssä yksittäiset arvot asetetaan prosenteissa. (Doyle 2014.)

```
<style>
div {
    width: 600px;
    height: 60px;
    left:200px;
    bottom: 200px;
    white-space: nowrap;
    overflow: hidden;
    padding: 10px;
    background-color: yellow;
    position: absolute;
    font-size: 48px;
    font-family: Arial;

    animation-name: in-animation;
    animation-duration: 2s;
}

@keyframes in-animation {
    0%   {background-color:red; left:0px; width: 0px;}
    25%  {background-color:red; left:250px;}
    50%  {background-color:yellow;}
    100% {width: 600px;left:200px;}
}
</style>
```

KUVIO 17. CSS-animaatio

GSAP-kirjasto tarjoaa seek()-komennon, jolla voi hypätä animaatiossa eteenpäin ja hienosäätää animaation loppuosaa ilman, että tarvitsee katsoa koko animaatiota läpi joka muutoksen jälkeen. Tämä säästää paljon aikaa. (Doyle 2014.)

GSAP-kirjastolla voi CSS-animaatioista poiketen animoida myös suhteellisilla arvoilla ja esimerkiksi kääntää elementtiä 30 astetta lisää sen nykyisestä arvosta (Doyle 2014).

GSAP mahdollistaa myös useita sisäkkäisiä animaatioita, jolloin kaikki sisäkkäiset animaatiot pysyvät tahdissa, vaikka pääanimaatiota muokattaisiin. Tämä rakenne helpottaa modulaarisen koodin luomista, jota on helpompi tuottaa ja ylläpitää. (Doyle 2014.)

CSS:llä luodut animaatiot ovat monimutkaisempia ja vaativat enemmän koodia, kuin suppeampi ja yksinkertaisempi GSAP:n syntaksi (Doyle 2014).

JavaScript-animaatioilla pystyy luomaan efektejä, jotka ovat pelkällä CSS-animaatiolla mahdottomia, kuten animoimaan kaariviivaa pitkin. JavaScript mahdollistaa myös erilaisten fysiikkamallien sekä monipuolisempien easing-vaihtoehtojen käyttämisen. (Doyle 2014.)

8 PILOTOINTI

Opinnäytetyön tavoitteena oli testata CasparCG:n toimintaa oikeassa tuotantoympäristössä. Tähän tarkoitukseen soveltuva tuotanto löytyi Lahden ammattikorkeakoulun (LAMK) kampusradio LiMu Radiolta.

LiMu Radion tekninen tuottaja Markus Juntunen oli tehnyt radiolle Die Echo Show -nimistä ohjelmasarjaa, joka keskittyi tietokonepelien maailmaan. Die Echo Show -ohjelmat tuotettiin muista radion ohjelmista poiketen videostriimeinä. LiMu Radio järjesti LAMK:n opiskelijoille Counter Strike Global Offensive (CS:GO) -turnauksen, jonka finaali oli ohjelmasarjan viides osa (DES #5).

Tapahtuma haluttiin toteuttaa teknisesti samaan tapaan kuin oikeat televisiotuotannot, joten se sopi hyvin CasparCG:n testaamiseen. Lähetys striimattiin Twitch-palvelun kautta maailmalle sekä lähetettiin myös Youtubeen.

8.1 Tekniikka

Pelitapahtuma järjestettiin Tekniikan laitoksen auditoriossa. Auditoriossa oli valmiina tietoverkot sekä 2 kappaletta videotykkejä ja televisio. Pelaajat toivat pelaamiseen tarvitut tietokoneet kotoaan. Haastattelupisteenä tapahtumassa toimi auditorion alapuolella oleva neuvottelutila, jossa oli myös yksi videotykki.

Tekniikan laitokselta saatiin käyttöön tilojen ja kiinteän tekniikan lisäksi kaksi Canon XF 100 -kameraa, joista toisessa oli chromatte-valo, Chromatte-kangas, kaksi litepanel LED -valaisinta, matrox mxo 2 mini videonkaappauslaite sekä yksi MacBook.

Limuradiolta käytettäväksi löytyivät mikrofonit, Behringer x32- ja s16 -äänimikserit sekä kannettava tietokone striimaamiseen ja ATEM Television Studion ohjaamiseen.

Musiikin laitokselta tapahtumaan saatiin PAR-valot dmx-ohjauksella sekä PA-järjestelmä. Muotoiluinstituutilta käytettäväksi saatiin ammattilaistason P2HD series -kamera ja BNC-kaapeleita sekä Blackmagic UltraStudio Express -videonkaappauslaite.

