TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

RIA-KEHITYS RUBY RSENCE RIA -OHJELMISTOKEHYKSELLÄ

Työn tekijä: Niko Hälvä
Työn ohjaajat: Simo Silander
Ilpo Kuivanen

Työ hyväksytty: ___. ___. 2008

Lehtori Simo Silander

Metropolia

ALKULAUSE

Tämä insinöörityö ja siihen liittyvät sovelluslaajennus tehtiin kesän 2010 kuluessa. Erityiskiitokset kuuluvat RSence:n kehittäjille Juha-Jarmo Heinoselle sekä Toni Nurmiselle. Kiitokset myös työn ohjaajille Simo Silanderille ja Ilpo Kuivaselle.

Helsingissä 5.10.2010

Niko Hälvä

Metropolia

TIIVISTELMÄ

Työn tekijä: Niko Hälvä

Työn nimi: RIA-Kehitys Ruby Rsence RIA -ohjelmistokehyksellä

Päivämäärä: 11.09.2010 **Sivumäärä:** 34 s. + 1 liitettä

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn ohjaaja: Lehtori Simo Silander

Työn ohjaaja: Lehtori Ilpo Kuivanen

Tämän insinöörityön tavoitteena on tutustua Internetin kehitykseen viime vuosina ja niihin puutteisiin, joita Ajax ja sen kilpailevat RIA-tekniikat Adobe Flex, Microsoft Silverlight ja Java ovat yrittäneet paikata. Insinöörityössä toteutetaan RIA-sovellus käyttäen Ruby-ohjelmointikielen RSence RIA –ohjelmistokehystä sekä tarkastellaan RSence:n sopivuutta rikkaiden Internet-sovellusten tekemisessä.

RSence on RIA-ohjelmistokehys, joka on suunniteltu nopeasti reagoivien graafisten käyttöliittymien toteuttamiseen. RSence edustaa Ajax RIA –tekniikkaa, ja se koostuu Ruby-ohjelmointikielellä toteutetusta palvelinpuolen ohjelmistokehyksestä ja asiakaspuolen optimoidusta JavaScript-ohjelmistokehyksestä. Usein Ajax ja varsinkin RIA-kehitys on teknisesti haastavaa. RSencen lupauksiin kuuluu, että RIA-kehitys sen avulla olisi helpompaa kuin kilpailevilla RIA-tekniikoilla.

RSence:llä toteuteutettavan sovelluslaajennuksen tarkemmalla läpikäynnillä mahdollistetaan tämän insinöörityön käyttäminen myös oppaana RSence-sovelluslaajennusten tekemiseen.

Avainsanat: Ajax, Rikkaat Internet-sovellukset, RSence RIA-ohjelmistokehys, Ruby

Metropolia

ABSTRACT

Name: Niko Hälvä

Title: RIA development with Ruby RSence Framework

Date: 11.09.2010 **Number of pages:** 34 pages + 1 attachment

Department: Information Technology **Study Programme:** Software Engineering

Instructor: Lecturer Simo Silander

Instructor: Lecturer Ilpo Kuivanen

The aim of this thesis is to learn about Internet development in recent years and especially the rich internet applications (RIA). Ajax and its competing RIA Technologies Adobe Flex, Microsoft Silverlight, and Java have been trying to fill the gap between traditional desktop software and static web software.

Ruby RSence is a RIA framework that is designed to make quickly responding graphic interfaces which work on web browsers.

This thesis can be used as a tutorial for RSence plugin development because of the extensive example which is provided in the end of thesis.

Keywords: Ajax, Rich Internet Applications, RSence RIA-framework, Ruby

SISÄLLYS

1 JOHDANTO

Internet syntyi 29. lokakuuta vuonna 1969, kun Kalifornian yliopiston ja Standfordin tutkimusinstituutin välillä lähetettiin ARPAnet-tietoverkon kautta ensimmäinen viesti: ”lo”. Tarkoituksena oli lähettää viesti ”log”, mutta Standfordin isäntäkone kaatui ennen koko viestin lähettämistä. [1.]

Näistä lähtökohdista Internet on kasvanut viimeisen 15 vuoden aikana kaikkien saataville. Kova kilpailu käyttäjistä on pakottanut ohjelmistotuottajat keksimään uusia tapoja luoda laadukkaita palveluita nykyisten standardien puitteissa. Yksi näistä menetelmistä on *Asynchronous JavaScript And XML* (Ajax). Ajaxin avulla voidaan luoda vuorovaikutteisia sovelluksia ja jopa työpöytäohjelmia muistuttavia rikkaita Internet-sovellutuksia (*Rich Internet Applications,* RIA), jotka toimivat verkkoselaimissa ilman erityisiä lisukkeita (*plugin)*. RIA-sovellusten kehittäminen on teknisesti haastavampaa ja siihen kuluu enemmän aikaa kuin perinteisillä Web-tekniikoilla. Tämän johdosta kehittäjien avuksi on luotu useita aikaa ja vaivaa säätäviä kilpailevia tekniikoita.

Tämän insinöörityön tavoitteena on tutustua Internetin kehitykseen viime vuosina ja niihin puutteisiin, joita Ajax ja sen kilpailevat RIA-tekniikat Adobe Flex, Microsoft Silverlight ja Java ovat yrittäneet paikata. Insinöörityössä toteutetaan RIA-sovellus käyttäen Ruby-ohjelmointikielen RSence RIA -ohjelmistokehystä sekä tarkastellaan RSence:n sopivuutta rikkaiden Internet-sovellusten tekemisessä.

RSence on RIA-ohjelmistokehys, joka on suunniteltu nopeasti reagoivien graafisten käyttöliittymien toteuttamiseen. RSence edustaa Ajax RIA-tekniikkaa, ja se koostuu Ruby-ohjelmointikielellä toteutetusta palvelinpuolen ohjelmistokehyksestä ja asiakaspuolen optimoidusta JavaScript-ohjelmistokehyksestä.[2.] Usein Ajax ja varsinkin RIA-kehitys on teknisesti haastavaa. RSencen lupauksiin kuuluu, että RIA-kehitys sen avulla olisi helpompaa kuin kilpailevilla RIA-tekniikoilla.

RSence:llä toteutettavan sovelluslaajennuksen tarkemmalla läpikäynnillä mahdollistetaan tämän insinöörityön käyttäminen myös oppaana RSence-sovelluslaajennusten tekemiseen.

Insinöörityön lopuksi pohditaan, ovatko nykyiset, tekniikan terävintä kärkeä edustavat RIA-tekniikat loppujen lopuksi jo vanhentuneita, sillä lähitulevaisuudessa ilmestyvä HTML5-standardi sisältää suuren osan nykyisten välineiden toiminnallisuudesta. Varsinkin Adobe Flexin moottorina toimiva Flash on joutunut viime aikoina arvostelun kohteeksi. Se voi nopeuttaa nykyisten RIA-tekniikoiden väistymistä HTML5-standardin tieltä.

2 INTERNET EILEN JA TÄNÄÄN

Tässä luvussa kerrotaan Internetin standardien muutoksista vuosien saatossa sekä näiden muutosten tuomista mahdollisuuksista.

2.1 Internet eilen

Tim Berners-Lee ja Rober Caillau muotoilivat ensimmäisen version HTML:stä, *Hypertext Markup Language,* vuonna 1989. HTML tunnetaan erityisesti kielenä, josta webbisivut rakentuvat. HTML:n kehityksestä vastasi välillä IETF, *The Internet Engineering Task Force,* ja välillä verkkoselainten valmistajat, jotka lisäsivät HTML:ään omia lisäyksiään, jolloin osa HTML-tageista ei ollut yhteensopivia selaimesta toiseen. Lopulta standardointi päätyi W3C:lle, *World Wide Web Consortium*, joka julkaisi HTML 3.2:n tammikuussa 1997. [3.]

Vuosien saatossa tietokoneet kehittyivät ja niiden rinnalla myös Internet, jonka räjähdysmäinen käyttäjämäärien kasvu alkoi graafisten verkkoselaimien ilmestyttyä 90-luvun puolivälin tienoilla. Mosaic Netscape 0.9-verkkoselain ilmestyi vuonna 1994, ja vuotta myöhemmin Microsoft julkaisi ensimmäisen Internet Explorerin.

Näihin aikoihin verkkosivut olivat staattisia tekstinomaisia ilman mahdollisuutta interaktiiviseen toiminnallisuuteen. Tähän puutteeseen Netscape Communications julkaisi JavaScript-komentosarjakielen vuonna 1995 Netscape 2.0 -verkkoselaimen yhteydessä. Microsoft vastasi tähän myöhemmin omalla JScriptillä, joka kilpaili suoraan JavaScriptin kanssa. JavaScript ja JScript tarjosivat kehittäjille mahdollisuuden luoda verkkosivuille dynaamisia elementtejä, kuten esimerkiksi aukeavia menuja, jotka olivat tuttuja työpöytäohjelmista.

JavaScript ja JScript aiheuttivat huomattavia yhteensopivuusongelmia, joiden johdosta Netscape Communications kääntyi Euroopan tietokonevalmistajien yhdistyksen, *European Computer Manufacturer Assosication* (ECMA), puoleen, jotta ECMA ottaisi vastuun JavaScriptin standardisoinnista. ECMA julkaisi standardoidun komentosarjakielen ECMAScriptin vuonna 1997. [4, s. 4-6.]

Macromedia julkaisi ensimmäisen version Flash-verkkoselainlaajennuksesta vuonna 1996. Sen avulla pystyttiin tekemään osasta verkkosivua dynaamisesti toimiva. Flash kasvattikin suosiotaan nopeasti ja erityisesti selaimen kautta pelattavien pelien alustana. Myöhemmin Macromedia Flash nimettiin uudelleen yrityskauppojen myötä Adobe Flashiksi.

JavaScriptin standardoinnin myötä oli mahdollista luoda interaktiivisia verkkosivuja, jotka toimivat kaikilla pääselaimilla. Näin myös otettiin ensimmäinen askel kohti perinteisten työpöytäsovellusten ja verkkosovellutusten välisten erojen hämärtämisessä.

2.2 Internet tänään

Viimeisen kymmenen vuoden aikana Internetin käyttäjät ovat kasvaneet räjähdysmäisesti myös kehittyvien maiden osalta.

WORLD INTERNET USAGE AND POPULATION STATISTICS

World Regions	Population (2009 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2009	Users % of Table
Africa	991,002,342	4,514,400	86,217,900	8.7 %	1,809.8 %	4.8 %
Asia	3,808,070,503	114,304,000	764,435,900	20.1 %	568.8 %	42.4 %
Europe	803,850,858	105,096,093	425,773,571	53.0 %	305.1 %	23.6 %
Middle East	202,687,005	3,284,800	58,309,546	28.8 %	1,675.1 %	3.2 %
North America	340,831,831	108,096,800	259,561,000	76.2 %	140.1 %	14.4 %
Latin America/Caribbean	586,662,468	18,068,919	186,922,050	31.9 %	934.5 %	10.4 %
Oceania / Australia	34,700,201	7,620,480	21,110,490	60.8 %	177.0 %	1.2 %
WORLD TOTAL	6,767,805,208	360,985,492	1,802,330,457	26.6 %	399.3 %	100.0 %

NOTES: (1) Internet Usage and World Population Statistics are for December 31, 2009. (2) CLICK on each world region name for detailed regional usage information. (3) Demographic (Population) numbers are based on data from the US Census Bureau . (4) Internet usage information comes from data published by Nielsen Online, by the International Telecommunications Union, by GfK, local Regulators and other reliable sources. (5) For definitions, disclaimer, and navigation help, please refer to the Site Surfing Guide. (6) Information in this site may be cited, giving the due credit to www.internetworldstats.com. Copyright © 2000 - 2010, Miniwatts Marketing Group. All rights reserved worldwide.

Kuva 1. Internetin käyttäjien lukumäärät vuosina 2000-2009 [5]

Kuvassa 1 näkyy, kuinka Internet-käyttäjien määrä on noussut viimeisen 10 vuoden aikana 400 % ja tavoittaa nykyään jo noin 1/3 ihmisistä. Kasvua on

vielä tulossa, mutta länsimaissa kasvu on ollut jo huomattavasti maltillisempaa. Suomessa kasvu on ollut 2-3 % luokkaa ja Internetin käyttäjiä väestöstä on yli 80 % [6].

Käyttäjät ovat nyt myös kokeneempia kuin ennen ja vaativat palveluiltaan enemmän. Tämä asettaa uusia vaatimuksia web-kehittäjille. Näiden vaatimusten täyttämiseen on kehitetty useita erilaisia tekniikoita, kuten esimerkksi Flash ja Silverlight, jotka mahdollistavat saumattoman käyttökokemuksen graafisessa ympäristössä: jokaisen napin painalluksen välillä ei tarvitse erikseen ladata verkkosivua uudestaan. Tämä toki oli mahdollista myös pelkän JavaScriptin avulla, mutta vuonna 2000 julkaistun *Document Object Model* (DOM) –määrittelyn kautta voidaan ladata palvelimelta tietoa verkkoselaimeen ilman, että välissä joudutaan lataamaan koko verkkosivu uudestaan [7, s, 13].

Tätä viime vuosien perustavanlaatuista muutosta on kuvattu termillä Web 2.0 ja sen tämän hetken terävintä kärkeä edustavat rikkaat Internet-sovellukset, *Rich Internet Applications* (RIA).

3 RIKKAAT INTERNET-SOVELLUKSET

Edellisessä luvussa todettiin, että DOM-määrittely mahdollisti rikkaiden Internet-sovellutusten esiinmarssin. Alla on tarkempi määritelmä:

> Rikkaat Internet-sovellukset ovat Ajax-verkkosovelluksia. Verkkoselain muutetaan perinteisestä kevyestä asiakaspäätteestä *(thin client)* raskaaksi asiakaspäätteeksi (*fat client)*, jolloin ne toimivat kuten perinteinen työpöytäsovellus. [7, s, 30.]

Rikkaiden Internet-sovellusten avulla saadaan aikaiseksi työpöytäsovelluksista tuttu saumaton käyttökokemus, jossa jokaisen napin painalluksen jälkeen ei näkymää ladata uudestaan. Tämä on huomattava muutos verrattuna perinteisiin web-sovelluksiin ja tuo työpöytäohjelmista tutut toiminnallisuudet myös verkkosovelluksiin.

