TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduste instituut
Informaatika õppekava

Priit Pihlamägi

JavaScript 2D graafika teekide võrdlus

Bakalaureusetöö (6 EAP)

Juhendaja: Siim Karus

Tartu 2015

JavaScript 2D graafika teekide võrdlus

Lühikokkuvõte:
Javascript 2D graafika teeke on palju ja seetõttu on raske valida konkreetse rakenduse jaoks sobivat teeki. Piirdudes vaid populaarsemate teekidega on neist igal ühel siiski omad eelised ja puudused. Käesoleva bakalaureusetöö eesmärgiks on analüüsida kuut populaarset JavaScript 2D graafika teeki ja standardeid *SVG* ning *Canvas 2D Context*. Töös antakse lühikokkuvõte vaadeldud teekidest ja antakse teekide erinevatele külgedele hinnangud. Lõputöö käigus loodi ka testrakendus, mida kasutatakse võrdluses. Analüüsi tulemused aitavad uute veebirakenduste projekteerimisel valida sobivat JavaScripti 2D graafika teeki.

Võtmesõnad: Canvas 2D Context, SVG, Raphaël, Snap.svg, Svg.js, Fabric.js, Paper.js, Pixi.js, teekide võrdlus, JavaScript

Comparison of JavaScript 2D graphics libraries

Abstract:
There are a lot of JavaScript 2D graphics libraries, which makes it difficult to choose the right one for a specific application. Every library has its advantages and disadvantages, even if confined to popular libraries. The aim of this thesis is to analyse six popular JavaScript 2D graphics libraries, the SVG standard and the Canvas 2D Context standard. A short summary is created and the different aspects of the libraries are graded. In addition, a test application was created, that was used in the analysis. The results of this analysis assist in choosing a library for a specific application.

Keywords: Canvas 2D Context, SVG, Raphaël, Snap.svg, Svg.js, Fabric.js, Paper.js, Pixi.js, comparison of libraries, JavaScript

Sisukord

Sissejuhatus

Suurimate veebipõhise multimeediumi kasutusvaldkondade hulgas on reklaamitööstus ja arvutimängutööstus. Lisaks kasutatakse seda tänapäeval ka veebilehes endas interaktsioonide loomiseks. Reklaamitööstuses on kasutusel ranged piirangud faili suuruse, interaktiivsuse ja ressursinõudlikkuse osas [1]. Need piirangud määravad, kas reklaam sobib edastamiseks ja reklaamimise hinna.

Varased veebireklaamid olid lihtsalt staatilised pildid, seejärel hakati kasutama ka animeeritud *GIF* (*Graphics Interchange Format*) faile, selleks et reklaamimiseks lubatud pinda paremini ära kasutada. Kuna reklaamipind oli piiratud, tekkis vajadus rikka meedia järele. Varasemad rikka meedia reklaamid kasutasid näiteks hiirega kohale liikumise omadust, selleks et reklaami pinda suurendada. Kuna reklaamiagentuurid tahtsid edastatava reklaami üle rohkem kontrolli, hakati kasutama Macromedia Flash Player [2] pistikprogrammi. Kuna Flash Player pistikkprogrammi abil sai luua keerukaid interaktiivseid reklaame, muutus see ruttu populaarseks. Flash Player pistikprogramm parandas ka varasemalt eksisteerinud probleemi, kus erinevate veebilehitsejatega tuli kasutada erinevat koodi. [3]

Kuigi enamusel töölaua masinatel on Flash Player installeeritud või siis vähemalt võimalik installeerida, pole see niiviisi kõigil platvormidel. Näiteks Apple ei luba iPhone telefonidele seda pistikprogrammi luua. Kuid kuna reklaamitööstus tahab oma tooteid ka nendele platvormidele luua, on tekkinud vajadus standardi HTML5 (*HyperText Markup Language 5*) järele. [3]

Varased veebilehitseja sisesed mängud olid tekstipõhised ja mängu elemendid olid serveripoolselt loodud. Sarnaselt reklaamile tuli siin mängu Macromedia Flash Player pistikprogramm, mis võimaldas kliendipoolseid interaktiivseid rakendusi luua. E-kaartide ajalugu on samuti veebireklaamide ajaloole sarnane. Sarnaselt reklaamitööstusele on ka ülejäänud tööstustes tahtmine oma tooteid iPhone telefonidele, iPad tahvelarvutitele ja teistele pistikprogrammi Adobe Flash Player poolt mitte toetatud platvormidele luua. Lisaks on HTML5 standardit kasutavatel rakendustel see eelis, et üks kood töötab paljudel platvormidel.

HTML5 standardist tuli 2014 aasta oktoobris välja esimene *W3C Recommendation* staatusega versioon [4]. Tegu on esimese HTML standardiga (vaadates HTMLi ja XHTMLi (*Extensible HyperText Markup Language*) eraldi standarditena), kus saab dokumendi sees *canvas* ja *SVG* (*Scalable Vector Graphics*) elemente kasutada. *SVG* on veebistandard veebilehitsejas vektorgraafika esitamiseks. Tegu on *XML* (*Extensible Markup Language*) keelel põhineva standardiga. *Canvas 2D Context* on veebistandard veebilehitsejas rastergraafika joonistamiseks veebis [5]. *WebGL* (*Web Graphics Library*) on rakendusliidesele *OpenGL ES 2.0* (*Open Graphics Library for Embedded Systems 2.0*) sarnanev rakendusliides 3D graafika joonistamiseks [6]. HTML *canvas* elemendi sees saab joonistamiseks kasutada *Canvas 2D Context* [5] või *WebGL* [6] standardit. Antud töös *WebGL* rakendusliidest eraldi ei uurita.

Standardeid *Canvas 2D Context*, *WebGL* ja *SVG* on keerukas kasutada. Standardid ise ei paku tingimata vajalike omadusi, transformatsioonide (nt objekti pööramine) kasutamine võib tekitada probleeme ja hiire interaktsioone on keerukas luua.

Keele JavaScript 2D graafika teekide puhul on tegu funktsioonide ja muude komponentide kollektsiooniga, mis lihtsustavad multimeediumi visuaalselt esindatavate elementide loomist. Lühidalt kirjeldades on multimeediumi visuaalselt esindatavad elemendid animatsioon, interaktiivsus, pilt, tekst ja video. Multimeediumi saab kaheks jagada: rikas

meedia (interaktiivne meedia) [7] ja tavaline meedia (mitte interaktiivne meedia).

Käesoleva bakalaureusetöö eesmärgiks on analüüsida kuut populaarset JavaScript 2D graafika teeki ja kahte standardit (*SVG* ja *Canvas 2D Context*). Analüüsi tulemus peaks abistama konkreetsete rakenduste jaoks teeki valida. Pooled teegid valitakse sellised, mis kasutavad joonistamiseks *Canvas 2D Context* standardit ja pooled, mis kasutavad joonistamiseks *SVG* standardit. Teeke ja standardeid võrreldakse vastavalt väljatöötatud võrdluse metoodikale. Võrdlusel tuginetakse teekide dokumentatsioonile, teegi kodulehel olevatele õpetustele ja muule teekide autorite poolt levitatud informatsioonile. Lisaks luuakse testrakendus, mida kasutatakse võrdluses ja valitud omadustele (interaktiivsus ja transformatsioonid) luuakse näidiskoodi. Võrreldakse multimeediumi visuaalselt esindatavaid omadusi, tööd lihtsustavaid omadusi, teekide arengut ja kasutajatuge, litsentsi tingimusi ning jõudlust ja ressursinõudlikkust.

Töö jaguneb kolmeks peatükiks. Esimeses peatükis kirjeldatakse põhjalikumalt võrreldavate standardite puudujääke, töös kasutatavaid termineid, teekide valikud, võrdluse kriteeriume ja üleminekumaatrikseid. Teises peatükis kirjeldatakse ja põhjendatakse võrdluse metoodikat. Kolmandas peatükis on teekide ja standardite vaheline võrdlus koos tulemustega. Tööle on kaasa pandud viis lisa. Esimeseks lisaks on testrakendus koos kasutusjuhendiga. Teiseks ja kolmandaks lisaks on teekidega loodud näidiskood. Neljandaks lisaks on mõõtmistulemused. Viienda lisa sees on teekidele antavate hinnangute põhjendused koos vastavate viidetega.