KUVA 1. Tekniikan laitoksen auditorio

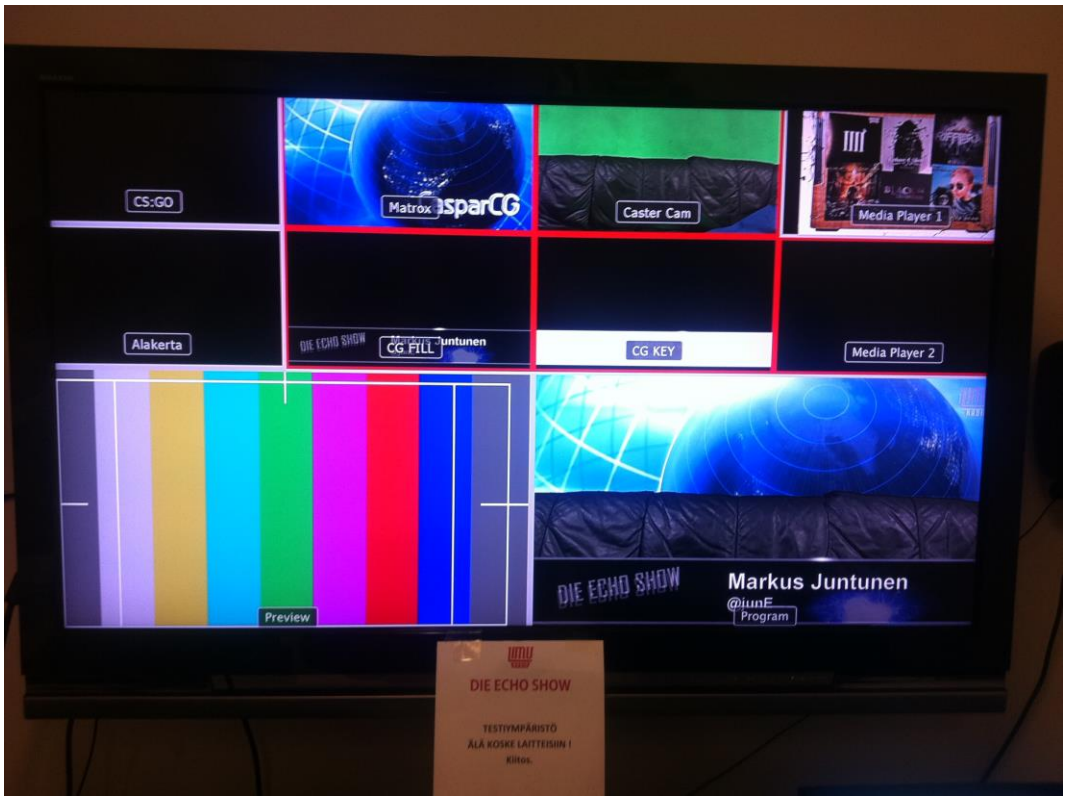
Kuvan 1 vasemmassa laidassa on näkyvissä fiilisprojektorin kuva ja sen alla miksaamiseen käytetty televisio. TV:n edessä ovat näkyvissä PAR-valot. Auditorion isommalle valkokankaalle heijastettiin lähtevän ohjelman kuva (PGM). Kankaan alapuolella oleva tila toimi tuotannossa tarkkaamona. Kuvan oikeassa laidassa on nähtävissä chromatte-valolla varustettu kamera ja kommentaattorien juontopiste.

Kuvassa 2 on nähtävissä auditorion alapuolinen tila sekä P2HD series -kamera. Tilan valkokankaalle heijastettiin myös lähtevän ohjelman kuva (PGM).

KUVA 2. Auditorion alapuolinen neuvottelutila (LiMu Radio 2015)

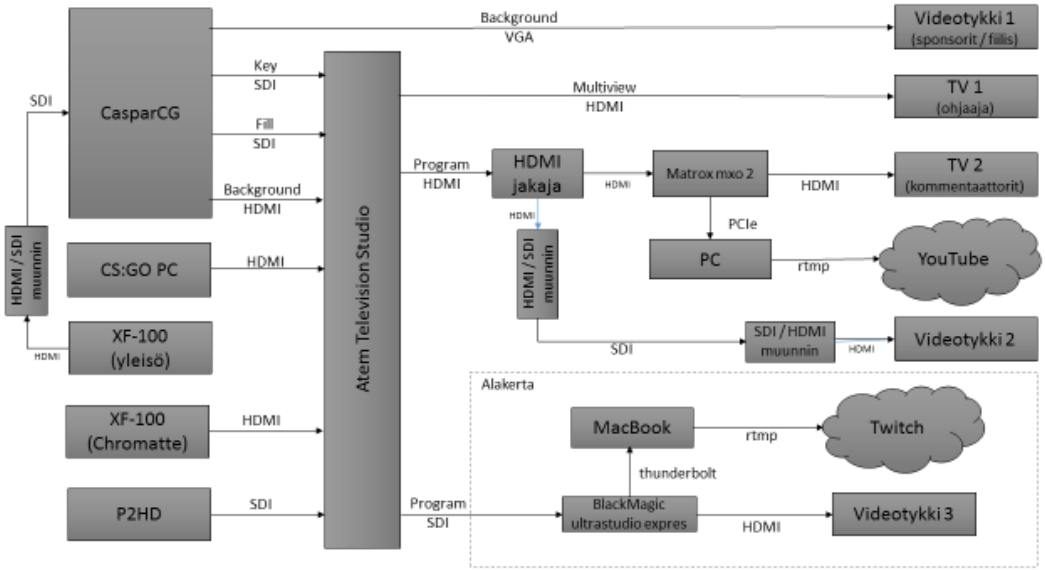
ATEM Television Studio kuvamikseri oli LiMu Radion teknisen tuottajan oma laite. Hän toi mukanaan myös erilaisia muuntimia ja kuvanjakajia.

ATEM Television Studio on Black Magic Designin valmistama kohtuuhintainen videomikseri, josta löytyy 4 HDMI-sisääntuloa ja 4 SDI-sisääntuloa. Laitteesta saa ulos lähtevän kuvan (PGM) niin HDMI- kuin SDI-versiona sekä miksaamiseen tarvittavan multiview-näkymän, niin SDI- kuin HDMI-versioina. Tämä näkymä on esitetty kuvassa 3.



KUVA 3. ATEM Television Studion multiview-näkymä

Grafiikkakone tapahtumaan saatiin pH kolme Oy:ltä. Tämä laite oli siis opinnäytetetyötä varten koottu demolaite. Tapahtuman videosignaalien kulku on esitetty kuviossa 18.



KUVIO 18. DES #5 -videosignaalit

Kuvassa 4 on nähtävissä, miten avainnus- ja täyttösignaalit yhdistyivät taustavideoon lähetyksessä. Lisäksi kuvassa on näkyvissä virtuaalinen taustanäyttö, joka toteutettiin chromatte-tekniikalla.