Taulukossa 1 on vertailtu perinteisen työpöytäsovelluksen, verkkosovelluksen ja rikkaan Internet-sovelluksen eroja ja samankaltaisuuksia. Taulukosta voidaan nähdä, että rikkaat Internet-sovellukset yhdistelevät työpöytäsovellusten ja verkkosovellusten parhaita ominaisuuksia.

Taulukko 1 - Työpöytäsovellus, verkkosovellus ja RIA vertailussa [8]

Ominaisuus	Työpöytäsovellus	Verkkosovellus	RIA
Rikas käyttäjäkokemus	Kyllä	Ei	Kyllä
Interaktiivinen, reagoiva	Kyllä	Ei	Kyllä
Vähäinen ylläpito	Ei	Kyllä	Kyllä
Helposti levitettävissä	Ei	Kyllä	Kyllä

RIA-teknologiat ovat laajasti käytössä varsinkin peliteollisuudessa, mutta myös Googlen tarjoamat palvelut, kuten esimerkiksi Gmail-sähköposti ja Documents-toimistotyökalut ovat esimerkkejä RIA-teknologian mahdollisuuksista. RIA:n eduiksi voidaan lukea se, että pääsääntöisesti ne toimivat yhdellä julkaisulla kaikilla käyttöjärjestelmillä. Tämä kuitenkin riippuu valituista teknologioista. Näin saavutetaan huomattavia ajallisia ja näin myös rahallisia säästöjä, koska yhdellä kehitystyöllä saavutetaan kaikki eri käyttöjärjestelmät.

Macromedian ja Intelin vuonna 2003 sponsoroimassa tutkimuksessa päädyttiin RIA-teknologioista seuraaviin johtopäätöksiin. [9.]

- RIA-teknologiat tarjoavat laaja-alaisia ratkaisuja yritysten Internet-, intranet- ja yritysohjelmistojen pohjaksi ilman, että nykyisiä sijoituksia verkkosovelluksiin joudutaan korvaamaan.
- RIA mahdollistaa yrityksille tavan luoda uudenlaisia innovatiivisia käyttäjäkokemuksia ja sovelluksia, joiden tekeminen perinteisillä verkkotekniikoilla olisi erittäin vaikeaa tai jopa mahdottonta.
- RIA-teknologiat tarjoavat välillisiä taloudellisia hyötyjä, kuten korkealuokkaisia asiakasehdokkaita, myynnin kasvua, brändin kasvatusta,

pidempiä käyntejä verkkosivuilla, uusiutuvia käyntejä verkkosivuille, pienentyneitä verkkokaistan vaatimuksia, vähentyneitä tukisoittoja ja syvempiä asiakassuhteita.

- RIA-teknologioiden mahdollisuudet tarjoavat perustavanlaatuisen muutoksen Internetin sovellusten käyttämisessä ja johtavat sovelluksiin, jotka lunastavat Internetin lupaukset.

Seuraavassa luvussa käydään tarkemmin läpi Ajax sekä tehdään katsaus yleisimpiin RIA-teknologioihin.

3.1 Ajax

James Garrett julkaisi vuonna 2005 artikkelin *"Ajax: A New Approach to Web Applications".* Artikkelissa Garret kuvaili olemassa olevia tekniikoita ja nimesi näiden yhdistäväksi ylätekniikaksi Ajaxin *Asynchronous JavaScript And XML*. Vaikka termi Ajax oli tässä vaiheessa uusi, olivat siinä käytettävät osaset olleet jo olemassa tovin. [10.]

Ajax ei siis itsessään ole mitään, muuta kuin termi, jolla kuvataan joukkoa ohjelmointitekniikoita. Nämä ohjelmointitekniikat mahdollistavat web-ohjelmoinnin, jossa verkkosivu tai osa verkkosivua voi tarvittaessa ladata palvelimelta lisää tietoa ilman, että koko verkkosivua joudutaan lataamaan uudestaan. Tällä saadaan aikaiseksi työpöytäsovellutuksia mukaileva saumaton käyttökokemus. [10.]

Ajax-sovellus koostuu siis joukosta erilaisia ohjelmointitekniikoita. Alla on listattu yksi mahdollinen Ajax kokonaisuus.

- *Extensible HyperText Markup Language* (XHTML)
- *Document Object Model* (DOM)
- JavaScript
- *Cascading Style Sheets* (CSS)
- *Extensive Markup Language* (XML).

Edellä olevassa kokonaisuudessa eri ohjelmointitekniikoiden roolit ovat seuraavat. XHTML pitää huolen varsinaisen datan näyttämisestä verk-

koselaimelle. DOM sisältää itse datan. CSS kertoo DOM:n välityksellä selaimelle, kuinka sisältö asemoidaan ja kuvataan. Javascript on vastuussa verkkoselaimen ja palvelimen välisestä liikenteestä sekä DOM:n muokkaamisesta. XML on protokolla, jonka avulla tieto kulkee verkkoselaimen ja palvelimen välillä. [7, s, 5.]

Yllä olevasta kokonaisuudesta varsinaisia pakollisia osia ovat JavaScript ja DOM. Muut osat voidaan vaihtaa toisiin tai jättää jopa pois. Esimerkiksi XML:n rinnalle on tullut muita mahdollisuuksia, kuten JavaScript Object Notation (JSON) Lisäksi Ajaxiin liitetään usein myös monia muita ohjelmointitekniikoita, mutta niiden käyttö on valinnaista.

3.1.1 DOM

Document Object Model (DOM) -ohjelmointirajapinta esittää dokumentin puurakenteen muodossa. Kaikkien XML-muodossa olevan datan rakenne voidaan esittää DOM:lla. XHTML-merkintäkieli on myös XML:ää, joten tämän johdosta www-sivujen rakenne voidaan esittää puumuodossa DOM:in avustuksella. Näin ollen voidaan sivujen rakennetta muuttaa DOM:n välityksellä suoraan lennosta ja tietoa voidaan myös välittää asynkronisesti verkkoselaimen ja palvelimen välillä.

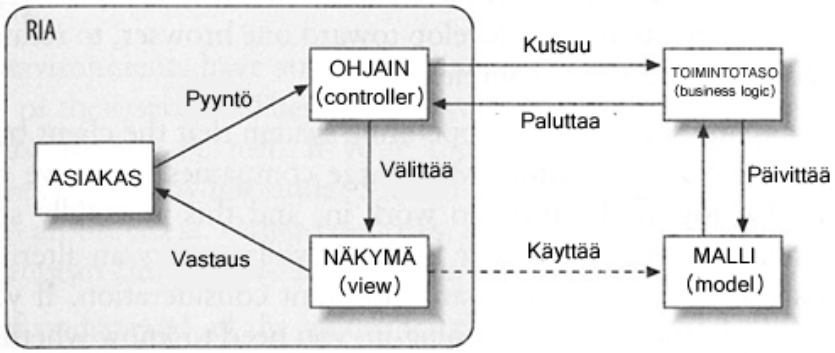
3.1.2 Ajaxilla toteutettujen Ria-sovellusten edut

Ajaxin ja puhtaasti Ajaxilla toteutettujen RIA-sovellutusten etuihin kuuluvat muun muassa seuraavat seikat [7, s, 31]:

- Ajax-sovellukset eivät vaadi erillistä asennusta, päivitystä eikä jakeluverkkoa, koska kaikki palvelut tulevat www-palvelimelta.
- Ajax-sovellutukset ovat normaaleja työpöytäsovelluksia turvallisempia.
- Ajax-sovellutukset sijaitsevat verkossa, joten niiden käyttö ei ole sidottu yhteen tietokoneeseen. Näin ollen niitä voidaan käyttää mistä vaan.
- Ajax-sovellukset toimivat kaikilla käyttöjärjestelmillä.

Laajemmat Ajax-sovellukset rakennetaan normaalisti käyttäen MVC-arkkitehtuuria (*Model-View-Controller)*. Näin ollen sovellusten myöhempi

muuttaminen on helpompaa ja esimerkiksi käyttöliittymiä voidaan luoda erilaisia eri käyttötarkoituksiin.



Kuva 2. Ajax RIA -sovellus MVC-arkkitehtuurilla [7, s, 31.]

Kuvassa 2 havainnollistetaan Ajax RIA -sovellusta MVC-arkkitehtuurilla. Erot web-sovelluksissa käytettyyn MVC-arkkitehtuurin ovat siinä, että RIA-sovelluksissa asiakas, ohjain ja näkymä muodostavat raskaan asiakkaan (fat client), joka on merkittynä kuvassa harmaalla laatikolla. Käytännössä tämä tarkoittaa sitä, että kun näkymää muutetaan, ei verkkosivua tarvitse ladata kokonaan uudestaan kuten perinteisessä web-sovelluksessa.

Ajax toimii pohjana kaikille RIA-teknologioille. Usein Ajax on kuitenkin toteutettu niissä niin, että kehittäjän ei varsinaisesti tarvitse sitä itse toteuttaa, vaan se on piilotettu rajapinnan taakse.

3.2 Yleisimmät RIA-teknologiat

Vaikka itsessään Ajax ei siis ole mitään, on sen ansiosta pystytty luomaan helpommin hallittavissa olevia ohjelmointitekniikoiden joukkoja. Tästä huolimatta on Ajax-sovellutusten tekeminen huomattavasti haastavampaa kuin perinteisten verkkosovellutusten. Tämän johdosta on isompien sovellutusten ja varsinkin rikkaiden Internet-sovellusten luomiseen tehty useita erilaisia helpottavia teknologioita. Näistä kolme suosituinta on Adobe Flex, Java ja Microsoft Silverlight [11].

Niitä yhdistävä tekijä on se, että niiden RIA-toiminnallisuus saavutetaan selainlaajennusten (*plugin)* kautta. Adobe Flexin toiminnallisuuden takaava Adobe Flash -laajennus löytyy 97 %:sta selaimia, Java-laajennus 80 %:sta ja Microsoft Silverlight 54 %:sta. [11.]

3.2.1 *Adobe Flex*

Adobe Flex on Adobe Systemsin alun perin vuonna 2004 julkaisema avoimen lähdekoodin ohjelmisto web-kehittäjille. Tällä hetkellä uusin versio Flexistä on maaliskuussa 2010 julkaistu Flex 4.0. Tunnetuin Flexiä käyttävä verkkosivu on Googlen videopalvelu Youtube. Youtube on tällä hetkellä Internetin kolmanneksi suurin sivusto[12].

Kuva 3. Kuvakaappaus Youtuben RIA-mediasoittimesta

Kuvassa 3 nähdään Youtuben mediasoitin, joka ei toiminnallisuudeltaan ja ulkonäöltään poikkea vastaavista työpöytäohjelmista, vaikka kyseessä on puhtaasti Internetissä toimiva sovellus.

Flexissä asiakkaan sovelluslogiikat luodaan käyttäen objektipohjaista ActionScript-ohjelmointikieltä, jonka perustat ovat standardoidussa EcmaScriptissä [13], eli siis samassa standardissa, johon myös JavaScript perustuu. Käyttöliittymän asettelu ja toimintojen määritys tehdään käyttäen MXML-kieltä, joka perustuu XML:ään. [13.]

Flexissä on valmiina laaja komponenttikirjasto, jota voidaan käyttää monipuolisten Internet-sovellusten ja RIA-sovellusten luomiseen. Nämä sovellu-

tukset toimivat selaimissa Adobe Flash Player –ohjelmiston avulla ja selainten ulkopuolella Adobe Air –sovelluksessa.[13]

Adobe Flexin vahvuuksiin kuuluu se, että suurimpaan osaan selaimista on asennettu Flash Player, joten sillä valmistetut sovellukset todennäköisesti toimivat asiakkailla. Flash on kuitenkin viime aikoina joutunut vahvaan vastatuuleen ja esimerkiksi Applen Steve Jobs on voimakkaasti kritisoinnut Flashin heikkouksia kuten esimerkiksi sen tehovaatimuksia ja herkyyttää kaatumisille [14].

3.2.2 Java

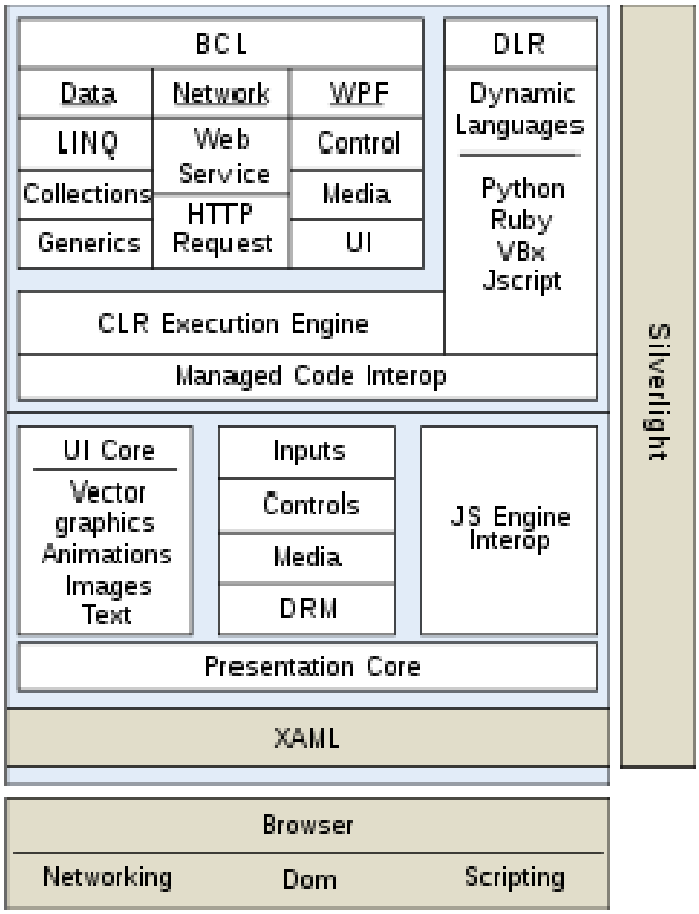
Java-alustalla voidaan toteuttaa RIA-sovelluksia monilla eri tavoin ja sille on tarjolla erilaisia ohjelmistokehyksiä, jotka tukevat RIA-kehitystä. Esimerkiki Javan JFC-luokkakirjaston Swingillä voidaan toteuttaa RIA-sovelluksia. Monipuolisempiakin tuotteita on tarjolla. Yksi näistä on Javan kehittäjän Sun Microsystemsin RIA-kehitystä tukeva tuote JavaFX [15].