1 Probleemi ülevaade

Selles peatükis kirjeldatakse standardite *Canvas 2D Context* ja *SVG* puudusi. Puudus-te kaudu näidatakse, miks teeke tarvis on ja moodustatakse hiljem osa võrdluse metoodi-kast. Esmalt tuuakse välja töös kasutatavaid termineid koos nende tähendustega. Seejärel kirjutatakse, millised on standardite koordinaadisüsteemid. Edasi selgitatakse täpsemalt, milline on nende standardite ja teekide üleminekumaatriks. Järgmiseks kirjutatakse, miks valiti võrdluse all olevad teegid. Viimaseks kirjeldatakse võrdluse kriteeriumit ja põhjen-datakse lühidalt, miks need olulised on.

1.1 Terminid

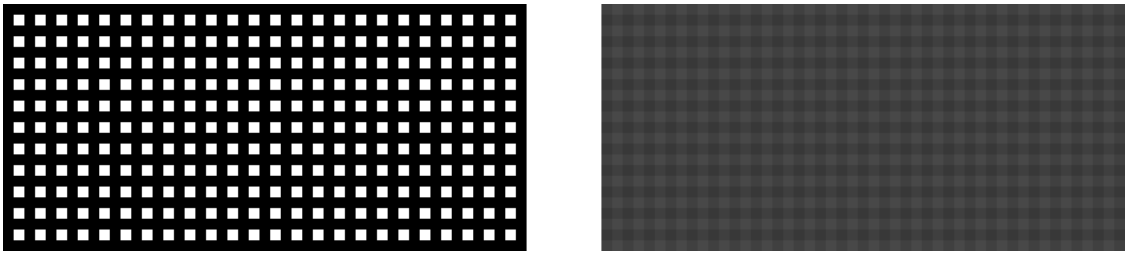
- Rõhutus (i.k. *emphasis*) – teksti osade esiletõstmine kirjastiili muudatuse kaudu.
- Hiirega lahkumine – antud töös mõeldakse selle all omadust tuvastada hiire kursori liikumist valitud piirkonnast välja.
- Hiirega kohale liikumine – antud töös mõeldakse selle all omadust tuvastada hiire kursori liikumist valitud piirkonna peale.
- Murdjoon (i.k. *polyline*) – pidevjoon, mis koosneb mitmest joonesegmendist.
- Hulk (i.k. *set*) – kogum objektidest, millel pole tehted määratud. Erinevalt rühmast ei ole sellel struktuuril kõik funktsioonid ja meetodid määratud.
- Kuhja profileerija (i.k *heap profiler*) – tarkvara, millega analüüsitakse rakenduse mälu kasutamist.
- Lõuend – antud töös mõeldakse selle all pinda, mille peale luuakse pilt. Standardi Canvas 2D Context puhul on selle all mõeldud nii *canvas* elementi, kui ka sellega seonduvat konteksti. Standardi *SVG* puhul on selle all mõeldud vaateava ja sellega seonduvat vaatekarpi.
- Objekt – geomeetriliste kujundite, piltide ja video jaoks kasutatatud üldistav nime-tus.
- Leevendamisfunktsiooni (i.k. *easing-function*) – funktsioon, mis saab sisendiks tava-liselt protsendilise väärtuse ja väljastab eelnevalt määratud omadusele uue väärtuse.
- Piksli kunst (i.k *pixel art*) – kunstistiil, milles pilti luuakse piksel haaval.
- Pöörlemisetelg (i.k. *pivot*) – punkt mille ümber objekti pööratakse.
- Puute sündmus (i.k. *touch event*) – tegevus, mida programm avastab, kui kasuta-takse puutetundlikku ekraani.
- Tee (i.k. *path*) – erinetavest kujunditest koosnev ühend, mille kaudu on moodusta-tud uus kujund.
- Sakitõrje (i.k. *anti-aliasing*) – tarkvaravahend joonte sakilisuse vähendamiseks.
- Spraidileht (i.k *sprite sheet*) – hulk pilte, mis on paigutatud ühe pildifaili sisse.
- Transformatsioon (i.k. *transformation*) – pööramise, kiivamise, skaleerimise ja asu-koha muutmise jaoks kasutatud üldistav nimetus.
- Vaateava (i.k *viewport*) – pind, mille sisse *SVG* standardis joonistatakse.
- Vaatekarp (i.k. *viewbox*) – ala, mida vaateavas näidatakse.

- Kiivamine (i.k. *skew*) – moonutus, mille puhul üks lõik jääb paigale, muud nihkuvad pikisuunas võrdeliselt kaugusega püsilõigust.
- Joonistamine – antud töös mõeldakse selle all protsessi mille käigus luuakse kuvamiseks sobiv pilt.
- Eel-visualiseerimine – antud töös mõeldakse selle all andmete joonistamist ilma kohese kuvamiseta.

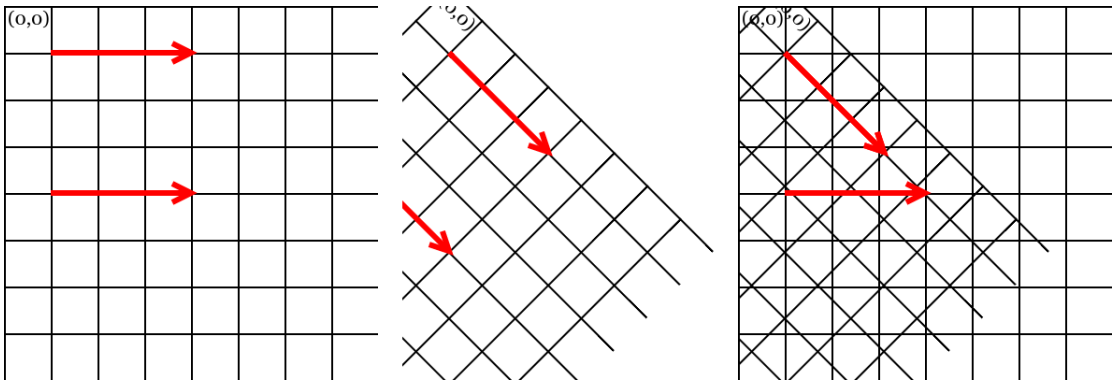
1.2 Koordinaadisüsteem, mida kasutab *Canvas 2D Context*

Standardi *Canvas 2D Context* koordinaadisüsteemi nullpunkt on vasakul ülemises nurgas [5]. Seda standardit kasutades ei saa transformatsioone otse objekti peal kasutada, seetõttu tuleb neid rakendada terve lõuendi peal [5]. Iga kord, kui on vaja objekti asukoht määrata või lõuendile transformatsiooni rakendada, kasutatakse seda koordinaadisüsteemi. Sellel koordinaadisüsteemil on paar probleemi.

Juhul kui objekte paigutatakse valesti, teeb sakitõrje pildi häguseks. Sakitõrje vältimiseks eraldi funktsioone pole. Sakitõrje vältimine on oluline näiteks piksli kunstis. Selleks et vältida olukorda, kus sakitõrje teeb pildi häguseks, tuleb jooni, mille paksus pikslites on paaritu arv, paigutada piksli keskele [8]. Jooned mille paksus pikslites on paarisarv ja muud objektid tuleb paigutada täisarvuliste koordinaatide peale. Kui sedasi ei tehta, siis võib pilt jääda hägune. See probleem on eriti märgatav ühe piksli paksuste joontega (vt Joonis 1).



Joonis 1: Probleem hägususega. Vasakul on pilt, kus jooned on liigutatud 0,5 pikslit alla ja paremale. Paremal on pilt, kus jooned on täisarvulistel koordinaatidel. Mõlemad pildid on 8 kordse suurendusega.



Joonis 2: Transformatsiooni rakendamine ainult ühele objektile. Vasak pilt on ilma pööramiseta, kesmises on rakendatud esimesed viis sammu ja paremal on kõik seitse sammu rakendatud.