KUVA 4. Kahden nimen grafiikka (LiMu Radio 2015)

8.2 Demolaite

Demolaitetta lähdettiin rakentamaan CasparCG:n suositusten pohjalta. Suositeltu HP:n malli Z420 ei ollut enää saatavilla tukkurilta, mutta tämän opinnäytetyön tarpeisiin löydettiin hyvin kelpaava käytetty Z400-kone. Näytönohjaimeksi valikoitui Nvidia Quadro K2200 ja videokortiksi BlackMagic Designin DeckLink pro.

Näytönohjain haluttiin valita suositusten mukaan Nvidian Quadro-sarjasta, vaikka CasparCG kykenee toimimaan minkä tahansa OpenGL 3.0 -näytönohjaimen avulla. Näytönohjaimeen ei kuitenkaan haluttu käyttää turhan paljoa rahaa, sillä tällä koneella toteutettaisiin kevyitä grafiikoita ja vain testattaisiin tekniikkaa.

DeckLink pro oli suhteellisen uusi tuote Blackmagicin valikoimassa. Se oli julkistettu vain puoli vuotta ennen tapahtumaa. Kortti päätettiin valita,

koska se oli noin puolta edullisempi kuin suositeltu DeckLink Extreme, mutta kortin ominaisuuksiin kuului kuitenkin sisäinen keyer.

Vaikka CasparCG osaa generoida erilliset avainnus- ja täyttökanavat, niin kanavien täydellinen tahdissa pysyminen haluttiin varmistaa käyttämällä videokortin ominaisuuksia. DeckLink pro sisälsi myös mahdollisuuden käytää ulkoista tahdistussignaalia. DeckLink pron ominaisuuksiin kuului myös kahden videosignaalin kaappaaminen. Myöhemmin kuitenkin ilmeni, että toinen sisääntulo olisi käytettävissä vain stereoskooppisen 3D-signaalin kaappaamiseen. Vaikka sisääntulevien lähteiden määrä laski yhteen, oli DeckLink pro onnistunut ja kustannustehokas ratkaisu.

Mahdollisuus luoda avainnus- ja täyttösignaali CasparCG:n toimesta tarjoaa tulevaisuudessa mahdollisuuden karsia laitteiston kuluja entisestään. Esimerkiksi Assembly Winter 2016 -tapahtumassa toimintaympäristön kytkennät oli toteutettu siten.

Demolaite oli muuten täysin suositusten mukainen, mutta videolevylle ei toteutettu RAID-0-lomitusta. Kaksi SSD-levyä RAID-0-asetuksilla nopeuttaisi videoiden lukua huomattavasti. Tapahtumassa olikin havaittavissa hitautta alun introvideon lataamisessa.

DES #5 -tapahtuma oli ensimmäinen eSports-tuotanto, jossa pH kolme on ollut mukana. Oikeista urheilulähetyksistä poiketen, ottelun aikana ei tarvittu grafiikkaa, sillä pistetilanne ja muut tilastotiedot tulivat pelistä.

Jälkeenpäin huomattiin, että kokonaispistetuloksen olisi voinut lisätä peligrafiikoiden joukkoon. Tavallisesti peligrafiikoihin lisätään myös sponsoreiden logoja. DES #5 -tapahtumassa ei kuitenkaan lisätty pelikuvan päälle mitään muuta kuin LiMu radion logo, joka sekin toteutettiin videomikserillä.

Asiakassovelluksina tapahtumassa käytettiin SVT:n tarjoamaa sovellusta sekä karttavalintaan omaa itsetehtyä asiakassovellusta. Grafiikkapohjat toteutettiin yksinkertaisina html-sivuina, joiden taustalle ajettiin taustavideo

CasparCG:n avulla. SVT:n asiakassovelluksella näille videoille saatiin asetettua sisääntulo- ja poistumissiirtymät.

CasparCG:llä toteutettuja toimintoja tuotannossa olivat alaTG:t, kello, karttavalinta, taukografiikka, turnauskaavio, joukkue-esittely, videoklippien toistaminen sekä yksi kamerakuva. Lisäksi CasparCG ohjasi sisältöä juontajien takana sijaitsevalle virtuaaliselle näytölle sekä auditorion fiilisprojektorille.

Kuvassa 5 on toteutetuista toiminnoista monimutkaisin. Taukografiikassa hyödynnettiin grafiikkalaitteelle tulevaa videokuvaa, joka animoitiin pienemmäksi luukuksi. Videon alta paljastui grafiikkapohja, jossa oli tieto seuraavaksi pelattavasta kartasta ja alaspäin laskeva kello. Takimmaisena olevaa taustavideota toistettiin silmukkana ja tämän videotason päällä toistettiin LiMu Radion mainosvideota.

KUVA 5. Des #5 -taukografiikka (LiMu Radio 2015)

Demokoneesta oli siis käytössä 2 ulostuloa näytönohjaimelta ja 2 ulostuloa videokortilta. Lisäksi demolaitetta kuormitettiin käyttämällä sitä grafiikoiden ulosajoon, mikä lisäsi näytönohjaimen kuormitusta yhden ulostulon verran lisää. Tämä ulostulo oli siis Windowsin työpöytää ja asiakassovellusta varten.

Demolaite suoriutui tästä kuormasta hyvin. Tosin Screen consumerin avulla toteutetuissa kanavissa oli havaittavissa pientä nykimistä, mutta tuo

nykiminen syntyi aina, vaikka kuorma vähennettiin yhteen ulostuloon. Kyseessä oli siis jokin asetus/ajuri ongelma, joka jäi selvittämättä. Tärkein ulostulo videokortin kautta kuitenkin toimi moitteetta koko ajan. Tuo oli ainut kytkentä, jota oli alun perin tarkoitus tukea. Myös jatkossa DeckLinkin tarjoamat SDI-signaalit ovat niitä, joita asiakkaille tultaisiin tarjoamaan.