JavaFX on varsin uusi tuote RIA-kehityksessä. Ensimmäinen versio siitä julkaistiin vuoden 2009 helmikuussa. Tämän johdosta sen suosio on vielä hyvin rajattu verrattuna esimerkiksi Flashiin pohjautuvaan Adobe Flexiin, joka julkaistiin jo vuonna 2004.

JavaFX-sovelluksia kehitetään staattisesti tyypitetyllä (*statically typed)* kuvaavalla ohjelmointikiellä (*declarative language)* JavaFX Scriptillä ja myös Java-koodia voidaan integroida sovelluksiin. JavaFX:n vahvuuksiin kuuluu se, että se toimii kaikilla Java Runtime Enviromentiä (JRE) ja Java ME:tä tukevilla alustoilla. Näin ollen sitä voidaan käyttää tietokoneissa, kännyköissä ja esimerkiksi jopa Blu-ray-soittimissa. [15.]

3.2.3 Microsoft Silverlight

Microsoft Silverlight on alun perin vuonna 2008 julkaistu web-ohjelmointiympäristö, jossa on hyvin paljon samanlaista toiminnallisuutta kuin Adoben Flashissä ja näin ollen myös Adobe Flexissä. Microsoft Silverlightin toiminnallisuus selaimissa tapahtuu selainlaajennuksella, joka on saatavissa kaikille suurimmille selaimille sekä käyttöjärjestelmille.



Kuva 4. Silverlight 2 -arkkitehtuuri [16]

Kuvassa 4 olevassa Silverlightin arkkitehtuurirakenteesta voidaan selvästi nähdä, kuinka Ajaxin perusrakenteet DOM ja JavaScript ovat sen osia. Nämä on kuitenkin irrotettu Silverlightsta, jolloin sitä voidaan teoriassa käyttää kaikkien eri selaimien kanssa, joihin on tehty Silverlight-selainlaajennus. XML:n virkaa Silverlight-sovelluksissa hoitaa XAML, ja sen avulla myös rakennetaan sovellusten graafinen ulkoasu. Valkoisella pohjalla on arkkitehtuurikuvauksessa palvelinpuoli, jonka ohjelmointi suoritetaan käyttäen .NET-ohjelmointikieltä, mutta myös muita ohjelmointikieliä, kuten Pythonia tai Rubyä. [16.]

4 RSENCE RIA -OHJELMISTOKEHYS

RSence on RIA-ohjelmistokehys, joka on suunniteltu reagoivien graafisen ulkoasun omaavien sovellutusten tekemiseen Internetiin. RSence saavutti version 2.0.0 heinäkuussa 2010, ja sen kehitystyö on jatkuvaa. RSencen kehittäjinä toimivat Juha-Jarmo Heinonen ja Toni Nurminen.

RSence on julkaistu GNU GPL -lisenssillä, mikä tarkoittaa sitä, että RSencen avulla tehtyjen ohjelmien lähdekoodit tulee jakaa eteenpäin. RSenceen on kuitenkin mahdollista saada myös muita lisenssejä, jotka mahdollistavat ohjelmistokehyksen käytön paremmin kaupallisiin tarkoituksiin.

RSence koostuu palvelinpuolen ohjelmistokehyksestä, joka on toteutettu Ruby-ohjelmointikielellä sekä asikaspuolen ohjelmistokehyksessä, joka puolestaan on toteutettu käyttäen JavaScriptiä. Itse palvelin on ohjelmoitu sekä Rubyllä, että C-ohjelmointikielellä. Näin on saavutettu käytettävyyden ja tehokkuuden kannalta toimiva yhdistelmä. [17.]

RSence-palvelin

Rsencen palvelinpuolen toteutus sisältää:

- sessioiden hallinnan
- asiakkaiden resurssien hallinnan
- sovelluslaajennusten hallinnan
    - automaattion kutsujen polutus, *routing*
    - SOAP-tuki (*Simple Object Access Protocol*)
- tiedon synkronoinnin
- lipukepalvelut (*ticket services).*

Haastavat tekniset ongelmat, kuten istuntojen hallinta ja tietojen synkronointi, hoituvat RSence-palvelimen toimesta. Näin säästetään aikaa, jota voidaan käyttää itse sovellusten kehittämiseen.

RSence-asiakaspuoli

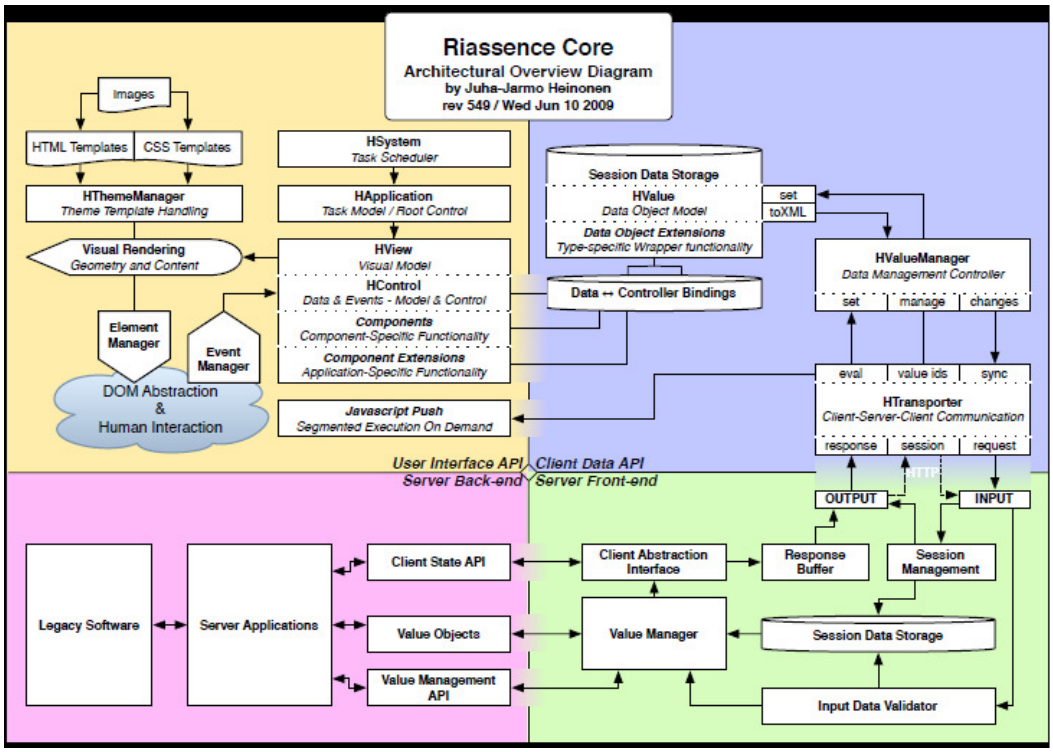
Asikaspuolen ohjelmistokehys sisältää seuraavat piirteet:

- modulaarinen, nopeasti vastaava, vakaa
- automaattinen datan synkronointi
- tehokas pakkausjärjestelmä *packaging system*

- tuki suurimmalle osalla verkkoselaimista
- YAML-merkintäkielellä toteuttettavissa oleva graafinen käyttöliittymä.

Käytännössä tämä tarkoittaa sitä, että RSence-asiakaspuoli huolehtii suurimmasta osata Ajax RIA -kehityksen työläistä ja teknisesti haastavista osista. Kehittäjän tehtäväksi jää graafisen käyttöliittymän luominen YAML-merkintäkielellä, eikä kehittäjän tarvitse murehtia, toimiiko sovelluslaajennus eri verkkoselaimilla.

Kuvassa 5 on kuvattu tarkemmin RSencen teknistä arkkitehtuuria.



Kuva 5. Näkymä RSencen arkkitehtuurista

Kuvassa 5 on yksityiskohtainen kuvaus RSencen arkkitehtuurista, joka koostuu neljästä isommasta kokonaisuudesta. Yllämainittu asiakaspuolen ohjelmistokehys koostuu käyttöliittymärajapinnasta (*User Interface API)* ja asiakasdatarajapinnasta (*Client Data API)*. Palvelinpuolen ohjelmistokehys on jaettu myös kahteen kokonaisuuteen palvelimen taustajärjestelmään (*Server Back-end)* sekä palvelimen päälijärjestelmään (*Server Front-end)*. Palvelinpuolen toiminnat on tehokkaasti sijoitettu rajapintojen taakse, joten kehittäessä RSence-sovelluslaajennuskia ei niiden toimintaan tarvitse puuttua.

RSencen dokumentaatio ja erilaiset sovellusesimerkit löytyvät osoitteesta http://rsence.org.

4.1 Ruby-ohjelmointikieli

Koska RSencen palvelinpuolen ohjelmistokehys on toteutettu käyttäen Ruby-ohjelmointikieltä, esitellään tässä alaluvussa hieman Rubyn perusperiaatteita ja toiminnallisuutta.

Ruby on tulkattava, dynaaminen ja dynaamisesti tyypittävä oliopohjainen ohjelmointikieli [18]. Vaikka Ruby onkin tulkattava ohjelmointikieli (skriptikieli), on sillä mahdollista toteuttaa ohjelmia käyttöjärjestelmien alimmilta tasoilta aina graafisiin käyttöliittymiin. Ruby toimii kaikissa yleisimmissä käyttöjärjestelmissä ja JRuby – Rubyn Java-implementaatio myös kaikissa alustoissa, joihin on saatavilla Java-virtuaalikone.

Klassinen ohjelmointiesimerkki ”Hei maailma” on Rubyllä toteutettuna seuraavanlainen:

```
puts ”Hei maailma!”
```

4.1.1 Rubyn historia

Rubyn historia alkaa vuodesta 1993, kun japanilainen Yukihiro ”Matz” Matsumoto ei löytänyt olemassa olevista ohjelmointikielistä hänen mieltymyksiinsä sopivaa. Matsumoto halusi ohjelmointikielen, jossa yhdistyisi Perlin tehokkuus ja joka olisi enemmän olio-ohjelmointiin suuntautuva kuin Python. Koska kyseistä yhdistelmää ei vielä ollut olemassa, päätti Matsumoto luoda sen itse. Näin Ruby sai alkunsa. Rubyn esikuvina on toiminut Smalltalk, Python, LISP, Eiffel, Ada ja C++, joista Matsumoto on ottanut mukaan Rubyyn mielestään parhaat palat. [19, s, 99-101.]

Rubyn versio 1.0 julkaistiin joulukuun 25. päivä vuonna 1996 ja varsinainen maailmanvalloitus alkoi vuonna 1997, kun Rubyn virallinen verkkosivu avattiin myös englanninkielisenä. Tätä ennen Rubyn leviämistä oli hidastanut se, että Matsumato oli tehnyt Rubyn dokumentaation vain japaniksi. Todellinen läpimurto tapahtui, kun vuonna 2003 Rubyllä ohjelmoitu Ruby on Rails ilmestyi.

4.1.2 Ruby on Rails

Ruby on Rails (RoR) on avoimen lähdekoodin ohjelmointikehys web-sovellusten luomiseen Ruby-ohjelmointikielellä. Sen pääkehittäjänä on toiminut David Heinmer Hanson. Vuoden 2005 aikana Ruby on Railsin ja samalla myös Rubyn käyttäjämäärät kasvoivat räjähdysmäisesti [19, s, 103-106].

Ruby on Rails mainitaan tässä insinöörityössä lähinnä sen takia, että sen käyttäjät muodostavat suuren osan Rubyn käyttäjistä. Ilman Ruby on Railsin suosiota Ruby olisi jäänyt huomattavasti pienempään osaan ohjelmointikielien kattavassa kirjossa. Ruby on Rails tukee myös sisäänrakennetusti Ajaxia, mutta varsinaisia RIA-sovellutuksia sillä ei ole tarkoitus tehdä. Ruby on Railsiä käyttävistä palveluista tunnetuin on Twitter.

4.1.3 RubyGems

RubyGems on Rubyn standardoitu pakkaus- ja asennusohjelmistokehys, jonka avulla voidaan laajentaa Rubyn toiminnallisuutta. RubyGems ei kuulu Rubyn perusasennukseen, joten se tulee itse asentaa osoitteesta http://rubygems.rubyforge.org löytyvien ohjeiden avulla. RubyGemsistä löytyy lukuisia erilaisia laajennuksia, kuten tässä insinöörityössä tarkemmin esitelttävä RSence.

4.2 RSencen esittely

RSence on hyvin lähellä Ajaxin perusmallia, mutta siihen liittyvät tekniset seikat on piilotettu rajapintojen taakse eikä kehittäjän näin ollen tule huolehtia erikseen sessionhallinnasta, mikä nopeuttaa sovellutusten tekemistä huomattavasti. Kehittäjän ei tarvitse edes välttämättä osata JavaScriptiä, sillä ulkoasu ja siihen liittyvä toiminnallisuus voidaan kuvata kokonaan YAML-merkintäkielellä. Varsinaista JavaScript osaamista tarvitaan vain, jos halutaan muuttaa RSencen komponenttikirjaston osasia tai luoda uusia komponenttejä. [17.]

RSencen erikoispiirteenä on se, että html-koodi generoidaan verkkoselaimessa eikä palvelimella, mikä vähentää palvelintaakkaa ja nopeuttaa sivujen lataamista.

Komponenttikirjasto sisältää suurimman osan työpöytäsovelluksista tutuista komponenteista kuten erilaisista painikkeista ja tekstinsyöttökomponenteista.

RSence RIA-sovellusesimerkkin yhteydessä luvussa 5 käydään tarkemmin läpi RSence-sovelluslaajennuksen rakennetta sekä siihen liittyvien tiedostojen merkitystä sovellusten toiminnallisuuden kannalta.