Teiseks probleemiks on objektile rakendatavate transformatsioonide puudumine. Objektide pööratud asendis joonistamiseks tuleb teha järgnevat (vt Joonis 2):

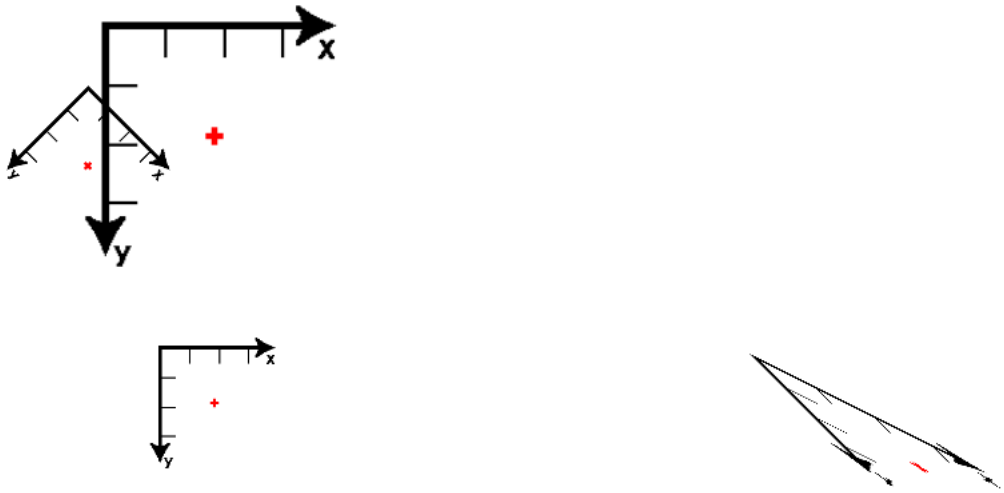
1. salvestada lõuendi koordinaadisüsteem,
2. liigutada lõuendi koordinaadisüsteemi nullpunkt pöörlemistelge,
3. pöörata lõuendi koordinaadisüsteem soovitud nurga alla,
4. liigutada koordinaadisüsteemi nullpunkt tagasi eelnevasse asendisse,
5. joonistada objekt,
6. pöörata koordinaadisüsteemi vastupidises suunas,
7. taastada esialgne koordinaadisüsteem.

Teekide kasutamine muudab objektide pööratud asendis joonistamise märkimisväärselt lihtsamaks. Nimelt vajalike sammude arv kahaneb seitsmelt kahele. Teekide puhul on nendeks sammudeks: loo objekt ja pööra seda. Kuigi üleminekumaatriksit kasutades saab asendada teise ja neljanda sammu, ei muuda see olukorda lihtsamaks. Kui joonistada objekt nullpunkti, võib viienda sammu ära jätta. Teisi transformatsioone kasutatakse sarnaselt pööramisele.

1.3 Koordinaadisüsteem, mida kasutab *SVG*

Standardi *SVG* koordinaadisüsteemi nullpunkt ülemises paremas nurgas. Objektile transformatsiooni rakendades luuakse ainult sellele objektile kehtiv uus koordinaadisüsteem. Kuigi terve lõuendi koordinaadisüsteemi ei ole võimalik muuta, saab luua vaatekarpe. Vaateava määrab *SVG* elemendi suuruse *HTML* failis. Kõik mis on vaatekarbis, joonistatakse vaateavas. Vaatekarbile on võimalik transformatsioone rakendada. [9]

Standardit *SVG* kasutades on pildi sakitõrje probleem veel keerulisem. Kuigi sakitõrje saab välja lükata [10], ei ole see alati sobiv lähenemine. Näiteks juhul kui on kasutuses korraga nii piksli kunsti kui ka kumeraid objekte, ei sobi sakitõrje täielik eemaldamine. Seetõttu oleks vaja kasutada *Canvas 2D Context*'i lahendusega sarnast lähenemist. Kahjuks ei ole standardit *SVG* kasutades seda probleemi nii lihtne lahendada. Erinevate veebilehitsejatega tuleb koordinaadisüsteemi erineva pikslite arvu võrra nihutada [11].



Joonis 3: Transformatsioonide rakendamine *SVG* standardis. All vasakul on transformatsioonideta, all paremal on kiivatud, üleval vasakpoolne on pööratud ja üleval parempoolne suurendatud objekt. Kõikide objektide x koordinaat on sama.

Kuna objektidele saab transformatsioone rakendada, siis standardis *SVG* on üks probleem vähem. Erinevatel objektidel kehtib erinev koordinaadisüsteem (vt Joonis 3). See tähendab et peale objektile transformatsiooni rakendamist peab kasutama üleminekumaatriksit või mõnda muud meetodit, selleks et objekt paigutada lõuendil soovitud kohta. Üleminekumaatriksi abil teisendatakse lõuendi koordinaadisüsteemi koordinaadid objekti koordinaadisüsteemi koordinaatideks.

1.4 Üleminekumaatriks

Standardid *Canvas 2D Context* ja *SVG* võimaldavad maatriksite kaudu transformatsioone kasutada. Nende maatriksite pöördmaatrikseid saab kasutada transformatsiooniga objektide koordinaatide lõuendi koordinaatideks muutmisel, kasutades valemit 1. Üleminekumaatriksid on sarnased maatriksid ja sarnaste maatriksite vahel on ekvivalentsiseos. Nelja kahemõõtmelise ruumi transformatsiooni saab kasutada 3×3 maatriksi abil (vt Joonis 4) [9].

$$\begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} e + a \cdot x_1 + c \cdot y_1 \\ f + b \cdot x_1 + d \cdot y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & tan(\alpha_x) & 0 \\ tan(\alpha_y) & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Joonis 4: Vasakult paremale on transformatsioonidega seonduvad maatriksid: skaleerimine, pööramine, kiivamine ja liigutamine.

1.5 Standardite *SVG* ja *Canvas 2D Context* interaktiivsus

Kuna puutetundlike ekraanidega seadmeid kasutatakse üha rohkem, on nende toetamine oluline. Puute sündmused on standardites *Canvas 2D Context* ja *SVG* olemas. Kuigi klaviatuuri sündmustel põhinevaid interaktsioone on lihtne luua, ei saa sama väita hiire ega puute sündmuste kohta. Nimelt hiire ja puute sündmuste puhul tuleb lisaks nupuvajutustele tuvastada ka see, mis on kursori all. Lisaks mõned interaktiivsuse omadused, nagu näiteks hiirega kohale liikumine, vajavad pidevat hiire kursori all olevate objektide tuvastamist. Standardis *Canvas 2D Context* saab hiire sündmusi rakendada tervele lõuendile, kuid mitte lõuendil asuvatele objektidele. Standardis *SVG* saab hiire sündmusi rakendada nii lõuendile kui ka objektidele.

Hiire interaktsioonide puhul võib olla standardi *SVG* koordinaadisüsteemil puudus. Kuna igal objektil on oma koordinaadisüsteem, siis pööratud objekti pukseerimiseks tuleb kasutada maatrikseid või muid vahendeid objekti hiire asukohta paigutamiseks.

1.6 Võrreldavad teegid

Võrdluses on ainult avatud lähtekoodiga teegid. Teekidelt nõutakse vastavust vähemalt ühe joonistamiseks kasutatava meetodiga, mis vastab *Candidate Recommendation*, *Proposed Recommendation*, *W3C Recommendation* või *Edited Recommendation* staatusega standardile [12]. Lisaks aktsepteeritakse *Living Standard* staatusega standardeid, kuna nende arengut ei lõpetata kunagi [13]. Teegid, mis interpreteerivad mingit muud keelt peale JavaScripti ning ei luba tavalist JavaScripti kasutada, jäetakse valikust välja.

Selles töös on võrdlemiseks valitud kuus populaarset JavaScripti graafilist teeki. Kolm nende hulgast joonistavad standardi *SVG* kaudu, kolm standardi *Canvas 2D Context* kaudu ja üks standardi *WebGL* kaudu. *SVG*-d kasutavad: *Raphaël* [14] (versioon 2.1.2), *Snap.svg* [15] (versioon 0.3.0) ja *svg.js* [16] (versioon 1.0.1). *Canvas 2D Context*'i kasutavad: *Fabric.js* [17] (versioon 1.4.3), *Paper.js* [18] (versioon 0.9.22) ja *pixi.js* [19] (versioon 2.2.7). *Pixi.js* [19] võimaldab lisaks *Canvas 2D Context*'ile joonistamiseks standardit *WebGL* kasutada. Lisaks võrreldakse neid teeke JavaScripti lahendustega, mis ei kasuta ühtegi teeki.

1.7 Võrdluse kriteeriumid

Kriteeriumi esimese kuue alampunkti moodustamiseks on kasutatud multimeediumi visuaalseid elemente ja interaktiivsust [7]. Seitsmes ja kaheksas alampunkt mõjutavad arendusmahtu.

Kõiki teeke võrreldakse järgneva kriteeriumiga:

1. Objektid - pildifailide toetus on vajalik realistliku informatsiooni esitamiseks [20]. Sarnaselt pildifailide toetusele, on videofailide toetus vajalik realistliku animatsiooni esitamiseks. Geomeetrilised kujundid võimaldavad luua joonistusi, jooniseid ja skeeme.