8.3 Asiakasohjelmisto

Karttavalitsin toteutettiin Windows Forms -ohjelmana. Ohjelman luomisessa käytettiin C#-ohjelmointikieltä ja hyödynnettiin Microsoftin .NET-sovelluskehystä. Ohjelmointiympäristönä toimi Microsoftin Visual Studio.

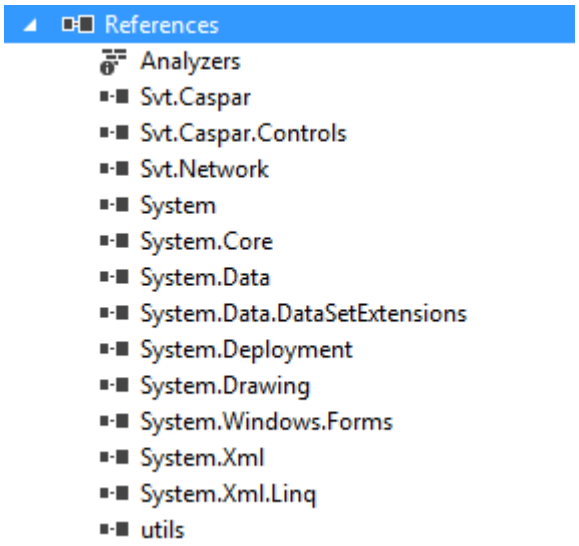
KUVA 6. Karttavalitsimen käyttöliittymä

Kuvassa 6 on näkyvissä ohjelman käyttöliittymä, jossa pystyi estämään kaksi karttaa ja valitsemaan kaksi. Tämän jälkeen ohjelma valitsi kolmannen kartan sattumanvaraisesti.

Illuusio Commodore 64 -ohjelmasta luotiin kuvaamalla kannettavan tietokoneen näyttöä ja välittämällä tämä kuva komposiittivideona suoraan Commodore 64:n näyttöön. Tämä on nähtävissä kuvassa 7.

KUVA 7. Karttavalitsin Commodore 64:n ruudulla

Kommunikointi palvelimeen tapahtui hyödyntämällä SVT.Caspar.dll-kirjastosta löytyvää CasparDevice-luokkaa. CasparCG:n dll-kirjastojen käyttöönotto C#-asiakasohjelmassa tapahtui lisäämällä halutut kirjastot Visual Studion references-osioon, kuten kuviossa 19 on esitetty.



KUVIO 19. Visual Studion references-näkymä

CasparDevice-luokan connect-funktiolla sai yhteyden palvelimeen ja luomalla ConnectionStatusChanged-tapahtumankäsittelijän pystyi käyttöliittymässä näyttämään ilmoituksen jos yhteys katkeaa.

Yhteyshän menetettiin tuotannossa ja karttavalitsin ei toiminut aivan kuten testatessa. Pelkän ilmoituksen sijaan olisi pitänyt ohjelmoida myös jonkin näköinen uudelleenyhdistäminen.

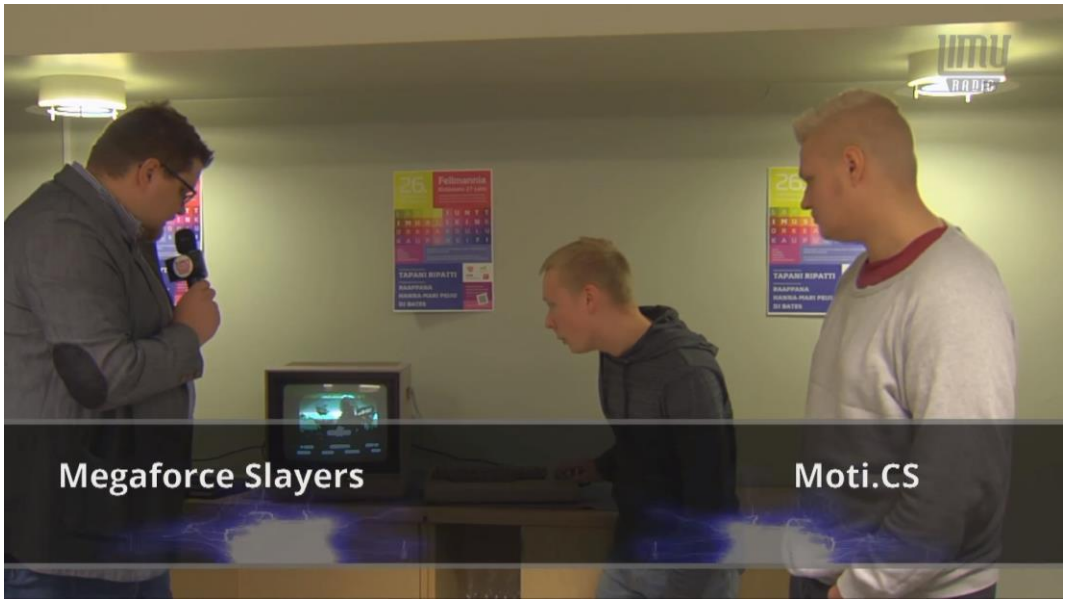
AMCP-protokollan mukaisia komentoja palvelimelle lähetettiin SendString-funktion avulla. Lähetettävän datan muotoiluun toteutettiin oma createMsg-apufunktio, joka on esitetty kuviossa 20. Näiden funktioiden hyödyntäminen ja AMCP-protokollan mukaisten käskyjen välitys CasparCG:lle on esitetty kuviossa 21.

```
/// <summary>
/// Encapsulates the message to xml fragment that can be sent to casparCG server.
/// </summary>
/// <param name="id">The id of the node in casparCG template</param>
/// <param name="txt">The content of the node in casparCG template</param>
/// <returns>xml fragment as string</returns>
public static string createMsg(string id, string txt) {

    string s = "";
    System.Xml.XmlWriterSettings XmlSettings = new System.Xml.XmlWriterSettings();
    XmlSettings.OmitXmlDeclaration = true;
    XmlSettings.NewLineChars = "";
    StringBuilder sb = new StringBuilder(s);
    System.Xml.XmlWriter xmlWriter = System.Xml.XmlWriter.Create(sb, XmlSettings);
    xmlWriter.WriteStartElement("templateData");
    xmlWriter.WriteStartElement("componentData");
    xmlWriter.WriteAttributeString("id", id);
    xmlWriter.WriteStartElement("data");
    xmlWriter.WriteAttributeString("id", "text");
    xmlWriter.WriteAttributeString("value", txt);
    xmlWriter.WriteEndElement();
    xmlWriter.WriteEndElement();
    xmlWriter.WriteEndElement();
    xmlWriter.Flush();
    xmlWriter.Close();

    return "\"" + sb.ToString().Replace("\"", "\\\"") + "\" \\r\\n";
}
```