4.2.1 RSencen asennus

RSencen toimii kaikissa yleisimmissä käyttöjärjestelmissä. RSencen asennus vaatii, että ympäristössä, johon asennus tehdään, on asennettuna Ruby ja RubyGems. Tämän jälkeen asennus voidaan suorittaa yksinkertaisesti RubyGemsin kautta:

```
gem install rsence
```

Tämän jälkeen voidaan alustaa uusi RSence-projekti käskyllä:

```
rsence initenv kohde_hakemisto
```

Alustamisen yhteydessä kysytään projektin nimeä ja sitä, että halutaanko projekti näkyviin myös kehitysympäristön ulkopuolelle eli Internetiin. Varsinaiset RSence-sovellukset lisätään kohde_hakemisto/plugins –kansioon.

Asennuksen jälkeen RSence käynnistetään asennuskansiossa suoritetulla käskyllä:

```
rsence run
```

Tämän jälkeen RSence näkyy verkkoselaimessa osoitteessa http://0.0.0.0:8001.

4.2.2 RSencen dokumentaatio

RSence.org-sivustolla on dokumentaatio jaettu kahteen suurempaan kokonaisuuteen: asiakaspuolen sekä palvelimen ohjelmointirajapintojen dokumentaatioihin. Näiden lisäksi on vielä osittain keskeneräinen RSencen yleisesittely, jossa käydään läpi sovellutusten perustat.

4.2.3 RSencellä tapahtuvan RIA-kehityksen tekninen vaativuus

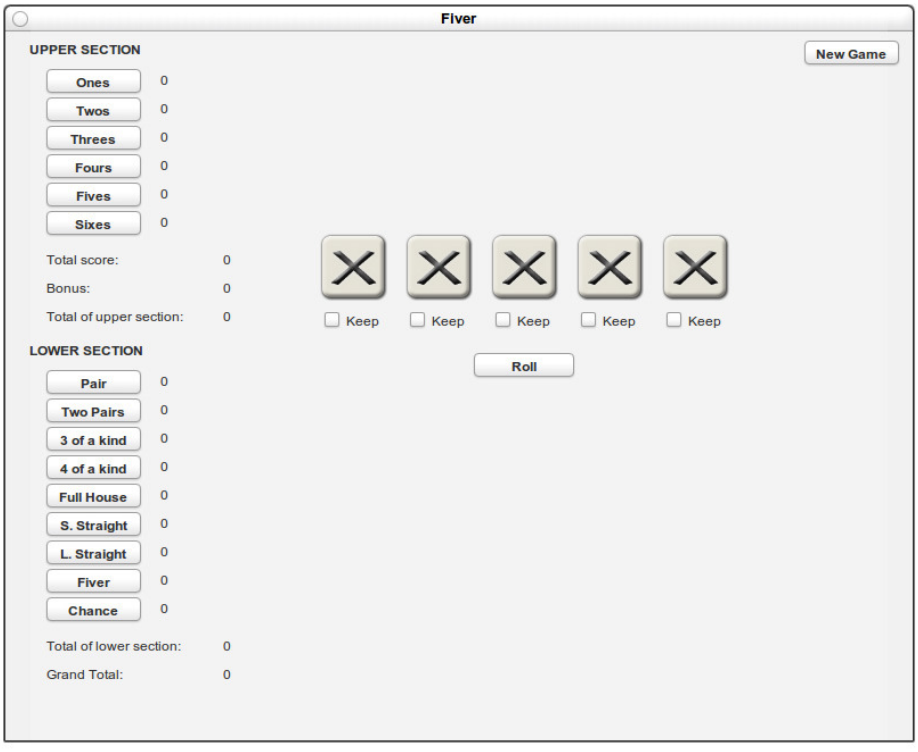
RSencellä tehtyjen RIA-sovellutusten tekeminen ei ole alkuun pääsyyn jälkeen kovinkaan hankalaa. Omien komponenttien tekemisessä tarvitaan kat-

tavampaa osaamista JavaScriptistä, mutta palvelinpuolen logiikoiden tekeminen vaatii tutustumista Ruby-ohjelmointikieleen. Rubyn hallintaa tarvitaan ymmärrystä olio-ohjelmoinnista, mutta se soveltuu myös ensimmäiseksi opeteltavaksi ohjelmointikieleksi.

Näistä lähdökohdista voidaan todeta, että RSencellä RIA-kehityksen pitäisi luonnistua helposti, jos kehittäjältä löytyy kokemusta nykyaikaisten verkkosovellusten tekemisestä, mutta kehitys onnistuu myös heikommista lähtökohdista.

5 RSENCE RIA -SOVELLUSESIMERKKI

RSence RIA -sovellusesimerkiksi tein suositun Yahtzee-noppapelin klooniin nimeltään Fiver. Pelissä heitetään viittä noppaa, joista yritetään saada erilaisia yhdistelmiä. Heittokertoja on yhteensä kolme, mutta niitä kaikkia ei tarvitse käyttää. Heittojen välissä voidaan laittaa syrjään ja heittää uudestaan vain osa nopista.



Kuva 6. RSence-sovelluslaajennus Fiver

Kuvassa 6 nähdään sovelluslaajennus Fiverin lopullinen ulkoasu. Sen eri komponentit käydään läpi tarkemmin myöhemmin tässä luvussa. Vasemmassa reunassa ovat pisteet sekä painikkeet, joilla noppien tulokset voidaan

asettaa eri pistekategorioihin. Keskellä näkymää sijaitsevat nopat, noppien lukitukset ja noppien heittämisestä vastaava painike. Oikeassa yläkulmassa näkyy New Game –painike, jonka avulla peli voidaan aloittaa alusta.

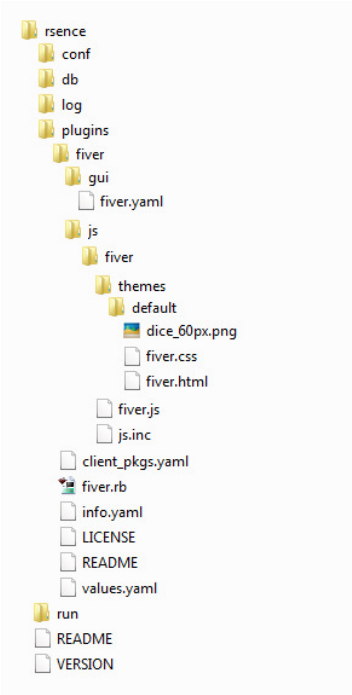
Ennen sovelluksen tekemistä minulla ei ollut varsinaisesti kokemusta JavaScriptistä, Yaml:sta, mutta pystyin tästä huolimatta suhteellisen helposti suoriutumaan sovelluksen tekemisestä. Tästä annan kiitoksen sille, että RSence on suunniteltu helposti käytettäväksi ja erityisesti RSencen kehittäjille, jotka neuvoivat hankalimmissa teknisissä yksityiskohdissa. Sovelluslaajennuksen esittelyn yhteydessä käydään läpi myös RSenceen liittyviä teknisiä asioita.

RSence-sovelluslaajennus Fiverin tarkempi läpikäynti mahdollistaa tämän luvun käyttämisen oppaana RSence-kehityksen aloittamiseen.

Ohjelman lähdekoodi on kokonaisuudessaan luettavissa liitteessä 1.

5.1 RSence-sovelluslaajennus Fiver

Kuten kaikki RSence-sovelluslaajennukset Fiver asennetaan siirtämällä sen sisältämät tiedostot plugins-alihakemistoon. Alla olevassa kuvassa havainnollistetaan siihen kuuluvia tiedostoja.



Kuva 7. RSence-sovelluslaajennuksen hakemistorakenne

Käydään seuraavaksi tarkemmin läpi Fiver-sovelluksen tiedostoja sekä niiden merkitystä sovelluslaajennuksen kokonaisuuden kannalta. Näistä tärkeimmät ovat asiakaspuolen ulkoasusta ja toiminnallisuuden kuvaava fiver/gui/fiver.yaml-tiedosto sekä palvelinpuolen logiikkaa hoitava fiver/fiver.rb.

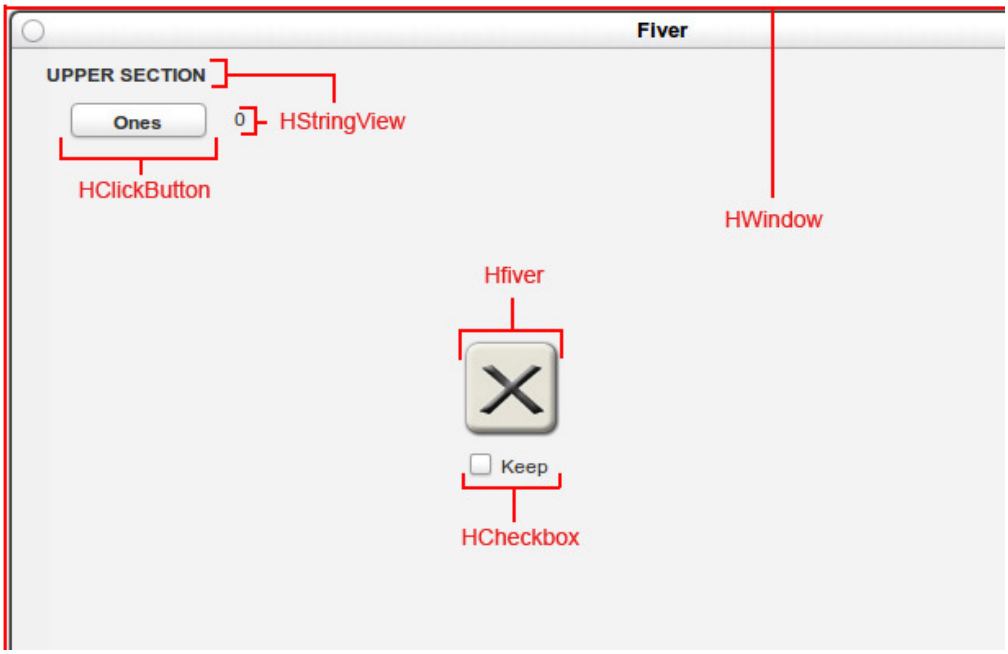
5.2 Asiakaspuolen ulkoasu ja toiminnallisuus – gui/fiver.yaml

RSencen dokumentaatio listaa seuraavat tarkoitukset ja ominaisuudet gui/main.yaml-tiedostolle [20].

- Se voidaan nimetä myös sovelluslaajennuksen mukaan, kuten esimerkiksi fiver.yaml.
- Se määrittelee käyttöliittymän rakenteen.
- Se sisältää asiakaspuolen arvojen sidonnaisuudet (*value bindings)*, jotka tulee olla määriteltynä esimerkiksi values.yaml-tiedostossa.
- Se voi määritellä myös staattisia tietoja tai paikallisia merkkijonoa.
- Järjestelmä rakentaa käyttöliittymän perustuen tähän tiedostoon automaattisesti, kun RSence verkkosivu ladataan tai kun se uudelleen ladataan käyttäjän toimesta.

Tehtäessä yaml-tiedostoja on huomioitava se, että yaml-merkintäkieli on tarkka sisennyksien suhteen ja väärin tehdyt sisennyt johtavat normaalisti siitä, että ohjelma ei toimi ollenkaan.

Lähdekoodi gui/fiver.yaml-tiedostosta on kokonaisuudessaan luettavissa liitteessä 1. Yaml-tiedoston alussa käydään läpi käyttöliittymään liittyvät JavaScript-luokat. Jos käytetään itse tehtyjä komponentteja, kuten Fiverissa olevaa noppakomponenttia, tulee sen sijainti ilmoittaa myös sovelluslaajennuksen päähakemistossa sijaitsevassa client_pkgs.yaml-tiedostossa, joka löytyy kokonaisuudessaan liitteestä 1. sivut 19-20.



Kuva 8. RSence-sovelluslaajennus Fiverin komponentit

Kuvassa 8 on merkittynä punaisella RSence-sovelluslaajennus Fiverin käyttämät komponentit. Komponentit käydään läpi lähdekooditasolla luvuissa 5.2.1 – 5.2.5. Fiverin käyttämät komponentit lukuun ottamatta Hfiver-komponenttia ovat yleiskäyttöisiä standardikomponentteja, joten lähdekooditason esimerkkien avulla voidaan toteuttaa myös muita Rsence-sovelluslaajennuksia.

5.2.1 HWindow

HWindow on Rsence-sovelluslaajennus Fivern alinäkymä, joka sitoo sisälleen muut sovelluslaajennuksessa käytettävät komponentit.

```
- class: HWindow
  rect: [0, 0, 800, 650]
  options:
    closeButton: true
    noResize: true
    label: Fiver
```

Yaml-kuvauksen ensimmäisellä rivillä todetaan, että kyseessä on HWindow-luokan vakiokomponentti. Tämän jälkeen kerrotaan sen sijainti selainikkunassa sekä komponentin mittasuhteet formaatissa [selain-x,selain-

y,komponentti-leveys,komponentti-korkeus]. RSencen palvelinpuolen API-dokumentaatio kuvaa erilaisia tapoja kuvata sijaintia, jolloin komponentit voidaan määritellä myös joustavilla mittasuhteilla.

Tämän jälkeen kuvataan komponentin asetukset, jotka tässä tapauksessa määrittävät seuraavat seikat:

- closeButton : komponentti on suljettavissa sovelluksen vasemmasta yläreunassa olevassa painikkeesta
- noResize: komponentin kokoa ei käyttäjän toimesta voida muuttaa
- label: komponentin, tässä tapauksessa ikkunan yläreunassa olevan nimen sisältö.

Lisää vaihtoehtoja asetuksiin löytyy asiakaspuolen API-dokumentaatiosta. [21.]