2. Objektide transformatsioonid ja töötlemist lihtsustavad omadused - objektide rühmadesse ja kihtidesse jaotamine lihtsustab nende töötlemist ning ühtselt lahendatud transformatsioonid tekitavad vähem segadust.

3. Teksti kujundamine - multimeediumi puhul võib olla vajalik esitada infot teksti kujul. Kujundatav tekst on pildifailist parem, kuna see on muudetav ja kulutab vähem ribalaiust.

4. Pilditöötlus - vajadusel saab pildile või joonisele anda soovitud omadused.

5. Animatsioon - see on vajalik ajas toimuvate muudatuste näitamiseks, objektide graafiliste esituste rikastamiseks ja tähelepanu püüdmiseks [20].

6. Interaktiivsus - interaktiivsuse omadused võimaldavad rakendusel kasutaja tegevusele reageerida.

7. Kasutajatugi - dokumentatsioon, õpetused ja koodinäited abistavad teegi õppimist.

8. Kogukonna aktiivsus - kogukonna aktiivsus on vajalik vigade leidmiseks, nende parandamiseks ja teegi omaduste kaasajastamiseks.

9. Litsents - litsents määrab ära, mis olukorras saab teeki kasutada ja mida sellega teha tohib.

10. Jõudlus ja ressursinõudlikkus - jõudlus ja ressursinõudlikkus määravad, kas antud teeki saab rakenduse loomiseks kasutada.

2 Võrdluse metoodika

Järgnevates alampunktides kirjeldatakse, kuidas võrdluse kriteeriumeid hinnatakse. Arvulisi väärtusi nimetatakse edaspidi punktideks. Igas alampunktis antav punktiväärtus on vahemikus null kuni kümme. Alampunktide punktiväärtuste summa moodustab teekide koguhinnangu vahemikus null kuni sada. Järgnevate alampunktide abil määratakse järgmises peatükis teekidele hinnangud.

Kuna antud töös vaadatakse teeke ja standardeid kõikjal kus võimalik ühtse hindamiskriteeriumiga, antakse teekidele punkte ainult neis endis leiduvate omaduste eest. Automaatselt standardis leiduvate omaduste eest punkte ei määrata. Juhul kui standardis leidub omadus, kuid seda kasutavas teegis ei leidu, siis selle omaduse eest teegile punkte ei määrata.

2.1 Objektid

Hinnatavate geomeetriliste kujundite hulgas on tee, kuna selle abil saab luua üldistatud kujul kõik muud kahemõõtmelised kujundid [21]. Kuna tee abil kujundite moodustamine on keerulisem, siis hinnatakse ka ristküliku ja ellipsi olemasolu. Punkti olemasolu hinnatakse, kuna selle abil saab objekte paigutada. Iga hinnatava objekti olemasolu annab ühe punkti. Rasterpiltide ja video tugi annavad mõlemad kaks punkti. *SVG* pildiformaadi toe eest antakse üks punkt, kui *SVG* pilti ei saa muuta ja kaks, kui seda saab teegi abil töödelda.

2.2 Objektide töötlemine

Olenevalt loodavast rakendusest võib igal objektil oma koordinaadisüsteemi olemasolu tekitada eelise või puuduse. Seetõttu selle eest otseselt punkte ei määrata. Samas on ilmselge, et teek või standard peab transformatsioonidega käituma ühtlaselt. Seetõttu antakse transformatsioonide eest punkte ainult siis, kui kõik transformatsioonid käituvad koordinaadisüsteemiga ühte moodi.

Objekti tranformatsiooni muutvatele funktsioonidele seatakse järgnevad kaks tingimust: esiteks transformatsiooni kasutamine ei muuda objekti või lõuendi koordinaadisüsteemi ja teiseks, et transformatsiooni rakendamiseks ei ole vaja kasutada üleminekumaatriksit. Kui mõlemad nõuded kehtivad kõigi olemasolevate transformatsioonide puhul, antakse iga transformatsiooni eest kaks punkti. Kui esimene nõue ei kehti ühegi olemasoleva transformatsiooni puhul, kuid teine kehtib, määratakse iga transformatsiooni eest kaks punkti. Ülejäänud juhtudel määratakse null punkti. Punkte antakse ainult selliste funktsioonide eest, mis muudavad ainult ühte omadust. Hinnatakse järgnevaid transformatsioone:

- nurga muutmine,
- kiivamine,
- suuruse muutmine.

Objektide rühmadesse jagamise võimalus annab kaks punkti. Objektide kloonimise ja kihtidesse jaotamise tugi annavad kumbki ühe punkti. Iga teegiga luuakse transformatsioonide testimiseks näidiskood (vt Lisa 2).

2.3 Tekstitöötlus

Tekstitööltuse puhul annab iga järgnev omadus kaks punkti:

- mitmerealise teksti tugi;
- võimalus anda tekstile taustavärv;
- võimalus teksti rööpjoondada;
- võimalus kasutada allajoonimise, kursiivi ja paksu kirja rõhutusi;
- võimalus kasutada rõhutusi rea sees üksikutel tähemärkidel.

Kuna šrifti muutmine; kirja suuruse valimine; vasakule, paremale ja keskele joondamine ning muud olulisemad omadused leiduvad kõigis teekides, siis neid hindamisel arvesse ei võeta.

Mitmerealine tekst on vajalik, kuna tekst ei mahu alati ühele reale. Taustavärv aitab kirja nähtavamaks teha ning seda on võimalik ka rõhutusena kasutada. Kirja rõhutused aitavad tekstist olulist infot välja tuua. Rööpjoonduse abil saab tekstile ametlikuma, meeldivama või tuttavama välimuse anda.

2.4 Pilditöötlus

Iga järgneva omaduse olemasolu annab ühe punkti.

- Objektile on võimalik anda vari. Varju kuju peab muutuma koos objektiga.
- Objektile on võimalik anda piirjoon. Teekide puhul, mis kasutavad joonistamiseks standardit *SVG*, nõutakse lisaks sakitõrjeprobleemi lahendamise lihtsustamist.
- Objekti on võimalik tekstuuriga täita.
- Objekti on võimalik gradientiga täita.

Pildifiltrite arvu k ja sujutamise viiside arvu l eest antav punktisumma leitakse valemist.

$$p = \min(\left\lceil \frac{k+l}{4} \right\rceil, 6) \tag{2}$$

Filtreid millega määratakse sujutamise viise arvesse ei võeta. Sujutamise viisidest võetakse arvesse kõik peale vaikimisi viisi.

Filtrite, varju, gradienti, tekstuuri ja piirjoone abil saab objektidele soovitud omadusi anda ja seeläbi ribalaiuse pealt kokku hoida.

2.5 Animatsioon

Iga järgneva animatsiooni omaduse olemasolu annab ühe punkti.

- Funktsioon või meetod objekti liigutamiseks.
- Funktsioon või sündmus, mida käivitatakse animatsiooni lõpus.
- Funktsioon uue kaadri loomiseks.

- Funktsioon rajal asuva punkti koordinaatide leidmiseks.
- Spraidilehtede kasutamise võimalus.

Leevendamisfunktsioonide arvu k eest antav punktisumma leitakse valemist.

$$p = \min(\left\lceil \frac{k}{5} \right\rceil, 5) \tag{3}$$

Lineaarset leevendamisfunktsiooni arvesse ei võeta.

Funktsioon objekti liigutamiseks on kasulik, kuna selle abil saab lihtsaid animatsioone teha. Animatsiooni lõpus käivitatava funktsiooni abil saab käivitada sündumste ahelaid. Kuna mõned veebilehitsejad ei toeta requestAnimationFrame funktsiooni, on hea kui teek pakub ise taanderežiimi. Funktsioon teel asuva punkti koordinaatide leidmiseks on kasulik, kuna selle abil saab keerukaid animatsioone luua, kus animatsiooni loomiseks saab kasutada vektorgraafika tarkvara abil loodud teid. Spraidilehed on kasulikud, kuna nende abil saab ribalaiust kokku hoida. Lisaks saab spraidilehti kasutada olukorras, kus video pole võimalik kasutada. Leevendamisfunktsioonide abil saab luua lihtsaid üldkasutatavaid animatsioone [22].

2.6 Interaktiivsus

Iga järgneva interaktiivsuse omaduse olemasolu annab ühe punkti, kui seda saab rakendada objektile. Kui seda rakendatakse tervele lõuendile, antakse null punkti.