KUVIO 20. Apufunktio viestien muodostamiseen

```
string s = Communication.createMsg("Team" + banned.ToString() + "Ban", (sender as Button).Text.Substring(2));
casparCG.SendString("CG 1-20 UPDATE 1 " + s);
casparCG.SendString("CG 1-20 INVOKE 1 show" + "Team" + banned.ToString() + "Ban");
```

KUVIO 21. Käskyjen välittäminen CasparCG:lle



KUVA 8. Karttavalitsin lähetyksessä (LiMu Radio 2015)

KUVIO 22. Lähetyksestä puuttumaan jäänyt osa

Koska yhteys karttavalitsimen ja CasparCG-palvelimen välillä menetettiin, lähetykseen saatiin vain joukkueiden nimet ja taustavideo, kuten kuvasta 8 voi nähdä. Kuviossa 22 on nähtävissä lähetyksestä puuttumaan jäänyt informaatio, eli valitut ja estetyt kartat. Ennen lähetystä tehdyissä testeissä tämä informaatio muodostui asiakasohjelmiston ruudulla ja lähetysgrafiikassa yhtä aikaa.

9 TUOTTEISTAMINEN

Jotta CasparCG:llä tuotetut televisiografiikat voitaisiin ottaa pH kolmen palveluvalikoimaan, täytyi grafiikkapohjien luontia testata vielä DES #5 -tapahtumaa enemmän. Koska CasparCG itsessään on ilmainen, ei siitä olisi mahdollista laskuttaa asiakkaitakaan. Sen sijaan grafiikkapohjien luomista ja grafiikoiden operointia voitaisiin tarjota palveluna.

Grafiikkapohjien luomisen lisäksi omien käyttöliittymien luominen operoinnin helpottamiseksi tarjoaa kilpailuetua ja syyn asiakkaalle valita pH kolme toimittamaan grafiikat. Tilaaja ei välttämättä ikinä näe käyttöliittymiä, mutta ne ilmenevät asiakkaalle toiminnan nopeutumisena ja mahdollistavat pienemmän vaatimustason operaattoreille.

CasparCG:n videontoisto-ominaisuudet tarjoavat myös mahdollisuuden korvata pH kolmen oman kuvan- ja äänenhallintajärjestelmän.

9.1 HTML-grafiikkapohja

pH kolmen henkilöstöllä oli enemmän kokemusta html-sivujen luomisesta kuin Flash-grafiikan luomisesta, joten HTML-pohjaiset grafiikkapohjat sopisivat paremmin yrityksen työnkulkuun.

Lisäksi pH kolme Oy halusi päästä hyötymään kokonaisten grafiikkaprojektien laajuisista tyylitiedostoista, jotka nopeuttaisivat projektien uudelleenkäyttämistä ja brändäämistä eri asiakkaille. Yrityksen tarpeisiin toteutettiin siis vielä sopiva mallipohja.

HTML-grafiikkapohjan teksti oli helppo asemoida ja tyylittää HTML 5:n ja CSS 3:n avulla. Grafiikkapohjan elementtien animoimista kokeiltiin CSS 3 -animaatioilla ja GSAP JavaScript -kirjaston avulla.

Tavallisella tietokoneella testatessa eri CSS 3 -animaatiot eivät pysyneet tahdissa. Demokoneella testatessa tämäkin tapa näytti silmämääräisesti tarkasteltuna toimivan. CasparCG-palvelin antoi kuitenkin varoituksen: *window.requestAnimationFrame() never called. Animations might have*