5.2.2 HClickButton

HClickButton-komponenttia käytetään erilaisten painikkeiden luomiseen. Fiverissa HClickButton on HWindow-komponentin alinäkymänä, joten esimerkiksi sen sijainti- ja mittasuhdemäärittelyn x = 0 ja y = 0 kordinaattipiste sijaitsee HWindow-komponentin vasemmassa yläreunassa, eikä siis koko selainikkunan, jossa sovelluslaajennus pyörii.

```
- class: HClickButton
rect: [35, 30, 85, 25]
bind: :values.submit_ones
options:
  label: "Ones"
```

HClickButton on sidottu bind:-määritteellä muuttujaan submit_ones, joka löytyy sovelluslaajennuksen juurikansiosta tiedostossa values.yaml, jonka lähdekoodi on kokonaisuudessaan luettavissa liittessä 1 sivuilla 17-19. Kun komponentti on sidottu muuttujaan, voidaan sen arvoa käsitellä palvelinpuolen logiikkaa määriteltässä. Normaalisti kaikki komponentit on sidottu muuttujiin, mutta poikkeuksiakin on kuten luvussa 5.2.1 esitetty HWindow kom-

ponentti. Seuraavana on listattu esitellyn HClickButtoniin sidottu :submit_ones:in lähdekoodi values.yaml-tiedostossa.

```
:submit_ones:
  :value: 0
  :responders:
    - :method: submit_ones
```

Edellä olevassa on aluksi esitelty muutujan nimi, sitten alkuarvo sekä vastaaja *responder*. Vastaaja laukaisee palvelinpuolen logiikasta metodin submit_ones, joka esitellään tarkemmin luvussa 5.3.3. Values.yaml-tiedosto mahdollistaa kaksisuuntaisen tiedon vaihdon asiakaspuolen ja palvelinpuolen välillä.

5.2.3 HStringView

HStringView on merkkijonojen esittämistä varten tehty komponentti. Alla olevassa lähdekoodinpätkässä gui/fiver.yaml-tiedostossa havainnollistetaan staatisen merkkijonon käyttämistä HStringViewin kautta.

```
- class: HStringView
  rect: [20, 5, 125, 20]
  options:
    value: "<b>UPPER SECTION</b>"
```

Merkkijonon sisältö laitetaan value:-asetukseen, jossa voidaan käyttää esimerkin mukaisesti HTML-merkintäkieltä.

5.2.4 HCheckbox

HCheckbox-komponentti tekee käyttöliittymään valintalaatikon, jonka arvo vaihtelee valitun, *True,* ja ei-valitun, *False,* välillä. HCheckBox toteutetaan muuten hyvin samalla tavalla kuin aiemmin selitetty HClickButton.

5.2.5 Hfiver –lisäkomponentti

Hfiver on Fiver-sovelluslaajennuksen ainoa vakiokomponenttikirjaston ulkopuolelta tuleva lisäkomponentti. Se perustuu RSencen verkkosivuilla olevaan komponenttiesimerkkiin, jota laajensin sopivaksi omaan sovelluslaajennukseeni.

Hfiver näyttää gui/fiver.yaml-tiedostossa samankaltaiselta kuin aiemmin esitetyt komponentit.

```
- class: Hfiver
  rect: [275, 175, 60, 60]
  bind: :values.die_one
```

Koska kyseessä on lisäkomponentti, on sen toiminnallisuus ja ulkoasu *theme* erikseen määritely sovelluslaajennuksen js-kansiossa tai kansiossa, joka on määritelty client_pkgs.yaml-tiedostossa.

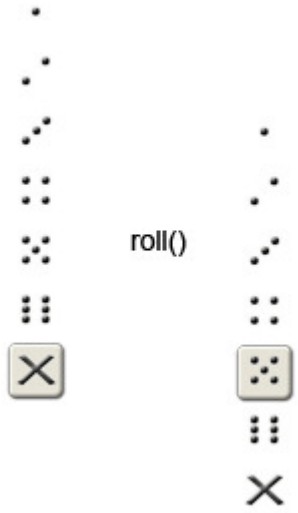
Noppa-komponentti toimii niin, että painettaessa Roll-painiketta, laukaisee se vastaajan, joka puolestaan laukaisee palvelinpuolen metodin roll(), joka arpoo satunnaisluvun 1-6 välillä ja palauttaa sen takaisin values.yaml-tiedoston kautta gui/fiver.yaml-tiedostoon, joka päivittää lukeman nopan silmäluvuksi.

Luvun arpominen suoritetaan palvelimella, koska näin voidaan varmistaa, että huijaamista nopan silmäluvuissa ei tapahdu. Kun luku on arvottu, Hfiver päivittää näkymää alla olevan funktion avulla, joka puolestaan muuttaa DOM:a fiver.css:n (liite 1 sivu 20) ja fiver.html:n (liite 1 sivu 20) osalta:

```
Hfiver = HControl.extend({
  tileSize: 60,
  componentName: 'fiver',
  refreshValue: function(){
    var
    _val = this.value,
    _tileHeight = this.tileSize;
    if(_val>=1 && _val<=6){
      var _tilePosY = 0-(_val*_tileHeight);
    this.setStyleOfPart('state','background-position','0px
'+_tilePosY+'px');
    }
    else{
```

```
        var _tilePosY = 0-(7*_tileHeight);

        this.setStyleOfPart('state','background-position','0px
'+_tilePosY+'px');

      }

    }

});
```

Funktion _val arvo on siis välitetty palvelinpuolen metodilta ja HFiver näyttää tuon arvon mukaisen silmäluvun, muuttamalla html-elementin taustakuvaa.



Kuva 9. Aloitustilanteesta arvotaan luku 5

Kuvassa 9 havainnollistetaan, kuinka yhdellä ja samalla kuvalla voidaan näyttää kaikki mahdolliset silmäluvut. Tätä tekniikkaa kutsutaan myös CSS-Sprite:ksi. Alkutilanteessa kuvasta näytetään silmälukua X, kun noppaa heitetään(*roll()*), arvotaan uusi silmäluku ja kuva liikutetaan näyttämään uutta silmäluka.

5.3 Palvelinpuolen logiikka – fiver.rb

Palvelinpuolen logiikkaa sijaitsee sovelluslaajennuksen juurihakemistossa. Se voidaan nimetä joko sovelluslaajennuksen nimen mukaan tai main.rb:ksi. Palvelinpuolen logiikkaa koostuu sovelluslaajennuksen omista metodeista sekä RSencen GUIPlugin-luokan metodeista. Jos sovelluslaajennus ei ole graafinen, se voidaan periyttää Plugin-luokasta.

Kokonaisuudessaan fiver.rb:n lähdekoodi on luettavissa liitessä 1 sivuilla 6-16. Seuraavaksi käydään alla tarkemmin läpi joitakin tärkeitä metodeja sekä kuriirin virkaa hoitava Message-luokka. Suurin osa metodeista lähdekoodissa on Fiver-pelin logiikkaan ja pisteiden laskuun liittyviä, eikä niitä käydä sen tarkemmin läpi lukuun ottamatta submit_ones-metodia luvussa 5.3.3.

5.3.1 RSence::Message-luokka

Message-instanssia käytettään ”kuriiriluokkana”, kun käsitellään asikaspuolen ja palvelinpuolen pyyntöjä. Se alustetaan järjestelmän toimesta, ja lyhytnimi instanssille on msg. [22.]

Message-luokan kautta käsitellään kaikkia asiakkaiden istuntoja (*sessions)*. Koska RSence luo jokaisesta sovelluslaajennuksesta vain yhden instanssin, joka on siis kaikkien palvelimeen yhdistettyjen asiakkaiden käytössä, nousee istuntojen hallinta merkittävään rooliin. Jokaiselle asiakkaalle luodaan RSence-järjestelmässä oma istunto, jossa säilytetään kaikki asiakkaan tunnistustiedot sekä asiakkaaseen liittyvä data. Näin ollen esimerkiksi Fiver-sovelluslaajennuksessa kaikki pelaajan pisteet ja muut peliin liittyvä tieto on haettava aina aluksi msg-oliolta ja sitten tallennettava siihen.

Seuraavassa esimerkissä haetaan msg-oliosta istunto muuttujaan get_ses()-metodilla ja tämän jälkeen muutetaan istunnon muuttujan arvo numeroksi 1.

```
ses = get_ses( msg )

ses[:muuttuja].set(msg,1)
```

Muuttuja :muuttuja olisi tässä tapauksessa määritelty values.yaml-tiedostossa ja sidottu gui/ohjelma.yaml-tiedostossa ja näin ollen nähtävissä heti metodin läpikäynnin jälkeen selaimessa. Muuttuja yllä olevassa esimerkissä voi olla siis mikä tahansa komponentin lukuarvo tai merkkijono. Käytännön esimerkki aiheesta löytyy luvusta 5.3.3, jossa esitellään submit_ones()-metodi.

Aiemmin totesin, että huijaaminen ilman varokeinoja on triviaalia. Tämän johdosta jokaisen pelaajan istunnossa säilytetään osia tiedoista ”tuplana”, osa on tarkoitettu asiakkaalle näytettäväksi ja osa pidetään aina palvelimella. Esimerkiksi kun heitetään noppaa, tallennetaan tulos muuttujiin

true_die_luku ja die_luku. Varsinaiset pisteiden laskemiset suoritetaan aina true_muuttujilla, koska niitä ei ole voitu vääristellä.

5.3.2 *Käytetyt palvelinpuolen metodit*

Fiver-sovelluslaajennuksessa on käytetty kolmea RSence-järjestelmän palvelinpuolen metodia: init_ses(), restore_ses() ja idle().

init_ses()

init_ses() metodi käynnistetään automaattisesti, kun uusi asiakas on havaittu.

restore_ses()

restore_ses() metodi käynnistetään, kun havaitaan, että asiakkaalla on jo vanha istunto. Näin voitaisiin palauttaa vanha istunto. Fiver-sovelluslaajennuksessa uudelleenlataaminen johtaa pelin palaamisen aloitustilanteeseen.

idle()

idle() ajatetaan joka kerta, kun dataa synkronoidaan asiakkaan ja palvelimen välillä, tai joka kerta, kun asiakas ottaa yhteyden. Fiver-sovelluslaajennuksessa jokaisen idle()-ajon yhteydessä sijoitetaan palvelinpuolen varmennetut kokonaispisteet asiakaspuolen kokonaispisteisiin. Tämän tarkoituksena on se, että kokonaispisteet näkyvät asiakkaalla aina oikeina, vaikka huijausta olisi yritetty.

5.3.3 *submit_ones*

Seuraavana esitelty yksi Fiver-sovelluslaajennuksen pisteenlaskumetodeista, metodin tehtävänä on laskea, kuinka monta nopista näyttää silmälukua yksi. Luvussa 5.2.2 on käyty läpi, kuinka kyseinen metodi on sidottu graafiseen käyttöliittymään values.yaml-tiedostossa.

```
def submit_ones(msg,value)
```

Summan nollaus.

```
ones = 0
```

Haettaan muuttujaan ses istuntokohtaiset tiedot msg-oliosta:

```
ses = get_ses(msg)
```

Tarkistetaan, että noppia on heitetty, eli ei yritetä antaa edellisen kierroksen noppia pisteinä.

```
if ses['true_roll_times'] != 0
```

Tarkistetaan jokaisen nopan kohdalla, onko sen silmäluku yksi. Jos on, lisätään luku summaan ones.

```
if ses['true_die_one'] == 1
  ones += ses['true_die_one']
end
if ses['true_die_two'] == 1
  ones += ses['true_die_two']
end
if ses['true_die_three'] == 1
  ones += ses['true_die_three']
end
if ses['true_die_four'] == 1
  ones += ses['true_die_four']
end
if ses['true_die_five'] == 1
  ones += ses['true_die_five']
end
```

Nollataan noppien lukitukset sekä noppien luvut:

```
reset_keeps(msg)
reset_dice(msg)
```

Päivitetään pisteitä:

```
update_upper_score(msg,ones)
ses[:ones].set(msg,ones)
else
```

Jos noppia ei ole vielä heitetty, niin ei tehdä mitään:

```
        ses[:submit_ones].set( msg, 0)
      end
    end
```

6 INTERNET HUOMENNA

Tulevaisuuden ennustaminen tietotekniikan saralla on erityisen vaikeaa, mutta poikkeuksen tekevät tulevat Internet-tekniikat. Uusien Internet-standardien tekeminen on hidasta, joten tulevaisuuden tekniikat ovat jo nyt olleet useita vuosia kehitysasteella. Näistä tärkein uudistus on HTML5, jonka määrittelyä on valmisteltu jo puolen vuosikymmenen ajan. HTML5 määrittelyssä HTML-merkintäkieleen saadaan uusia elementtejä sekä DOM:in toiminnallisuutta laajennetaan.

Nykyisistä verkkoselaimista löytyy jo tuki osaan HTML5:n uusista mahdollisuuksista, joten niiden käyttäminen ei enää ole kokeilujen asteella, vaan HTML5:llä voidaan jo tehdä verkkosivujakin. CSS3:a ollaan myös julkaisemassa samoihin aikoihin ja myös sitä tuetaan, mutta vaihtelevilla toteutuksilla suurimmissa verkkoselaimissa.

Suurin ongelma HTML5:n ja CSS3:n käyttämisessä ovat vanhemmat verkkoselaimet, jotka ovat vielä paljon käytössä. Osa näistä selaimista ei tue vielä standardien mukaisesti edes CSS1:tä puhumattakaan CSS2:sta. Varsinkin Microsoftin Internet Explorer on ollut pahamaineisen kuuluisa epästandardin mukaisista ratkaisuistaan. Internet Explorer on vuosien saatossa menettänyt markkina-asemaansa niin, että myös Microsoft on joutunut reagoimaan suurimpiin puutteisiin selaimissaan ja lähitulevaisuudessa julkaistava Internet Explorer 9 tukeekin jo varsin kattavasti HTML5:n ja CSS3:n määrittelemiä standardeja.