1. Pukseerimine
2. Klikk
3. Topeltklikk
4. Hiirega kohale liikumine
5. Hiirega lahkumine
6. Hiire nupu vajutamine
7. Hiire nupu lahti laskmine

Hiire kursori liikumisel asukoha tuvastamine, hiirega kerimise tuvastamine ja puutetundliku ekraani tugi annavad iga üks ühe punkti. Kui teek ei võimalda objektidele või lõuendile puute sündmusi määrata, väidetakse et puutetundliku ekraani tugi puudub. Iga teegi- ja standardiga luuakse hiire interaktsioonide testimiseks näidiskood (vt Lisa 3).

Hinnatavate interaktiivsuse omaduste vajalikkus on juba eelnevates peatükkides osaliselt põhjendatud. Hiirega kerimist hinnatakse seetõttu, et erinevad veebilehitsejad kasutavad selleks erinevaid funktsioone [23].

2.7 Kasutajatugi

Kasutajatoe hindamise eesmärgiks on teada saada, kui lihtne on teegi kasutamist õppida. Sellest tulenevalt hinnatakse kasutajatoe all järgnevaid punkte.

- Dokumentatsioon on olemas.
- Dokumentatsioon on kergesti leitav, s.t. link on teegi veebilehe esilehel.
- Dokumentatsioonil on otsingusüsteem või kogu dokumentatsioon on ühel lehel.
- Dokumentatsioon on grupeeritud teemade kaupa ja täielik.
- Dokumentatsioonis on koodinäited või leidub põhjalik õpetus.

Iga punkti täidetus annab kaks punkti. Kui hindamismetoodikas hinnatavate omaduste kontrollimise ja testrakenduse loomise käigus ei leidu omadusi, mis dokumentatsiooni puudu on, väidetakse et dokumentatsioon on täielik. Koodinäiteid arvestatakse ainult siis, kui tegu on JavaScript keeles näidetega.

Dokumentatsiooni, koodinäidete ja õpetuse olemasolu lihtsustab teegi kasutama õppimist. Dokumentatsiooni leitavus, otsingusüsteem ja teemade kaupa grupeerimine lihtsustavad soovitud omaduste otsimist. Kui dokumentatsioon on ühel lehel, saab veebilehitseja sisest otsingut kasutada.

2.8 Kogukonna aktiivsus

Ametliku foorumi ja sotsiaalvõrgustiku konto olemasolu annavad mõlemad 2,5 punkti. Viimase 52 nädala jooksul koodihoidlasse tehtud muudatuste arvu k eest antakse punkte järgneva valemi järgi.

$$p = \min(\left\lfloor \frac{k}{52} \right\rfloor, 5) \tag{4}$$

Ilmselt ei saa veebistandardeid selle valemiga mõõta. Standardite implementatsioonides leiduvad vead parandatakse veebilehitsejate poolt. Seega vead mis leiduvad standardi implementatsioonis, on tõenäliselt olemas ka teegis, mis seda standardit joonistamiseks kasutab. Seetõttu antakse standarditele 2,5 punkti.

Foorumi ja sotsiaalvõrgustiku olemasolu aitavad teegi veaparanduste ja uuendustega kursis olla. Kui teek on aktiivses arenduses, ei pea teeki kasutava rakenduse looja veaparandusi üksinda kirjutama. Teegi aktiivse arenduse abil väheneb rakenduse arendaja töö hulk.

2.9 Litsents

Teegile antakse 2,5 punkti, kui seda saab kasutada ärilistel eesmärkidel ilma, et rakenduse lähtekoodi oleks vaja avalikustada. Kui teek ja rakenduse kood on lubatud ühte faili kokku panna, antakse 2,5 punkti. Litsentsist tulenevate kohustuste arvu k ja muude piirangute arvu l järgi antakse punkte järgneva valemiga.

$$p = \max(5 - (k + l), 0) \tag{5}$$

Litsentsi abil määratakse kuidas teeki tohib kasutada. Kõik kohustused ja piirangud on olukorrad, mille tõttu võib seadustega hätta jääda.

2.10 Jõudlus ja ressursinõudlikkus

Objektide arv mõjutab standardit *SVG* rohkem, kui *Canvas 2D Context*'i [24]. Eelneva väite tõesuses veendumiseks luuakse iga teegiga testrakendus. Omadused, nagu eel-visualiseerimine ja sümbolid, võimaldavad koodi optimeerida. Teegi väiksem suurus võimaldab kasutada vähem ribalaiust. Reklaami puhul mõõdetakse suurust peale kokkupakkimist [1] ja reklaami müüjad panevad suurusele piirangud [25]. Hindamiskriteeriumis on teegi kokkupakkimiseks kasutuses Closure Compiler [26] ja gzip. Nende tingimuste põhjal jagatakse punkte järgnevalt.

- Üks punkt antakse, kui teegis on joonistamiseks kasutusel *Canvas 2D Context* või *WebGL*. Lisaks nõutakse punkti andmisel, et testrakendus oleks kinnitanud iga raamistiku korral *SVG* ja *canvas*'e jõudluse kohta käivat väidet.
- Eel-visualiseerimise annab kolm punkti.
- Sümbolite tugi annab ühe punkti.
- Teegi suuruse kilobaitides k eest määratavad punktid leitakse valemist

$$p = \min(\left\lfloor \frac{50 - k}{5} \right\rfloor, 5). \tag{6}$$

Hindamisse on võetud faili suuruseks 50 kilobaiti, kuna yahoo kasutab seda piirangut mobiilsete seadmete jaoks [25].

2.11 Testrakendus

Testrakenduseks (vt Lisa 1) on valitud lumesaju animatsioon, kus lumehelveste arv on muutuja. Testrakenduseks valiti lumesaju animatsioon, kuna seda on talvehooajal Internetis, reklaamides, e-kaartidel ja veebilehtede taustaks tihti kasutatud. Sarnast animatsiooni on kasutatud ka Internetis mängudes. Rakenduse abil mõõdetakse, kui palju mõjutab teegi kasutamine pildi joonistamise kiirust erinevate objektide (lumehelveste) arvu korral, kus igal objektil (lumehelbel) on väike hulk üksteisest erinevaid parameetreid nagu suurus, kaldenurk, läbipaistvus. Lisaks on see testrakendus kasulik ka näidiskoodina. Erinevate teekide testrakenduse lähtekood on jäetud võimalikult sarnaseks. Kui võimalik, on muudetud vaid seda, kuidas faile laetakse ja objekte luuakse. Vajaduse korral on lumehelveste asukohta määravat osa muudetud selleks, et lumehelbed ekraani peale jääksid. Kui teegi abil on võimalik luua testrakendus mitmel erineval viisil, võetakse töö sisse neist kiireim.

Kaadrisageduse mõõtmiseks kasutatakse veebilehitsejaid Firefox Developers Edition [1] (versioon 38.02a) koos selle sees oleva jõudlusmonitoriga [27] ja Google Chrome [2] (versioon 42.0.2311.90) koos JavaScript'is loodud kaadrisageduse analüüsimise vahendiga. Mõõdetakse nende kahe veebilehitsejaga, kuna hetkel on need kaks kõige populaarsemat veebilehitsejat [28]. Testi tulemused esitatakse ühikuga "kaadrit sekundis".

Mälu analüüsimiseks kasutatakse veebilehitsejas Firefox leiduvat "about:memory" nimelist tööriista [29]. Veebilehitseja Google Chrome puhul kasutatakse "Record Heap Allocations" tööriista. Testi tulemused esitatakse megabaitides.

[1] `https://www.mozilla.org/en-US/firefox/developer/`
[2] `http://www.google.com/chrome/`

Protsessori koormuse analüüsimiseks kasutatakse Google Chrome puhul sisseehitatud "Task Manager" tööriista [30]. Veebilehitseja Firefox puhul mõõdetakse protsessori koormust välise programmiga htop [31]. Testi tulemused esitatakse protsentides. Üks protsent tähistab ühe arvuti tuuma puhul mõõtmisvahemikus olevat aega, mis läks testi jooksmisele. Väiksemad väärtused on paremad, kuna sama tulemuse jaoks on kasutusel vähem ressursse.

Iga test algab 500 lumehelbega ja lumehelveste arvu tõstetakse 500 võrra, kuni neid on ekraanil korraga 2500. Kõiki teste mõõdetakse iga teegi ja mõõdetava lumehelveste arvu korral kümme korda. Testide alusel luuakse iga raamistiku kohta tabel, kus on peal tulemuste keskmine, mood ja mõõtmistulemuste vahemik.