been laggy. Palvelin siis varoitti siitä, että CEF 3 -selaimen animaatioiden käyttämää ruudunpäivitysnopeutta ei ole tahdistettu CasparCG-palvelimen käyttämään tahtiin. Tahdistaminen on mahdollista vain käytettäessä JavaScript-animaatioita. Sama animaatio toteutettiin siis vielä GSAP-kirjaston avulla, jolloin varoitus poistui.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
    <head>
        <title>HiTech</title>
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
        <script type="text/javascript" src="jquery.js"></script>
        <script type="text/javascript" src="GSAP/TweenMax.min.js"></script>
        <script src="GSAP/jquery.gsap.min.js"></script>
        <script type="text/javascript" src="CasparCG.js"></script>
        <style>
        <script>
        function show(){
        $("#inClip").show();
        $("#outClip").hide();
        document.getElementById("inClip").play();
        TweenMax.to("#f0", 0.3, {opacity:1, repeat:3, repeatDelay:0.2, delay: 0.3});
        TweenMax.to("#f1", 0.3, {opacity:1, repeat:3, repeatDelay:0.2, delay: 0.6});
        }
        function hide(){
        $("#outClip").show();
        $("#inClip").hide();
        document.getElementById("outClip").play();
        TweenMax.to("#f0", 0.2, {opacity:0, repeat:3, repeatDelay:0.2, delay: 0.3});
        TweenMax.to("#f1", 0.2, {opacity:0, repeat:3, repeatDelay:0.2});
        }
        </script>
    </head>
    <body>
        <span id="f0">CS</span>
        <span id="f1">Etunimi Sukunimi</span>
        <video id="inClip">
        <source src="clips/in.webm" type="video/webm">
        </video>
        <video id="outClip">
        <source src="clips/out.webm" poster="clips/poster.png" type="video/webm">
        </video>
    </body>
</html>
```

KUVIO 23. Esimerkki GSAP-kirjaston avulla toteutetusta grafiikkapohjasta

Grafiikkapohjan tausta luotiin ffmpeg-ohjelman avulla png-sekvenssistä. Sekvenssi muunnettiin ffmpeg:n avulla webm-videoksi, joka on videoformaatti, joka tukee läpinäkyvyyttä ja toimii CEF 3 -selaimessa. Valmis mallipohja, jossa kaikille grafiikkapohjille yhteiset asiat ovat koottuina head-osioon, on esitetty kuviossa 23.

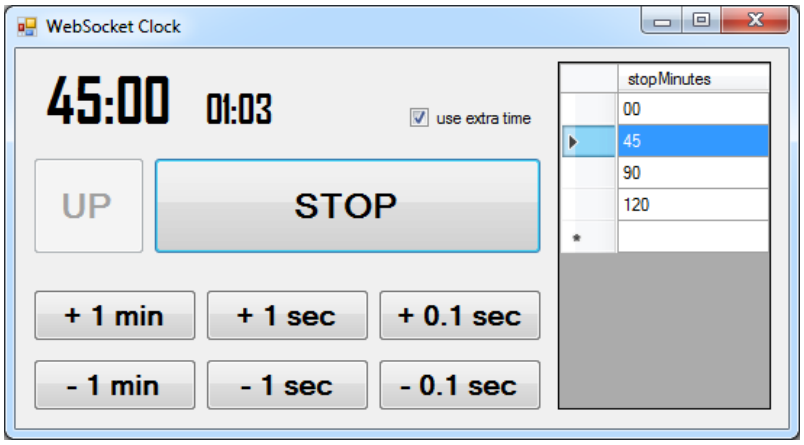
9.2 Pelikello

Pelikello on tärkein reaaliaikainen elementti urheilulähetyksissä, joten haluttiin vielä toteuttaa asiakasohjelma, jolla kello olisi helppo lisätä CasparCG:llä toteutettuihin tv-grafiikoihin. Kellolle oli pH kolmen taholta vaatimuksina se, että kelloa olisi mahdollista ohjata eri koneelta kuin muita grafiikoita. Pelikellon olemassaolo on myös ollut tärkeässä roolissa pH

kolmen omassa kuvan ja äänenhallintajärjestelmässä, eikä tätä järjestelmää voitaisi korvata CasparCG-pohjaisella ratkaisulla ilman pelikelloa.

Pelikellon toteuttamiseen löytyi CasparCG:n kotisivuilta esimerkkiprojektiksi World Cup 2010 -asiakasohjelmisto. Tässä ohjelmassa kello oli toteutettu lähettämällä palvelimelle grafiikkapohjan päivityskäsky sekunnin välein.

Esimerkkiprojektissa oli kuitenkin pari merkittävää puutetta. Ohjelmassa ei ollut kellolle monitorointia, sillä tällä ohjelmalla ohjattiin muitakin grafiikoita. Kellon näkisi siis ulosmenevästä signaalista, mutta pH kolmen tapauksessa kellolle haluttiin mahdollisuus ohjata sitä kentän laidalta, ei pelkästään ulkotuotantoautosta. Lisäksi tässä ratkaisussa asiakasohjelman pitää tietää, mille grafiikkapohjalle kelloa ollaan lähettämässä.



KUVIO 24. Websocket-kellon käyttöliittymä

Ohjelma päätettiin tehdä hyödyntämällä websocket-tekniikkaa ja html-grafiikkapohjia. Kellosignaali kulkisi tällä tavoin suoraan grafiikkapohjaan ja grafiikkapohjia voisi olla useita. Websocketilla toteutetun kellon voi myös jakaa esimerkiksi toimitsijoille, sillä sitä on mahdollista seurata myös selaimella. Kuviossa 24 on näkyvissä ohjelman käyttöliittymä, jonka vasemmasta yläkulmasta löytyy mahdollisuus monitoroida kelloa.

Kuviossa 25 on ohjelmaan toteutettu kello-olio. Kellon ominaisuuksina on kellon nimi, tieto siitä, onko kello käynnissä, kellon aika formatoituna tavallisesti käytettyyn muotoon ja TimeSpan-muotoinen aika. Luokkaan toteutettiin myös toJSON-funktio, jolla olion saisi serialisoitua JSON-muotoon. Tuo JSON-muotoinen esitys lähetetään ajastetusti kaikille sen haluaville grafiikkapohjille.

```
class clockObject
{
    public string Name { get; }
    public bool clockIsRunning { get; set;}
    public string FormattedTime { get; set; }
    public TimeSpan time { get; set; }

    public clockObject(string Name)
    {
        this.Name = Name;
        this.clockIsRunning = false;
        this.FormattedTime = "00:00";
        this.time = new TimeSpan(0, 0, 0, 0, 0);
    }

    public string toJSON() {
        var json = new JavaScriptSerializer().Serialize(this);
        return json;
    }

}
```

KUVIO 25. Kello-olio C#-koodissa

Jokaiseen grafiikkapohjaan, joka haluaisi näyttää kellon, tarvitsisi vain lisätä clock.js-tiedosto. Tämä scripti on vastuussa kellosovellukseen yhdistämisestä ja viestien vastaanottamisesta. Viestien vastaanottaminen on esitetty kuviossa 26. Scripti korvaa grafiikkapohjasta elementin, jolle on asetettu ID:n arvoksi kellon nimi. Näin grafiikkapohjan tekijälle riittää, että osaa luoda HTML-merkkauskielellä grafiikkapohjan ja tyylittää sen CSS:n avulla. Scripti lisää tai poistaa elementille myös clockIsStopped- tai clockIsRunning-luokan, riippuen siitä, onko kello pysähdyksissä vai ei.

Kuviossa 27 on esitetty html-grafiikkapohja ja kuviossa 28 tyylitiedosto, jossa pysähdyksissä oleva kello piilotetaan ja vain käynnissä oleva kello näytetään. Kuviossa 29 on nähtävillä grafiikka selaimella testattuna.