6.1 HTML5

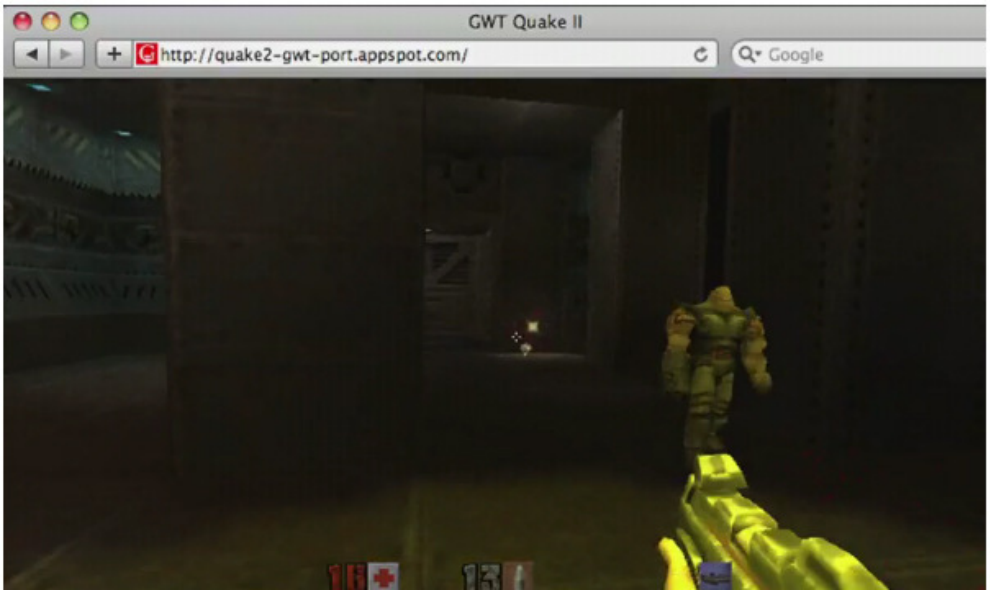
HTML5 koostuu kolmesta eri tekniikasta: HTML:stä, CSS:stä sekä JavaSript API:sta, joita on laajennettu vastaamaan paremmin nykypäivän tarpeita. HTML5:ssä spesifikaatiossa laajennetaan DOM:n toimintaa, jonka ansiosta voidaan toteuttaa esimerkiksi mediasoittimia ja työpöydältä tuttu vedä ja pudota (*drag and drop)*. [23.]

Tällaistä toiminnallisuutta on ennen saatu vain asentamalla verkkoselaimeen laajennuksia, kuten Adoben Flash, mutta nyt ne ovat saatavilla suoraan verkkoselaimen kautta. HTML5:n tuomia mahdollisuuksia onkin käytetty aseena varsinkin Flashiä kohtaan ja varsinkin tehovaatimuksia kohtaan. Esimerkiksi Applen toimitusjohtaja Steve Jobs on voimakkaasti ilmaissut kantansa HTML5:n puolesta ja Apple on jo lopettanut mobiililaitteissaan Flashin tukemisen [14].

Seuraavana on esitelty joitakin HTML5:n uusista elementeistä, joiden toiminnallisuus on ennen saavutettu vain selainlaajennusten kautta.

Canvas

Canvas-elementti on resoluutiosta riippuvainen bittikartta, jota voidaan käyttää kuvaajien esittämiseen, pelien grafiikkaan ja muiden visuaalisten elementtien piirtämistä varten. [24.]



Kuva 10. Quake 2-pelin verkkoselaimessa pelattava versio

Yllä olevassa kuvassa 10 on havaintoesitys canvas-elementistä ja HTML5: n tuomista mahdollisuuksista. Quake 2:n HTML5-version on tehnyt Googlen insinööri Joel Webber. [25.]

Video

Video-elementti mahdollistaa videoiden upottamisen verkkosivuille. Aiemmin tämä ei ollut mahdollista ilman verkkoselaimen laajennuksia, kuten esimerkiksi Apple QuickTimea ja Adobe Flashiä. [24.]

Web Workers

Web Workers on standardoitu tapa, jolla verkkoselain voi ajaa JavaScriptiä taustalla [24]. Web Workersit mahdollistavat näin siis muista ohjelmointikielistä tutut säikeet, joilla voidaan suorittaa samanaikaisesti useita eri laskutoimenpiteitä tai verkkopyyntöjä.

6.2 CSS3

CSS3 tuo mukaan useita laajennuksia, joita voidaan käyttää verkkosovellusten graafisen ulkoasun parantamiseen. Aiemmin vastaavanlaisia efektejä luotaessa on jouduttu käyttämään kuvatiedostoja tai selainlaajennuksia kuten Adobe Flashia.

CSS3 sisältää tuen muun muassa omien fonttien käyttämiseen, fonttien varjostamiseen ja div-elementtien rotaatioille.

Varsinkin div-elementtien rotaatiot ovat tervetullut lisä standardiin, koska ne mahdollistavat erilaiset pyörivät elementit, joita ennen ollaan jouduttu luomaan sprite-grafiikan avustuksella, mikä on tarkoittanut sitä, että yhden elementin näyttämiseksi on jouduttu lataamaan suuri määrä kuvia.

Sivullisen näkökulmasta CSS3:n tuomat muutokset voivat näyttää pieniltä, mutta varsinkin verkkosivujen suunnittelijat ovat ottaneet ne avosylin vastaan. CSS3:n uudistusten avulla voidaan tehdä uudenlaisia ulkoasuja, joita pystytään käsittelemään dynaamisesti DOM:n avulla. Näin pidetään sivujen fyysinen koko ja sen tehovaatimukset pienenä, mutta saavutetaan silti aikaisemmin vain selainlaajennusten kautta saatuja ominaisuuksia.

6.3 Selainlaajennusten loppu

Keväällä 2010 Applen toimitusjohtaja Steve Jobs esitti useita vahvoja mielipiteitä Flashiä vastaan. [14] Nämä mielipiteet pätevät suurimmalta osin myös muihin verkkoselainlaajennuksilla toimiiviin RIA-tekniikoihin, kuten Silverlightiin tai JavaFXään.

Jobsin mukaaan verkkoselainlaajennuksilla toimivat tekniikat ovat tulleet tiensä päähän, koska HTML5den tuomat uudistukset mahdollistavat samojen toiminnallisuuksien toteuttamisen standardoiduilla tekniikoilla. Tämä tarkoittaa sitä, että RSencen kaltaisten Ajaxiin pohjautuvien RIA-ohjelmistokehysten, jotka toimivat ilman verkkoselainlaajennuksia, aika on todennäköisesti vasta alkamassa.

Flash ja sillä toteutetut Flex RIA-sovellukset tulevat todennäköisesti vähenemään tulevina vuosina, mutta niiden tämän hetkinen markkinajohtajan asema varmistaa sen, että niiden kokonaan poistuminen ei tule tapahtumaan lyhyessä ajassa.

7 YHTEENVETO

Tämän insinöörityön tavoitteina oli tutustua Internetin kehitykseen viime vuosina ja niihin puutteisiin, joita Ajax ja sen kilpailevat RIA-tekniikat ovat yrittäneet paikata, sekä tarkastella RSence RIA -ohjelmistokehyksen toimivuutta RIA-sovellusten kehittämisessä.

HTML5 sisältää suuren osan toiminnallisuudesta, jota aikaisemmin on saatu vain erillisten verkkoselainlaajennusten kautta. Tämä tulee väistämättä vähentämään niiden käyttöä tulevaisuudessa. Lisäksi niiden tehovaatimukset rajoittavat niiden käyttöä suosiota keräävissä mobiilitietokoneissa. Adobe Flex ja Microsoft Silverlightin kaltaiset verkkoseilainlaajennukseen pohjautuvat RIA-tekniikat tulevat kuitenkin elämään vielä pitkään, johtuen siitä, että niitä hallitsevia kehittäjiä on nyt niin paljon ja niillä tehtyjä RIA-sovelluksia on paljon.

AJAXilla toteutetut RIA-sovellutukset tulevat todennäköisesti yleistymään HTML5:n myötä. Vaikka osa kehityksestä muuttuukin, voidaan nykyistä tietämystä ja tekniikoita hyödyntää myös HTML5:n aikana. RIA-sovellusten määritelmä tulee todennäköisesti myös muuttumaan, kun Internet alkaa hiljalleen muistuttumaan enemmän ja enemmän työpöytäsovellutuksia. Google Chrome OS–käyttöjärjestelmä on hyvä esimerkki Internetin ja työpöydän sulautumisesta. Chrome OS:n sovellukset sijaitsevat suurelta osin Internetissä ja käyttäjän dataa tallennetaan myös Internetiin. Toisin sanoen Chrome OS on kehittynyt verkkoselain.

Tämä tarkoittaa myös, että RSence RIA-ohjelmistokehyksellä voidaan tehdä ohjelmia, jotka toimivat myös tulevaisuudessa. RSence osoittautui käyttökelpoiseksi ohjelmistokehykseksi, mutta tämän hetken dokumentaatiolla on kehitystyön aloittaminen haastavaa. Dokumentaation suurin ongelma on se, että siinä keskitytään lähinnä kuvailemaan ”kovaa” teknistä puolta, mutta ”pehmeä” alustus jää vähälle. Tämä rajoittaa käyttäjäkunnan määrää, sillä kynnys RIA-kehitykseen RSencellä voi olla liian suuri. Kehittäjät kuitenkin ovat valmiita ottamaan vastaan palautetta, joten dokumentaation ongelmat voivat poistua tulevaisuudessa. Lopputyötä tehdessäni olin yhteydessä kehittäjiin epäkohtien ilmaantuessa ja sain vastaukset nopeasti. Kehittäjät myös laajensivat RSence.org-sivun esimerkkejä antamani palautteen pohjalta.

RSencen tulevaisuus riippuukin hyvin paljon sen kehittäjien työpanoksesta ja siitä, tuleeko sille lisää aktiivisia käyttäjiä, jotka voivat omalla panoksellaan laajentaa ohjelmistokehyksen toiminnallisuutta sekä luoda yleishyödyllisiä sovelluslaajennuksia.

RSencen vahvuuksiin kuuluu se, että sillä toteutetut ohjelmat toimivat nopeasti eikä niiden kehittäminen ole kovinkaan vaativaa teknisesti, kunhan pystyy ylittämään alun ongelmat. Omien komponenttien tekeminen vaati tuntemusta JavaScriptistä, mutta muuten Yaml-merkintäkielellä tehtyjen ulkoasujen luominen on varsin nopeaa.

VIITELUETTELO

[1] Sutton Chris. University of California. *Internet Began 35 Years Ago at Ucla; Forum to Mark Anniversary Oct.29* [verkkodokumentti]. 14.9.2004 [viitattu 2.6.2010]. Saatavissa: http://newsroom.ucla.edu/portal/ucla/Internet-Began-35-Years-Ago-at-5464.aspx?RelNum=5464.

[2] RSence Project. *Portaalin etusivu* [verkkodokumentti]. [viitattu 3.6.2010]. Saatavissa: http://rsence.org/trac/.

[3] Wikipedia. *HTML* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 3.6.2010]. Saatavissa: http://fi.wikipedia.org/wiki/HTML.

[4] Duffy, Scott. How to Do Everything with JavaScript. California: McGraw-Hill/Osborne. 2003.

[5] Internet World Stats. *Internet Usage Statistics* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 1.6.2010]. Saatavissa: http://www.internetworldstats.com/stats.htm.

[6] Tilastokeskus. *Internetin käyttäjiä enemmän kuin vuosi sitten* [verkkodokumentti]. Julkaistu 25.8.2008 [viitattu 1.6.2010]. Saatavissa: http://www.stat.fi/til/sutivi/2008/sutivi_2008_2008-08-25_tie_001.html.

[7] Holdener, Anthony T*. Ajax: The Definite Guide.* California: O'Reilly Media. 2008.

[8] Dep, Brijesh. Jax Magazine. *Rich Internet Applications : A Look Into Available Technology Choices* [verkkodokumentti] Julkaisuaika tuntematon. [viitattu 10.9.2010]. Saatavissa: http://www.jaxmag.com/itr/online_artikel/psecom,id,828,nodeid,147.html.

[9] Duhl, Joshua. IDC. *Impact of RIA's* [verkkodokumentti] 11.2003. [viitattu 10.8.2010]. Saatavissa: http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf.

[10] Ullman, Chris. Wrox. *What is Ajax?* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 10.7.2010]. Saatavissa: http://www.wrox.com/WileyCDA/Section/id-303217.html.

[11] Wikipedia. *Rich Internet application* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 2.7.2010]. Saatavissa: http://en.wikipedia.org/wiki/Rich_Internet_application.

[12] Alexa. *Top Sites*[verkkodokumentti]. Julkaisuaika tuntematon [viitattu 5.8.2010]. Saatavissa: http://www.alexa.com/topsites.

[13] Adobe*. Flex-yleiskuva* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 12.7.2010]. Saatavissa: http://www.adobe.com/fi/products/flex/overview/.

[14] Jobs, Steve. Apple. *Thoughts on Flash* [verkkodokumentti]. 5.2010 [viitattu 2.8.2010]. Saatavissa: http://www.apple.com/hotnews/thoughts-on-flash/.

[15] Wikipedia. *Java FX* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 5.7.2010]. Saatavissa: http://en.wikipedia.org/wiki/Java_FX.

[16] Wikipedia. *Microsoft Silverlight* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 5.7.2010]. Saatavissa: http://en.wikipedia.org/wiki/Microsoft_Silverlight.

[17] RSence. *Server API* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 10.8.2010]. Saatavissa: http://rsence.org/server_api/index.html.

[18] Wikipedia. *Ruby* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 2.6.2010]. Saatavissa: http://fi.wikipedia.org/wiki/Ruby.

[19] Cooper, Pete. *Beginning Ruby: From Novice to Professional, Second Edition*. New York: Apress. 2009.

[20] RSence. *Plugin Bundles* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 10.8.2010]. Saatavissa: http://rsence.org/server_api/file.PluginBundles.html.

[21] RSence. *Client API* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 12.8.2010]. Saatavissa: http://rsence.org/client_api/.

[22] RSence. *Message* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 12.8.2010]. Saatavissa: http://rsence.org/server_api/RSence/Message.html.

[23] Pilgrim, Mark. Dive into HTML5. *Dive into HTML5* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 10.8.2010]. Saatavissa: http://diveintohtml5.org/.

[24] Pilgrim, Mark. Dive into HTML5. *Detecting HTML5 Features* [verkkodokumentti]. Julkaisuaika tuntematon [viitattu 10.8.2010]. Saatavissa: http://diveintohtml5.org/detect.html.

[25] Cromwell, Ray. Timefire. *GwtQuake: Taking the Web to the Next Level* [verkkodokumentti] 1.4.2010 [viitattu 11.8.2010]. Saatavissa. http://timepedia.blogspot.com/2010/04/gwtquake-taking-web-to-next-level.html.