Kaadrisageduse testil lastakse iga kord töötada 1 minut. Mälu ja protsessori testide puhul võetakse pärast 10 sekundilist programmi tööd olev hetkeolukord.

Aeglaseima *canvas* elementi kasutava testrakenduse ja kiireima *SVG* standardit kasutava testrakenduse keskmiste kaadrisageduste tulemuste kohta luuakse graafikud. Graafikute püsttelgedel on kaadrisagedus ja rõhttelgedel on lumehelveste arv.

3 Kirjeldus ja võrdlus

Järgnevates alampunktides hinnatakse vastavalt võrdluse metoodikale teeke. Esmalt tuuakse välja testrakenduse kaadrisagedused ja mälulekete olemasolud. Seejärel kirjeldatakse ja hinnatakse teeke. Lõpuks tuuakse välja võrdluse tulemuste (vt Lisa 5) lühikokkuvõte.

3.1 Jõudlusvõrdlus

Testrakenduse tulemusi mõõdeti Kubuntu [32] operatsioonisüsteemis, mille peale oli ainult tarkvarauuendused ja draiverid installitud. Ühtegi ebavajalikku rakendust samal ajal käima ei jäetud. Masinas, milles tulemusi mõõdeti, oli Intel Core i7 3630QM protsessor [33], 8 GB muutmälu ja Intel HD Graphics 4000 graafikakaart.

Teekide *Raphaël* ja *snap.svg* testrakendust mõõdeti sättega, mis kasutab maatrikseid. Standardi *SVG* testrakendust mõõdeti sättega, kus transformatsiooniga objektide koordinaate arvutatakse valemiga 1. Andmetest (vt Tabel 1) ilmneb, et *SVG* standardit kasutavatest testrakendustest oli Firefox veebilehitsejas kiireim *SVG* standardiga loodud testrakendus, mis ei kasutanud ühtegi teeki. Ka veebilehitsejaga Google Chrome oli andmete (vt Tabel 3) järgi *SVG* standardit kasutavatest testrakendustest kiireim *SVG* standard ise.

Tabel 1: *SVG* teekide kaadrisagedus Firefox'is. Tabelis on ainult mõõtmistulemuste (vt Lisa 4) üldstatistikud. Mõõtmistulemused on esitatud ühikuga kaadrit sekundis.

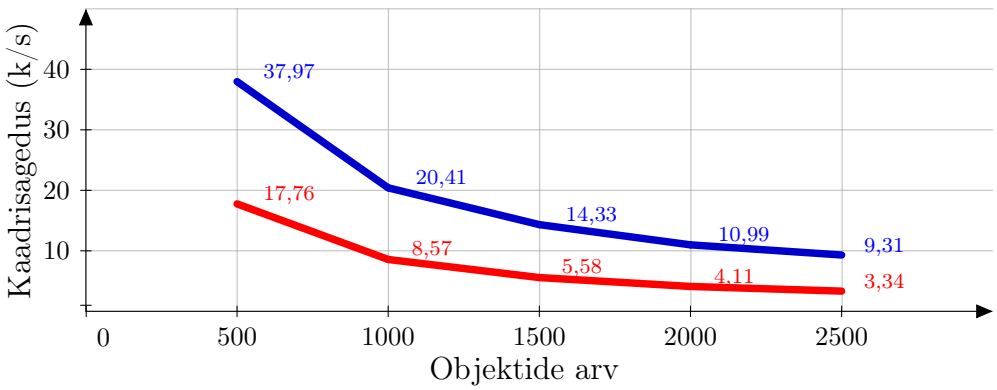
Teek	Objekte	Miinimum	Maksimum	Keskmine	Mediaan
Raphaël	500	9,00	10,13	9,85	9,91
	1000	4,55	4,97	4,79	4,79
	1500	2,99	3,08	3,04	3,04
	2000	2,14	2,26	2,21	2,22
	2500	1,70	1,83	1,76	1,76
Snap.svg	500	11,12	11,74	11,47	11,45
	1000	5,56	5,69	5,64	5,66
	1500	3,71	3,86	3,79	3,80
	2000	2,77	2,83	2,81	2,81
	2500	2,09	2,30	2,23	2,24
Svg.js	500	6,56	6,73	6,63	6,62
	1000	2,46	2,59	2,52	2,51
	1500	1,27	1,36	1,32	1,33
	2000	0,82	0,89	0,86	0,87
	2500	0,55	0,62	0,59	0,60
SVG	500	17,47	18,10	17,76	17,71
	1000	8,50	8,72	8,57	8,54
	1500	5,48	5,73	5,58	5,57
	2000	3,98	4,20	4,11	4,10
	2500	3,27	3,43	3,34	3,33

Teegi *Paper.js* ja standardi *Canvas 2D Context* testrakendusi mõõdeti sättega, mis kasutab eel-visualiseerimist. Teegi *Pixi.js* testrakendust mõõdeti mõlema sättega. Andmetest (vt Tabel 2) ilmneb, et *Canvas 2D Context* standardit kasutavatest testrakendustest

oli Firefox veebilehitsejas aeglaseim *Paper.js* teegiga loodud testrakendus. Veebilehitsejas Google Chrome oli andmete (vt Tabel 4) järgi aeglaseimaks *canvas* elementi kasutavaks teegiks samuti *Paper.js*.

Tabel 2: *Canvas* teekide kaadrisagedus Firefox'is. Tabelis on ainult mõõtmistulemuste (vt Lisa 4) üldstatistikud. Mõõtmistulemused on esitatud ühikuga kaadrit sekundis.

Teek	Objekte	Miinimum	Maksimum	Keskmine	Mediaan
Fabric.js	500	47,33	50,23	48,53	48,57
	1000	26,51	27,98	27,10	27,08
	1500	17,45	19,37	18,13	18,06
	2000	13,29	14,26	13,75	13,75
	2500	10,68	11,71	11,12	11,06
Paper.js	500	36,16	39,48	37,97	38,10
	1000	19,37	21,25	20,41	20,52
	1500	14,15	14,58	14,33	14,30
	2000	10,72	11,53	10,99	10,86
	2500	8,72	9,61	9,31	9,36
Pixi.js ja *WebGL*	500	58,89	59,15	59,03	59,02
	1000	58,62	59,16	58,94	58,93
	1500	58,68	59,18	58,95	58,98
	2000	58,49	59,04	58,78	58,84
	2500	58,84	59,08	58,96	58,95
Pixi.js ja *Canvas 2D Context*	500	42,52	46,30	45,09	45,50
	1000	22,35	25,99	24,60	24,68
	1500	16,09	17,77	17,15	17,26
	2000	12,55	13,78	12,94	12,87
	2500	10,14	11,57	10,41	10,25
Canvas 2d Context	500	56,28	56,78	56,48	56,43
	1000	52,78	56,92	54,72	54,55
	1500	39,78	43,88	41,48	41,38
	2000	32,39	33,26	33,01	33,06
	2500	27,28	28,54	27,73	27,64



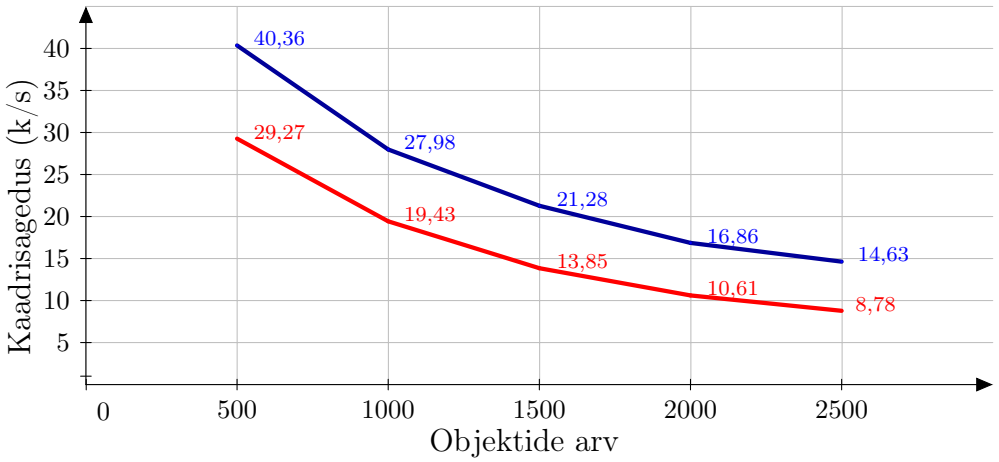
Joonis 5: *Paper.js* ja *SVG* keskmine kaadrisagedus Firefox'is. Sinine joon tähistab teegi *Paper.js* ja punane joon tähistab standardi *SVG* kaadrisagedust.