```
// when data is comming from the server, this metod is called
ws.onmessage = function (evt) {
  var data = $.parseJSON(evt.data);

  //check what kind of message was recieved
  switch(data.Name){
      case "mainTime":
      case "extraTime":

        var postFix = "";

        if($( "#"+data.Name+"LastChild" ).length){
            //postFix = $( "#"+data.Name+"LastChild" ).html();
            postFix = document.getElementById(data.Name+"LastChild").innerHTML;
        }

        if(data.FormattedTime){
          if($( "#"+data.Name ).hasClass( "fixed-pitch" )){
              $("#"+data.Name).html( encapsulateCharactersInSpanElements(data.FormattedTime) +postFix );
          }
          else{
            $("#"+data.Name).html( data.FormattedTime + postFix );
          }
        }

          if(data.clockIsRunning){
            $( "#"+data.Name ).removeClass( "clockIsStopped" );
            $( "#"+data.Name ).addClass( "clockIsRunning" );
          }
          else{
            $( "#"+data.Name ).removeClass( "clockIsRunning" );
            $( "#"+data.Name ).addClass( "clockIsStopped" );
          }

      break;
      default:
        console.log("received data has no name or the name is unknown");
      break;
  }

};
```

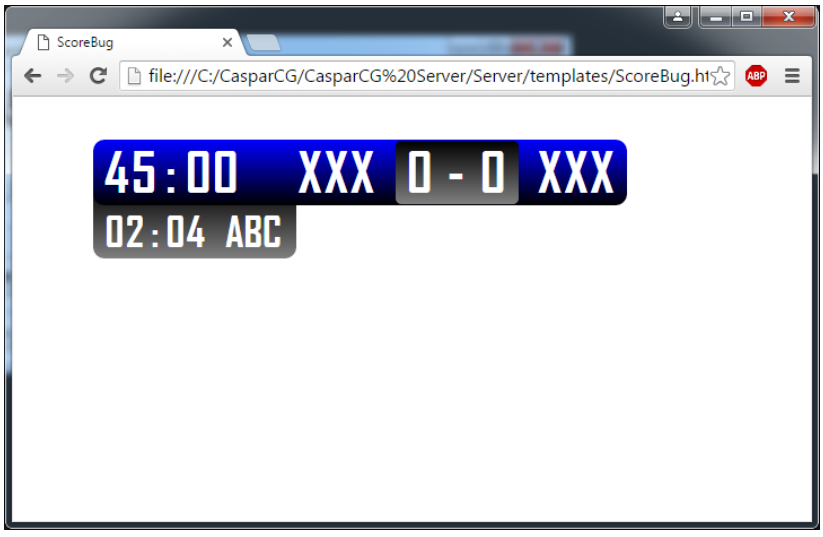
KUVIO 26. Viestien vastaanotto clock.js-tiedostossa

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
    <head>
        <title>ScoreBug</title>
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
        <script type="text/javascript" src="jquery.js"></script>
        <script type="text/javascript" src="clock.js"></script>
        <script type="text/javascript" src="CasparCG.js"></script>
        <link rel="stylesheet" type="text/css" href="gfxStyle.css">
        <script>
        function show(){
        $("#scoreBug").css("max-height","60px");
        $("#scoreBug").css("max-width","600px");
        $("body").css("opacity","1");
        $("#extraTime.clockIsRunning").hide().delay( 800 ).fadeIn( 200 );
        }
        function hide(){
        $("body").css("opacity","0");
        }
        </script>
    </head>
    <body>
        <div id="scoreBug">
            <span id="mainTime" class="fixed-pitch">00:00</span>
            <span id="homeTeamShort">XXX</span>
            <span id="score">0 - 0</span>
            <span id="awayTeamShort">XXX</span>
        </div>
        <div id="scoreBugExtra">
            <span id="extraTime" class="fixed-pitch">00:00</span>
            <div id="extraTimeLastChild">
            <span id="additionalTime" class="extraTimeChild">ABC</span>
            </div>
        </div>
    </body>
</html>
```

KUVIO 27. CasparCG-grafiikkapohja

```
#extraTime{
    transition: height 1s  ease-in, padding-top 0.5s,opacity 0.5s;
    position: absolute;
    top: 10%;
    left: 10%;
    margin-top: 40px;
    background: linear-gradient(black,gray);
    color: white;
    font-family: "Agency FB";
    font-size: 36px;
    font-weight: bold;
    padding: 10px;
    padding-top: 20px;
    padding-bottom: 5px;
    border-bottom-left-radius: 10px;
    border-bottom-right-radius: 10px;
    z-index: -1;
    opacity: 0;
    width: 160px;
}

#extraTime > span{
width: 18px;
display: inline-block;
text-align: center;
}

#extraTime.clockIsStopped{
    height: 0px;
    opacity: 0;
    padding-top: 0px;
}

#extraTime.clockIsRunning{
    height: 40px;
    opacity: 1;
    padding-top: 20px;
}
```

KUVIO 28. Tyylitiedosto



KUVIO 29. Valmis grafiikka selaimella testattuna

10 YHTEENVETO JA JOHTOPÄÄTÖKSET

Opinnäytetyön tavoitteena oli tutustua avoimen lähdekoodin CasparCG-ohjelmistoon ja testata sen toimivuutta grafiikoiden tuottamisessa suoraan tv-lähetykseen. Tavoitteena oli myös tutustua palvelimen rajapintoihin ja toteuttaa oma asiakassovellus tuotannon avuksi.