Liite 1. Fiver-sovelluslaajennuksen lähdekoodit

gui/fiver.yaml

```
type: GUITree
version: 0.6
# Default dependencies
# This example uses only standard components
# Please refer to client side API for standard components available at
# http://rsence.org/
dependencies:
- default_theme
- controls
- fiver
class: RSence.GUIApp
options:
  # A small priority number makes the onIdle calls more frequent,
  # A large priority number makes the onIdle colls less frequent.
  priority: 20
  # A short description or a name of the application. Visible in task managers and such.
  label: Fiver
subviews:
  - class: HWindow
    rect: [0, 0, 800, 650]
    options:
      closeButton: true
      noResize: true
      label: Fiver
    subviews:
   #UPPER SECTION
      - class: HStringView
        rect: [20, 5, 125, 20]
        options:
          value: "<b>UPPER SECTION</b>"
      #ONES
      - class: HClickButton
        rect: [35, 30, 85, 25]
        bind: :values.submit_ones
        options:
          label: "Ones"
      - class: HStringView
        rect: [135, 32, 85, 25]
        bind: :values.ones
      #TWOS
      - class: HClickButton
        rect: [35, 55, 85, 25]
        bind: :values.submit_twos
        options:
          label: "Twos"
      - class: HStringView
        rect: [135, 57, 85, 25]
        bind: :values.twos
      #THREES
      - class: HClickButton
```

```
  rect: [35, 80, 85, 25]
  bind: :values.submit_threes
  options:
    label: "Threes"
- class: HStringView
  rect: [135, 82, 85, 25]
  bind: :values.threes
#FOURS
- class: HClickButton
  rect: [35, 105, 85, 25]
  bind: :values.submit_fours
  options:
    label: "Fours"
- class: HStringView
  rect: [135, 107, 85, 25]
  bind: :values.fours
#FIVES
- class: HClickButton
  rect: [35, 130, 85, 25]
  bind: :values.submit_fives
  options:
    label: "Fives"
- class: HStringView
  rect: [135, 132, 85, 25]
  bind: :values.fives
#SIXES
- class: HClickButton
  rect: [35, 155, 85, 25]
  bind: :values.submit_sixes
  options:
    label: "Sixes"
- class: HStringView
  rect: [135, 157, 85, 25]
  bind: :values.sixes
#UPPER SECTION SCORES
#upper score
- class: HStringView
  rect: [35, 190, 85, 25]
  options:
    value: "Total score:"
- class: HStringView
  rect: [190, 190, 85, 25]
  bind: :values.upper_score
#bonus
- class: HStringView
  rect: [35, 215, 125, 25]
  options:
    #jos 63 tai yli -> 50 pistettä
    value: "Bonus:"
- class: HStringView
  rect: [190, 215, 85, 25]
  bind: :values.bonus
#total upper score
- class: HStringView
  rect: [35, 240, 125, 25]
```

```
  options:
    value: "Total of upper section:"
- class: HStringView
  rect: [190, 240, 85, 25]
  bind: :values.total_upper_score
#LOWER SECTION
- class: HStringView
  rect: [20, 270, 125, 25]
  options:
    value: "<b>LOWER SECTION</b>"
#PAIR
- class: HClickButton
  rect: [35, 295, 85, 25]
  bind: :values.submit_pair
  options:
    label: "Pair"
- class: HStringView
  rect: [135, 297, 85, 25]
  bind: :values.pair
#TWO PAIRS
- class: HClickButton
  rect: [35, 320, 85, 25]
  bind: :values.submit_two_pairs
  options:
    label: "Two Pairs"
- class: HStringView
  rect: [135, 322, 85, 25]
  bind: :values.two_pairs
#THREE OF A KIND
- class: HClickButton
  rect: [35, 345, 85, 25]
  bind: :values.submit_three_of_a_kind
  options:
    label: "3 of a kind"
- class: HStringView
  rect: [135, 347, 85, 25]
  bind: :values.three_of_a_kind
#FOUR OF A KIND
- class: HClickButton
  rect: [35, 370, 85, 25]
  bind: :values.submit_four_of_a_kind
  options:
    label: "4 of a kind"
- class: HStringView
  rect: [135, 372, 85, 25]
  bind: :values.four_of_a_kind
#FULL HOUSE
- class: HClickButton
  rect: [35, 395, 85, 25]
  bind: :values.submit_full_house
  options:
    label: "Full House"
- class: HStringView
  rect: [135, 397, 85, 25]
  bind: :values.full_house
```

```
#SMALL STRAIGHT
- class: HClickButton
  rect: [35, 420, 85, 25]
  bind: :values.submit_s_straight
  options:
    #12345
    label: "S. Straight"
- class: HStringView
  rect: [135, 422, 85, 25]
  bind: :values.s_straight
#LARGE STRAIGHT
- class: HClickButton
  rect: [35, 445, 85, 25]
  bind: :values.submit_l_straight
  options:
    #23456
    label: "L. Straight"
- class: HStringView
  rect: [135, 447, 85, 25]
  bind: :values.l_straight
#fiver
- class: HClickButton
  rect: [35, 470, 85, 25]
  bind: :values.submit_fiver
  options:
    #50 pistettä
    label: "Fiver"
- class: HStringView
  rect: [135, 472, 85, 25]
  bind: :values.fiver
#CHANCE
- class: HClickButton
  rect: [35, 495, 85, 25]
  bind: :values.submit_chance
  options:
    label: "Chance"
- class: HStringView
  rect: [135, 497, 85, 25]
  bind: :values.chance
#LOWER SECTION SCORES
- class: HStringView
  rect: [35, 530, 125, 25]
  options:
    value: "Total of lower section:"
- class: HStringView
  rect: [190, 530, 85, 25]
  bind: :values.total_lower_score
- class: HStringView
  rect: [35, 555, 125, 25]
  options:
    value: "Grand Total:"
- class: HStringView
  rect: [190, 555, 85, 25]
  bind: :values.grand_total
#DICE
```

```
- class: Hfiver
  rect: [275, 175, 60, 60]
  bind: :values.die_one
- class: HCheckbox
  rect: [275, 240, 60, 25]
  bind: :values.die_one_check
  options:
    label: "Keep"
- class: Hfiver
  rect: [350, 175, 60, 60]
  bind: :values.die_two
- class: HCheckbox
  rect: [350, 240, 60, 25]
  bind: :values.die_two_check
  options:
    label: "Keep"
- class: Hfiver
  rect: [425, 175, 60, 60]
  bind: :values.die_three
- class: HCheckbox
  rect: [425, 240, 60, 25]
  bind: :values.die_three_check
  options:
    label: "Keep"
- class: Hfiver
  rect: [500, 175, 60, 60]
  bind: :values.die_four
- class: HCheckbox
  rect: [500, 240, 60, 25]
  bind: :values.die_four_check
  options:
    label: "Keep"
- class: Hfiver
  rect: [575, 175, 60, 60]
  bind: :values.die_five
- class: HCheckbox
  rect: [575, 240, 60, 25]
  bind: :values.die_five_check
  options:
    label: "Keep"
- class: HClickButton
  rect: [410, 280, 90, 25]
  bind: :values.roll
  options:
    label: "Roll"
- class: HClickButton
  rect: [700, 5, 85, 25]
  bind: :values.reset_game
  options:
    label: "New Game"
```