Tabel 3: *SVG* teekide kaadrisagedus Google Chrome'is. Tabelis on ainult mõõtmistulemuste (vt Lisa 4) üldstatistikud. Mõõtmistulemused on esitatud ühikuga kaadrit sekundis.

Teek	Objekte	Miinimum	Maksimum	Keskmine	Mediaan
Raphaël	500	16,10	17,10	16,64	16,65
	1000	8,20	9,10	8,64	8,55
	1500	5,30	5,70	5,44	5,40
	2000	4,00	4,30	4,07	4,00
	2500	3,10	3,40	3,25	3,25
Snap.svg	500	16,20	18,00	17,04	17,00
	1000	9,70	10,10	9,91	9,90
	1500	6,60	6,90	6,84	6,90
	2000	4,90	5,40	5,21	5,20
	2500	4,00	4,40	4,25	4,30
Svg.js	500	8,10	8,40	8,27	8,30
	1000	3,30	3,40	3,39	3,40
	1500	1,70	1,80	1,79	1,80
	2000	1,10	1,10	1,10	1,10
	2500	0,70	0,80	0,72	0,70
SVG	500	28,00	29,80	29,27	29,45
	1000	17,90	20,40	19,43	19,70
	1500	13,50	14,20	13,85	13,90
	2000	10,10	10,80	10,61	10,65
	2500	8,30	9,10	8,78	8,80

Tabel 4: *Canvas* teekide kaadrisagedus Google Chrome'is. Tabelis on ainult mõõtmistulemuste (vt Lisa 4) üldstatistikud. Mõõtmistulemused on esitatud ühikuga kaadrit sekundis.

Teek	Objekte	Miinimum	Maksimum	Keskmine	Mediaan
Fabric.js	500	58,00	60,10	59,20	59,35
	1000	40,80	47,20	43,36	42,85
	1500	30,70	32,90	31,75	31,70
	2000	25,80	27,90	27,12	27,30
	2500	21,50	24,40	23,09	23,40
Paper.js	500	38,90	41,50	40,36	40,50
	1000	26,50	30,10	27,98	27,60
	1500	20,40	22,60	21,28	20,90
	2000	16,30	18,20	16,86	16,75
	2500	13,90	15,60	14,63	14,25
Pixi.js ja *WebGL*	500	59,80	60,10	59,96	59,95
	1000	59,50	60,00	59,89	59,90
	1500	59,30	59,80	59,55	59,55
	2000	59,20	59,70	59,48	59,50
	2500	59,00	59,50	59,31	59,40
Pixi.js ja *Canvas 2D Context*	500	58,80	60,00	59,78	59,85
	1000	45,70	48,40	47,49	47,55
	1500	36,40	38,90	37,69	38,15
	2000	28,40	30,90	29,62	29,60
	2500	24,80	27,70	26,70	27,50
Canvas 2D Context	500	58,70	60,00	59,35	59,35
	1000	57,80	59,60	58,82	58,90
	1500	47,70	50,60	49,00	48,90
	2000	41,80	44,50	42,91	42,90
	2500	39,30	41,30	40,63	41,05



Joonis 6: *Paper.js* ja *SVG* keskmine kaadrisagedus Google Chrome'is. Sinine joon tähistab teegi *Paper.js* ja punane joon tähistab standardi *SVG* kaadrisagedust.

Aeglaseim *canvas* elementi kasutav testrakendus oli Firefox (vt Joonis 5) ja Google Chrome (vt Joonis 6) veebilehitsejates kiirem kui kiireim *SVG* standardit kasutav test-

rakendus. Seega väide kehtib ka Firefox ja Chrome veebilehitsejate puhul ja joonistamise standardit võetakse ka edaspidises hinnangus kasutusse.

Veebilehitsejates Firefox ja Google Chrome kasutasid protsessorit kõige säästlikumalt *PixiJS*, *Fabric.js* ja *Canvas 2D Context* teekide ning standarditega loodud testrakendused.

Veebilehitsejas Firefox olid kõige mälusäästlikumad standardeid *Canvas 2D Context* ja *SVG* ning teeki *Pixi.js* kasutavad testrakendused. Kõige mälusäästlikum *SVG* standardit kasutav teek oli *Svg.js*. *Svg.js* kasutas 2500 lumehelbe korral mälu kaks ja pool korda rohkem, kui kõige kehvem *canvas* elementi kasutav testrakendus. Veebilehitsejas Google Chrome olid kõige mälusäästlikumad testrakendused *Pixi.js*, *Canvas 2D Context* ja *SVG* abil loodud. Vaatamata sellele, et *Canvas 2D Context* abil loodud testrakenduses kasutati eel-visualiseerimist, oli see mälusäästlikuse vaatepunktist parem kui paljud teegid.

3.2 *Raphaël*

Raphaël loodi Dmitry Baranovskiy poolt eesmärgiga luua adapter, mis ühtlustaks vektorgraafika joonistamist erinevate veebilehitsejate vahel. Teek kasutab pildi joonistamiseks *SVG* või *VML* (*Vector Markup Language*) standardit ja lihtsustab veebis vektorgraafikaga töötamist. *Raphaël* toetab veebilehitsejaid Firefox 3.0, Safari 3.0, Chrome 5.0, Opera 9.5, Internet Explorer 6.0 ja nende uuemaid versioone. [14]

Teegil leiduvad objektide pööramise ja suuruse muutmise funktsioonid. Kiivamiseks tuleb kasutada maatriksit. Rühmadesse jagamisele lähim omadus on hulk [34].

Teegi *Raphaël* suurimaks eeliseks teiste teekide ees on parim veebilehitsejate tugi. Teegi *Raphaël* abil saab kasutada vektorgraafikat ka mõnes veebilehitsejas, mis *SVG* standardit ei toeta. Standardiga *SVG* loodud testrakendus oli teegiga *Raphaël* loodud testrakendusest Firefox veebilehitsejas kuni 1,9 korda kiirem ja veebilehitsejas Google Chrome kuni 2,7 korda kiirem. Teegi suurimateks puudusteks on vähene arendustegevus ja aegunud funktsionaalsus.

3.3 *Snap.svg*

Snap.svg kirjutati Dmitry Baranovkiy poolt eesmärgiga luua standardi *SVG* moodsaid omadusi kasutav ja tööd lihtsustav teek. Tegu on sama inimesega, kes lõi teegi *Raphaël*. Uus teek loodi, kuna standardi *SVG* moodsate omaduste kasutuselevõtt takistab vanemate veebilehitsejate toetamist. Teek kasutab pildi joonistamiseks ainult standardit *SVG*. *Snap.svg* toetab järgnevaid veebilehitsejaid: Internet Explorer 9 ja selle uuemad versioonid, Safari, Google Chrome, Firefox ning Opera. [35]

Selles teegis ei leidu objektide transformatsioonide jaoks eraldi funktsioone. Kõik kolm hinnatavat transformatsiooni on saavutatavad, kuid nende kasutamiseks on vaja maatrikseid. Lisaks on neid transformatsioone võimalik saavutada muutes otse *SVG* objekti atribuute (s.t. tegu pole eraldi funktsioonidega). Kõik transformatsioonid muudavad objektile kehtivat koordinaadisüsteemi. Teegi abil saab objekte rühmadesse jaotada ja kloonida, kuid mitte kihtidesse jaotada.

Teegi *Snap.svg* süntaks on sarnane teegi *Raphaël* süntaksile. *Snap.svg* eelis teegi *Raphaël* ees on *SVG* standardi uuemate omaduste toetamine, parem kaadrisagedus ja väiksem mälukasutus. Standardiga *SVG* loodud testrakendus oli teegiga *Snap.svg* loodud testrakendusest Firefox veebilehitsejas kuni 1,5 korda kiirem ja veebilehitsejas Google Chrome kuni kaks korda kiirem. Teegi *Snap.svg* veebilehitsejate tugi on kehvem kui teegil *Raphaël*.