Opinnäytetyötä tehdessä saatiin kerrytettyä yrityksen kokemus- ja tietopohjaa sekä palvelimen että SVT:n tarjoaman asiakassovelluksen käyttämisestä. Samalla opittiin myös hyödyntämään CasparCG-yhteisön tarjoamia työkaluja grafiikkapohjien luomiseksi. Myös yhteisön tarjoamaa .NET-kirjastoa hyödynnettiin oman asiakassovelluksen luomiseksi.

Isoimmat haasteet grafiikoiden luomisessa syntyivät CasparCG:n ulkopuolisten ohjelmien, kuten Adoben Flash CC:n, käyttämisestä. Html-grafiikkapohjissa kuitenkin ilmeni vielä hieman rosoisuutta tekstien reunoilla. Pelikellon toteuttaminen myös paljasti noin 0,5 sekunnin viiveen palvelimen viestien vastaanottamisessa ja itse grafiikan piirtämisessä.

Tekstien rosoisuutta voitaisiin koettaa korjata päivittämällä CasparCG:n käyttämän selaimen versiota. Ratkaisuja voidaan myös jäädä odottamaan yhteisöstä. GSAP-animaatiokirjaston hyödyntämistä tulisi tulevaisuudessa tutkia ja opetella vielä lisää, jotta asiakkaille tarjottaviin grafiikoihin saataisiin tarjottua helpommin visuaalisesti näyttäviä animaatioita.

Tekniikka on nyt kuitenkin testattu toimivaksi ja kilpailukykyiseksi verrattuna aiemmin pH kolmella käytössä olleisiin ratkaisuihin. Vaikka CasparCG-pohjaisista ratkaisuista ei koskaan tulisikaan yrityksen isoimpia ja tärkeimpiä tuotteita, tarjoaa tämä avoimen lähdekoodin vaihtoehto hyvän pohjan, jonka avulla pienenkin yrityksen on täysin mahdollista tarjota ratkaisuja tv-grafiikan tuottamiseksi suoriin lähetyksiin, olematta kuitenkaan riippuvainen grafiikkalaitevalmistajien kanssa neuvoteltujen lisenssien hinnoista. CasparCG:n avulla pH kolmen on mahdollista saavuttaa aiempaa suurempi asiakaskunta, jota voidaan nyt palvella kaupallisten grafiikkalaiteratkaisuden avulla, mutta myös ilman niitä.

LÄHTEET

About Ffmpeg 2016. [viitattu 10.5.2016]. Saatavissa: http://ffmpeg.org/about.html

CasparCG Wiki 2016a. CasparCG 2.0 AMCP Protocol [viitattu 16.2.2016]. Saatavissa: http://casparcg.com/wiki/CasparCG_2.0_AMCP_Protocol

CasparCG Wiki 2016b. CasparCG Server [viitattu 16.2.2016]. Saatavissa: http://casparcg.com/wiki/CasparCG_Server

CasparCG Wiki 2016c. Content / Media [viitattu 16.2.2016]. Saatavissa: http://casparcg.com/wiki/Content_/_Media

CG Director 2011. Quick-tip: Straight alpha vs premultiplied alpha [viitattu 16.2.2016]. Saatavissa: http://www.cgdirector.com/quick-tip-straight-alpha-vs-premultiplied-alpha/

Didikunz 2014. Re: CasparCG Server 2.0.7 released [viitattu 16.2.2016]. Saatavissa: http://www.CasparCG.com/forum/viewtopic.php?f=5&t=2857#p18992

Doyle, J. 2014. Myth Busting: CSS Animations vs. JavaScript [viitattu 16.2.2016]. Saatavissa: https://css-tricks.com/myth-busting-css-animations-vs-JavaScript/

Hummelstrand, J. 2012. From elections to rocket launches: CasparCG - how a small piece of software is changing live productions [viitattu 16.2.2016]. Saatavissa: https://tech.ebu.ch/docs/techreview/trev_2012-Q4_CasparCG_Hummelstrand.pdf

Hummelstrand, J. 2014. CasparCG Server 2.0.7 beta 2 released [viitattu 16.2.2016]. Saatavissa: http://www.casparcg.com/forum/viewtopic.php?f=9&t=1978

Karlsson, P. 2014. CasparCG Server 2.0.7 released [viitattu 16.2.2016]. Saatavissa: http://www.casparcg.com/forum/viewtopic.php?f=5&t=2857

Lee, D. 2004. Television Technical Theory: Unplugged Version 5.0 [viitattu 16.2.2016]. Saatavissa: http://www.danalee.ca/ttt/Whats_In_A_TV_Station.htm

LiMu Radio 2015. DIE ECHO SHOW #5. [viitattu 29.3.2016]. Saatavissa: https://www.youtube.com/watch?v=W5PcO_dyYVU

Millerson, G. & Owens, J. 2009. Television production. 14. painos. Oxford: Focal Press.

Scully, T. 2013. 7 Reasons Why HD-SDI is Better Than HDMI [viitattu 16.2.2016]. Saatavissa: https://churchm.ag/hdmi-vs-hd-sdi/

Tektronix, Inc. 1997. A Guide to Digital Television Systems and Measurements [viitattu 16.2.2016]. Saatavissa: http://www.tek.com/document/primer/guide-digital-television-systems-and-measurements

Weston, B. 2009. The basics of video keying [viitattu 15.2.2016]. Saatavissa: https://www.renewedvision.com/downloads/tfwm_keying_article.pdf

Wright, M. 2002. The Open Sound Control 1.0 Specification Version 1.0 [viitattu 29.3.2016]. Saatavissa: http://opensoundcontrol.org/spec-1_0