fiver.rb

```
class Fiver < GUIPlugin

 # Session initialization, whenever a new client is detected this method will be invoked
 def init_ses( msg )
  super
  ses = get_ses(msg)
  ses['true_upper_score'] = 0
  ses['true_bonus'] = 0
  ses['true_bonus_set'] = 0
  ses['true_total_upper_score'] = 0
  ses['true_total_lower_score'] = 0
  ses['true_grand_total'] = 0
  ses['true_roll_times'] = 0
  ses['true_die_one'] = 0
  ses['true_die_two'] = 0
  ses['true_die_three'] = 0
  ses['true_die_four'] = 0
  ses['true_die_five'] = 0
 end
 # Restore session will be invoked when an old session in client is detected
 def restore_ses( msg )
  super
  ses = get_ses(msg)
  ses['true_upper_score'] = 0
  ses['true_bonus'] = 0
  ses['true_bonus_set'] = 0
  ses['true_total_upper_score'] = 0
  ses['true_total_lower_score'] = 0
  ses['true_grand_total'] = 0
  ses['true_roll_times'] = 0
  ses['true_die_one'] = 0
  ses['true_die_two'] = 0
  ses['true_die_three'] = 0
  ses['true_die_four'] = 0
  ses['true_die_five'] = 0
 end

 # Runs every time during the value synchronization (every time the client connects)
 # Does not need super
 # anti cheat routines could be implemented here, I will settle with grand_total
 def idle( msg )
  ses = get_ses( msg )
  ses[:grand_total].set(msg,ses['true_grand_total'])
  return true
 end

 def reset_game(msg,value)
  ses = get_ses(msg)
  #Resetting true variables
  ses['true_upper_score'] = 0
  ses['true_bonus'] = 0
  ses['true_bonus_set'] = 0
  ses['true_total_upper_score'] = 0
```

```
  ses['true_total_lower_score'] = 0
  ses['true_grand_total'] = 0
  ses['true_roll_times'] = 0
  ses['true_die_one'] = 0
  ses['true_die_two'] = 0
  ses['true_die_three'] = 0
  ses['true_die_four'] = 0
  ses['true_die_five'] = 0

  ses.each do |key,value|
    if value.respond_to? :value_id
      ses[key ].set(msg,0)
    end
  end

  return true

end

def update_upper_score(msg,score)
  ses = get_ses(msg)
  ses['true_upper_score'] += score
  ses[:upper_score].set(msg,ses['true_upper_score'])
  if ses['true_bonus_set'] == 0
    ses['true_bonus'] = ses['true_upper_score'] > 62 ? 50 : 0
    ses['true_bonus_set'] = ses['true_bonus'] == 50 ? 1 : 0
    score += ses['true_bonus']
  end
  ses[:bonus].set(msg,ses['true_bonus'])
  ses['true_total_upper_score'] += score
  ses[:total_upper_score].set(msg,ses['true_total_upper_score'])
  ses['true_grand_total'] +=score
  ses[:grand_total].set(msg,ses['true_grand_total'])
end

def update_lower_score(msg,score)
  ses = get_ses(msg)
  ses['true_total_lower_score'] += score
  ses[:total_lower_score].set(msg,ses['true_total_lower_score'])
  ses['true_grand_total'] +=score
  ses[:grand_total].set(msg,ses['true_grand_total'])
end

#resets keep buttons after score is submitted
def reset_keeps(msg)
  ses = get_ses(msg)
  ses[:die_one_check].set(msg,0)
  ses[:die_two_check].set(msg,0)
  ses[:die_three_check].set(msg,0)
  ses[:die_four_check].set(msg,0)
  ses[:die_five_check].set(msg,0)
end

def die_check(msg,value)
  ses = get_ses(msg)
```

```
  if ses['true_roll_times'] == 0
    reset_keeps(msg)
  else
    return true
  end
 end

 #resets dice after score is submitted
 def reset_dice(msg)
  ses = get_ses(msg)
  ses[:die_one].set( msg,"0")
  ses[:die_two].set( msg,"0")
  ses[:die_three].set( msg,"0")
  ses[:die_four].set( msg,"0")
  ses[:die_five].set( msg,"0")
  ses['true_roll_times'] = 0
  ses[:roll].set( msg, 0)
 end

 def roll(msg,value)
  ses = get_ses( msg )

  if ses['true_roll_times'] <= 2
    #Roll each die which is not checked to be kept
    if ses[:die_one_check].data() != true
      ses['true_die_one'] = rand(6)+1
      ses[:die_one].set( msg,ses['true_die_one'])
    end
    if ses[:die_two_check].data() != true
      ses['true_die_two'] = rand(6)+1
      ses[:die_two].set( msg,ses['true_die_two'])
    end
    if ses[:die_three_check].data() != true
      ses['true_die_three'] = rand(6)+1
      ses[:die_three].set( msg,ses['true_die_three'])
    end
    if ses[:die_four_check].data() != true
      ses['true_die_four'] = rand(6)+1
      ses[:die_four].set( msg,ses['true_die_four'])
    end
    if ses[:die_five_check].data() != true
      ses['true_die_five'] = rand(6)+1
      ses[:die_five].set( msg,ses['true_die_five'])
    end
  end

  ses['true_roll_times']+=1
  #Makes the roll button re-clickable
  if ses['true_roll_times'] < 3
    ses[:roll].set( msg, 0)
  end
  return true
 end
#BELOW THIS ARE ALGO'S FOR fiver SCORING LOGIC
 def submit_ones(msg,value)
```

```
  ones = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 1
      ones += ses['true_die_one']
    end
    if ses['true_die_two'] == 1
      ones += ses['true_die_two']
    end
    if ses['true_die_three'] == 1
      ones += ses['true_die_three']
    end
    if ses['true_die_four'] == 1
      ones += ses['true_die_four']
    end
    if ses['true_die_five'] == 1
      ones += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,ones)
    ses[:ones].set(msg,ones)
  else
    ses[:submit_ones].set( msg, 0)
  end

end

def submit_twos(msg,value)
  twos = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 2
      twos += ses['true_die_one']
    end
    if ses['true_die_two'] == 2
      twos += ses['true_die_two']
    end
    if ses['true_die_three'] == 2
      twos += ses['true_die_three']
    end
    if ses['true_die_four'] == 2
      twos += ses['true_die_four']
    end
    if ses['true_die_five'] == 2
      twos += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,twos)
    ses[:twos].set(msg,twos)
  else
    ses[:submit_twos].set( msg, 0)
  end
end
```

```
def submit_threes(msg,value)
  threes = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 3
      threes += ses['true_die_one']
    end
    if ses['true_die_two'] == 3
      threes += ses['true_die_two']
    end
    if ses['true_die_three'] == 3
      threes += ses['true_die_three']
    end
    if ses['true_die_four'] == 3
      threes += ses['true_die_four']
    end
    if ses['true_die_five'] == 3
      threes += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,threes)
    ses[:threes].set(msg,threes)
  else
    ses[:submit_threes].set( msg, 0)
  end
end

def submit_fours(msg,value)
  fours = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 4
      fours += ses['true_die_one']
    end
    if ses['true_die_two'] == 4
      fours += ses['true_die_two']
    end
    if ses['true_die_three'] == 4
      fours += ses['true_die_three']
    end
    if ses['true_die_four'] == 4
      fours += ses['true_die_four']
    end
    if ses['true_die_five'] == 4
      fours += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,fours)
    ses[:fours].set(msg,fours)
  else
    ses[:submit_fours].set( msg, 0)
  end
```

```
end

def submit_fives(msg,value)
  fives = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 5
      fives += ses['true_die_one']
    end
    if ses['true_die_two'] == 5
      fives += ses['true_die_two']
    end
    if ses['true_die_three'] == 5
      fives += ses['true_die_three']
    end
    if ses['true_die_four'] == 5
      fives += ses['true_die_four']
    end
    if ses['true_die_five'] == 5
      fives += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,fives)
    ses[:fives].set(msg,fives)
  else
    ses[:submit_fives].set( msg, 0)
  end
end

def submit_sixes(msg,value)
  sixes = 0
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    if ses['true_die_one'] == 6
      sixes += ses['true_die_one']
    end
    if ses['true_die_two'] == 6
      sixes += ses['true_die_two']
    end
    if ses['true_die_three'] == 6
      sixes += ses['true_die_three']
    end
    if ses['true_die_four'] == 6
      sixes += ses['true_die_four']
    end
    if ses['true_die_five'] == 6
      sixes += ses['true_die_five']
    end
    reset_keeps(msg)
    reset_dice(msg)
    update_upper_score(msg,sixes)
    ses[:sixes].set(msg,sixes)
  else
    ses[:submit_sixes].set( msg, 0)
```

```
    end
  end

  def submit_pair(msg,value)
    ses = get_ses(msg)
    if ses['true_roll_times'] != 0
      pair = 0
      pair_array = []
      pair_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three'] <<
ses['true_die_four'] << ses['true_die_five']
      pair = pair_array.count(1) >= 2 ? 2 : 0
      pair = pair_array.count(2) >= 2 ? 4 : pair != 0 ? pair : 0
      pair = pair_array.count(3) >= 2 ? 6 : pair != 0 ? pair : 0
      pair = pair_array.count(4) >= 2 ? 8 : pair != 0 ? pair : 0
      pair = pair_array.count(5) >= 2 ? 10 : pair != 0 ? pair : 0
      pair = pair_array.count(6) >= 2 ? 12 : pair != 0 ? pair : 0
      update_lower_score(msg,pair)

      reset_keeps(msg)
      reset_dice(msg)
      ses[:pair].set(msg,pair)
    else
      ses[:submit_pair].set( msg, 0)
    end
  end

  def submit_two_pairs(msg,value)
    ses = get_ses(msg)
    if ses['true_roll_times'] != 0
      two_pairs = 0
      pair_one = 0
      pair_two = 0
      two_pairs_array = []
      two_pairs_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three']
<< ses['true_die_four'] << ses['true_die_five']

      pair_one = two_pairs_array.count(1) >= 2 ? 2 : 0
      pair_one = two_pairs_array.count(2) >= 2 ? 4 : pair_one != 0 ? pair_one : 0
      pair_one = two_pairs_array.count(3) >= 2 ? 6 : pair_one != 0 ? pair_one : 0
      pair_one = two_pairs_array.count(4) >= 2 ? 8 : pair_one != 0 ? pair_one : 0
      pair_one = two_pairs_array.count(5) >= 2 ? 10 : pair_one != 0 ? pair_one : 0
      pair_one = two_pairs_array.count(6) >= 2 ? 12 : pair_one != 0 ? pair_one : 0

      pair_two = two_pairs_array.count(1) >= 2 ? pair_one != 2 ? 2 : 0 : 0
      pair_two = two_pairs_array.count(2) >= 2 ? pair_one != 4 ? 4 : pair_two != 0 ?
pair_two : 0 : pair_two != 0 ? pair_two : 0
      pair_two = two_pairs_array.count(3) >= 2 ? pair_one != 6 ? 6 : pair_two != 0 ?
pair_two : 0 : pair_two != 0 ? pair_two : 0
      pair_two = two_pairs_array.count(4) >= 2 ? pair_one != 8 ? 8 : pair_two != 0 ?
pair_two : 0 : pair_two != 0 ? pair_two : 0
      pair_two = two_pairs_array.count(5) >= 2 ? pair_one != 10 ? 10 : pair_two != 0 ?
pair_two : 0 : pair_two != 0 ? pair_two : 0
      #pair_two can't be six because pairs have to be different
      if (pair_one != 0) && (pair_two !=0)
        two_pairs = pair_one + pair_two
```

```
    end
    update_lower_score(msg,two_pairs)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:two_pairs].set(msg,two_pairs)
   else
    ses[:submit_two_pairs].set( msg, 0)
   end

  end


  def submit_three_of_a_kind(msg,value)
   ses = get_ses(msg)
   if ses['true_roll_times'] != 0
    toak = 0
    toak_array = []

    toak_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three'] <<
ses['true_die_four'] << ses['true_die_five']

    if toak_array.count(6) >= 3
     toak = 18
    elsif toak_array.count(5) >= 3
     toak = 15
    elsif toak_array.count(4) >= 3
     toak = 12
    elsif toak_array.count(3) >= 3
     toak = 9
    elsif toak_array.count(2) >= 3
     toak = 6
    elsif toak_array.count(1) >= 3
     toak = 3
    else
     toak = 0
    end

    update_lower_score(msg,toak)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:three_of_a_kind].set(msg,toak)
   else
    ses[:submit_three_of_a_kind].set( msg, 0)
   end
  end

  def submit_four_of_a_kind(msg,value)
   ses = get_ses(msg)
   if ses['true_roll_times'] != 0
    foak = 0
    foak_array = []

    foak_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three'] <<
ses['true_die_four'] << ses['true_die_five']
```

```
    if foak_array.count(6) >= 4
      foak = 24
    elsif foak_array.count(5) >= 4
      foak = 20
    elsif foak_array.count(4) >= 4
      foak = 16
    elsif foak_array.count(3) >= 4
      foak = 12
    elsif foak_array.count(2) >= 4
      foak = 8
    elsif foak_array.count(1) >= 4
      foak = 4
    else
      foak = 0
    end

    update_lower_score(msg,foak)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:four_of_a_kind].set(msg,foak)
  else
    ses[:submit_four_of_a_kind].set( msg, 0)
  end
 end

 def submit_full_house(msg,value)
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    full_house = 0
    full_house_array = []
    f_h_three = 0
    f_h_pair = 0
    full_house_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three']
<< ses['true_die_four'] << ses['true_die_five']

    if full_house_array.count(6) == 3
      f_h_three = 18
    elsif full_house_array.count(5) == 3
      f_h_three = 15
    elsif full_house_array.count(4) == 3
      f_h_three = 12
    elsif full_house_array.count(3) == 3
      f_h_three = 9
    elsif full_house_array.count(2) == 3
      f_h_three = 6
    elsif full_house_array.count(1) == 3
      f_h_three = 3
    else
      f_h_three = 0
    end

    if full_house_array.count(6) == 2
      f_h_two = 12
    elsif full_house_array.count(5) == 2
      f_h_two = 10
```

```
    elsif full_house_array.count(4) == 2
      f_h_two = 8
    elsif full_house_array.count(3) == 2
      f_h_two = 6
    elsif full_house_array.count(2) == 2
      f_h_two = 4
    elsif full_house_array.count(1) == 2
      f_h_two = 2
    else
      f_h_two  = 0
    end

    if (f_h_three != 0)&&(f_h_two != 0)
      full_house = f_h_three + f_h_two
    else
      full_house = 0
    end

    update_lower_score(msg,full_house)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:full_house].set(msg,full_house)
  else
    ses[:submit_full_house].set( msg, 0)
  end

 end


 def submit_s_straight(msg,value)
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    s_straight = 0
    s_straight_array = []
    s_straight_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three']
<< ses['true_die_four'] << ses['true_die_five']
    if (s_straight_array.count(1) == 1)&&(s_straight_array.count(2) ==
1)&&(s_straight_array.count(3) == 1)&&(s_straight_array.count(4) ==
1)&&(s_straight_array.count(5) == 1)
      s_straight = 20
    end
    update_lower_score(msg,s_straight)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:s_straight].set(msg,s_straight)
  else
    ses[:submit_s_straight].set( msg, 0)
  end
 end

 def submit_l_straight(msg,value)
  ses = get_ses(msg)
  if ses['true_roll_times'] != 0
    l_straight = 0
    l_straight_array = []
```

```
    l_straight_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three']
<< ses['true_die_four'] << ses['true_die_five']
    if (l_straight_array.count(2) == 1)&&(l_straight_array.count(3) ==
1)&&(l_straight_array.count(4) == 1)&&(l_straight_array.count(5) ==
1)&&(l_straight_array.count(6) == 1)
      l_straight = 25
    end
    update_lower_score(msg,l_straight)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:l_straight].set(msg,l_straight)
   else
    ses[:submit_l_straight].set( msg, 0)
   end
  end

  def submit_fiver(msg,value)
   ses = get_ses(msg)
   if ses['true_roll_times'] != 0
    fiver = 0
    fiver_array = []
    fiver_array << ses['true_die_one'] << ses['true_die_two'] << ses['true_die_three'] <<
ses['true_die_four'] << ses['true_die_five']
    if (fiver_array.count(1) == 5)||(fiver_array.count(2) == 5)||(fiver_array.count(3) ==
5)||(fiver_array.count(4) == 5)||(fiver_array.count(5) == 5) || (fiver_array.count(6) == 5)
      fiver = 50
    end
    update_lower_score(msg,fiver)
    reset_keeps(msg)
    reset_dice(msg)
    ses[:fiver].set(msg,fiver)
   else
    ses[:submit_fiver].set( msg, 0)
   end
  end

  def submit_chance(msg,value)
   ses = get_ses(msg)
   if ses['true_roll_times'] != 0
    chance = ses['true_die_one'] + ses['true_die_two'] + ses['true_die_three'] +
ses['true_die_four'] + ses['true_die_five']
    reset_keeps(msg)
    reset_dice(msg)
    update_lower_score(msg,chance)
    ses[:chance].set(msg,chance)
   else
    ses[:submit_chance].set( msg, 0)
   end
  end

end
```

values.yaml

```
#UPPER SECTION
:ones:
  :value: 0
:submit_ones:
  :value: 0
  :responders:
    - :method: submit_ones

:twos:
  :value: 0
:submit_twos:
  :value: 0
  :responders:
    - :method: submit_twos

:threes:
  :value: 0
:submit_threes:
  :value: 0
  :responders:
    - :method: submit_threes

:fours:
  :value: 0
:submit_fours:
  :value: 0
  :responders:
    - :method: submit_fours

:fives:
  :value: 0
:submit_fives:
  :value: 0
  :responders:
    - :method: submit_fives

:sixes:
  :value: 0
:submit_sixes:
  :value: 0
  :responders:
    - :method: submit_sixes

#UPPER SECTION SCORES
:upper_score:
  :value: 0
:bonus:
  :value: 0
:total_upper_score:
  :value: 0

#LOWER SECTION
:pair:
```

```
  :value: 0
:submit_pair:
  :value: 0
  :responders:
    - :method: submit_pair
:two_pairs:
  :value: 0
:submit_two_pairs:
  :value: 0
  :responders:
    - :method: submit_two_pairs
:three_of_a_kind:
  :value: 0
:submit_three_of_a_kind:
  :value: 0
  :responders:
    - :method: submit_three_of_a_kind
:four_of_a_kind:
  :value: 0
:submit_four_of_a_kind:
  :value: 0
  :responders:
    - :method: submit_four_of_a_kind
:full_house:
  :value: 0
:submit_full_house:
  :value: 0
  :responders:
    - :method: submit_full_house
:s_straight:
  :value: 0
:submit_s_straight:
  :value: 0
  :responders:
    - :method: submit_s_straight
:l_straight:
  :value: 0
:submit_l_straight:
  :value: 0
  :responders:
    - :method: submit_l_straight
:fiver:
  :value: 0
:submit_fiver:
  :value: 0
  :responders:
    - :method: submit_fiver
:chance:
  :value: 0
:submit_chance:
  :value: 0
  :responders:
    - :method: submit_chance
```

```
#LOWER SECTION SCORES
:grand_total:
  :value: 0
:total_lower_score:
  :value: 0

#DICES
:die_one:
  :value: "0"
:die_one_check:
  :value: 0
  :responders:
    - :method: die_check

:die_two:
  :value: "0"
:die_two_check:
  :value: 0
  :responders:
    - :method: die_check
:die_three:
  :value: "0"
:die_three_check:
  :value: 0
  :responders:
    - :method: die_check
:die_four:
  :value: "0"
:die_four_check:
  :value: 0
  :responders:
    - :method: die_check
:die_five:
  :value: "0"
:die_five_check:
  :value: 0
  :responders:
    - :method: die_check
#DICE ROLLER
:roll:
  :value: 0
  :responders:
    - :method: roll
#RESET
:reset_game:
  :value: 0
  :responders:
    - :method: reset_game
```

client_pkgs.yaml

```
:src_dirs:
  - ./js
:packages:
  fiver:
```

- fiver

fiver.js

```
Hfiver = HControl.extend({
  tileSize: 60,
  componentName: 'fiver',
  refreshValue: function(){
    var
    _val = this.value,
    _tileHeight = this.tileSize;
    if(_val>=1 && _val<=6){
      var _tilePosY = 0-(_val*_tileHeight);
      this.setStyleOfPart('state','background-position','0px '+_tilePosY+'px');
    }
    else{
      var _tilePosY = 0-(7*_tileHeight);
      this.setStyleOfPart('state','background-position','0px '+_tilePosY+'px');
    }
  }
});
```

fiver.css

```
.default > .dice-bg,
.default > .dice-id {
  position: absolute;
  left: 0px; top: 0px; width: 60px; height: 60px;
  background-image: url(#{this.getThemeGfxFile('/dice_60px.png')});
```

fiver.html

```
<div class="dice-bg" id="bg#{_ID}"></div>
<div class="dice-id" id="state#{_ID}"></div>
```