3.4 *Svg.js*

Svg.js on kergekaaluline teek *SVG* piltide töötlemiseks ja animeerimiseks [16]. Tegu on väiksemahulise teegiga, millel on lihtne süntaks. Omadustele, mida kasutatakse tihti, on loodud funktsioonid. Teek toetab veebilehitsejaid: Firefox (versioon 3), Google Chrome (versioon 4), Safari (versioon 3.2), Opera (versioon 9), Internet Explorer 9, ja nende uuemaid versioone [36]. *Svg.js* autoriks on Wout Fierens [16].

Tegu on ühega kahest võrreldavast teegist, milles leidub iga hinnatava transformatsiooni jaoks eraldi funktsioon. Kuigi objekti suuruse muutmine ei muuda objekti koordinaadisüsteemi, teevad seda objekti pööramine ja kiivamine. Ühegi transformatsiooni jaoks pole vaja kasutada maatriksit. Teegi *svg.js* abil saab objekte rühmadesse jaotada ja kloonida, kuid mitte kihtidesse panna.

Svg.js on *SVG* standardit kasutavatest teekidest tekstitöötluse ja mälukasutuse poolest parim. *Svg.js* on kõige väiksema suurusega teek. Samas on *Svg.js* kõige väiksema kaadrisagedusega teek. Standardiga *SVG* loodud testrakendus oli teegiga *Svg.js* loodud testrakendusest Firefox veebilehitsejas kuni 5,7 korda kiirem ja veebilehitsejas Google Chrome kuni 12 korda kiirem.

3.5 *Fabric.js*

Fabric.js arendamist alustati 2010. aastal [37] eesmärgiga luua JavaScript'i põhine redaktor vektorgraafika kujundite- ja piltidega töötamiseks [38]. Teek toetab veebilehitsejaid: Firefox (versioon 2), Safari (versioon 3), Opera (versioon 9.64), Google Chrome (kõiki versioone), Internet Explorer 9 ja nende uuemaid versioone [38]. Lisaks pakutakse osalist tuge veebilehitseja Internet Explorer versioonidele 6, 7 ja 8 kasutades *ExplorerCanvas* teeki [38]. Teegi arenduseesmärkideks on kiirus ja modulaarsus [38].

Teegis leiduvad funktsioonid objektide pööramiseks ja suuruse muutmiseks [39]. Nende kahe transformatsiooni kasutamine ei muuda koordinaadisüsteemi. Kiivamiseks on vaja kasutada üleminekumaatriksit ja see muudab objekti koordinaadisüsteemi. Lisaks saab objekte kloonida, rühmadesse jaotada ja kihtidesse jaotada [39].

Fabric.js on kõikide tulemuste peale kokku suurima punktisummaga teek (vt Tabel 5). Teegil on parimad pilditöötluse ja animatsiooni omadused ning aktiivseim kogukond. Samas ei saa väita, et *Fabric.js* sobiks kõikjal kasutamiseks. Eel-visualiseerimise tugi puudub ja teek on ilmselt reklaamitööstuses kasutamiseks liiga suur. Standardiga *Canvas 2D Context* loodud testrakendus oli teegiga *Fabric.js* loodud testrakendusest Firefox veebilehitsejas kuni 2,5 korda kiirem ja veebilehitsejas Google Chrome kuni 1,8 korda kiirem.

3.6 *Paper.js*

Paper.js kasutab joonistamiseks *Canvas 2D Context* standardit. Teegi autoriteks on Jürg Lehni ja Jonathan Puckey [40]. *Paper.js* võimaldab peale JavaScript keele kasutada ka PaperScript keelt [40].

Kõigi transformatsioonide jaoks leidub funktsioon, mida saab objektil rakendada ja nende kasutamine ei muuda objekti koordinaadisüsteemi. Objekte saab rühmadesse jaotada, kloonida ja kihtidesse jaotada [41]. Tegu on ainsa teegiga, kus leiduvad kõik selles punktis hinnatavad omadused.

Paper.js on kõikide tulemuste peale kokku teekidest ja standarditest punktisummalt kolmandal kohal (vt Tabel 5). *Paper.js* puhul on tegu ühega kahest teegist, millel leidub eel-visualiseerimise tugi. Reklaamitööstuses kasutamiseks *Paper.js* ei sobiks, kuna

on suurem kui 50 KB (vt Lisa 5). Standardiga *Canvas 2D Context* loodud testrakendus oli teegiga *Paper.js* loodud testrakendusest Firefox veebilehitsejas kuni kolm korda kiirem ja veebilehitsejas Google Chrome kuni 2,8 korda kiirem. *Paper.js* oli *canvas* elementi kasutavate teekide hulgast aeglaseim.

3.7 *Pixi.js*

Pixi.js võimaldab kasutada riistvarakiirendust ilma standardi *WebGL* kasutusoskuseta [42]. Teek kasutab joonistamiseks *WebGL* või *Canvas 2D Context* standardit ja oskab nende vahel automaatselt valida [19]. *Pixi.js*'i eesmärk on pakkuda kiiret ja kergekaalulist teeki 2D pildi joonistamiseks [42]. Teek toetab veebilehitsejaid: Internet Explorer 9, Firefox (versioon 10), Google Chrome (versioon 11), Safari (versioon 2.0), Opera (versioon 12) ja nende uuemaid versioone [43].

Objektide pööramiseks ja suuruse muutmiseks leiduvad funktsioonid ning nende kasutamine ei muuda koordinaadisüsteemi. Kiivamise jaoks funktsoon puudub, kuid seda plaanitakse teeki lisada [44]. Objekte saab rühmadesse jaotada [45], kuid mitte kihtidesse jaotada ega kloonida.

Teegi *Pixi.js* suurim eelis teiste teekide ees on suurim kaadrisagedus ja väikseim mälu ning protsessori nõudlikkus. Kasutades *WebGL* standardit oli *Pixi.js* isegi standardiga *Canvas 2D Context* loodud testrakendusest suurema kaadrisagedusega (vt Lisa 4). Mõõtes teegi *Pixi.js* testrakendust sättega *Canvas 2D Context*, oli standardiga *Canvas 2D Context* loodud testrakendus Firefox veebilehitsejas 2,7 korda kiirem ja Google Chrome veebilehitsejas 1,5 korda kiirem. Reklaamitööstuses kasutamiseks *Pixi.js* ei sobiks, kuna on suurem kui 50 KB (vt Lisa 5).

3.8 *Canvas 2D Context* ja JavaScript

Canvas 2D Context on veebistandard veebilehitsejas rastergraafika joonistamiseks. Standardi uusim versioon valmis 2014 aasta augustis [5]. Pilt luuakse standardi poolt *HTML* failis *canvas* elemendi sisse.

Kuigi kõiki transformatsioone saab kasutada, rakendatakse neid lõuendile, mitte objektile. Objekte ei saa rühmadesse ega kihtidesse jaotada. Puudub ka objektide kloonimise võimalus.

Kasutades standardit *Canvas 2D Context* ei ole vaja kaasata lisafaile, mis suurendaksid ribalaiuse kasutamist. Ainus testrakendus mis oli parema kaadrisagedusega kasutas *WebGL* standardit. Tegu on ka mälu ja protsessori kasutamise vaatepunktist säästliku lahendusega. Ilmselt on standardi *Canvas 2D Context* abil on võimalik teha efektiivseim rakendus, kuid vajab ka kõige rohkem rakenduse arendaja poolset tööd. Standardis *Canvas 2D Context* on kõige vähem tööd lihtsustavaid omadusi (vt Tabel 5). Kasutades standardit *Canvas 2D Context* ei sega rakenduse arendajat litsentsi tingimused.

3.9 *SVG* ja JavaScript

SVG on veebistandard veebilehitsejas vektorgraafika esitamiseks. Tegu on *XML* (*Extensible Markup Language*) keelel põhineva standardiga. Hetkel uusim versioon sellest standardist on *Scalable Vector Graphics (SVG) 1.1 (Second Edition)* [46], mis on pärit 2011 aastast. Pilt luuakse standardi poolt *HTML* failis *svg* elemendi sisse.

Transformatsioonide jaoks eraldi funktsioone pole, nende kasutamiseks tuleb objekti *transform* omadust maatriksi või sõne abil muuta. Kõik transformatsioonid muudavad objektile kehtivat koordinaadisüsteemi. Objekte saab jaotada rühmadesse *g* elemendi sisse [47]. Objekte ei saa kihtidesse jaotada ega kloonida.

Kasutades standardit *SVG* ei ole vaja kaasata lisafaile, mis suurendaksid ribalaiuse kasutamist. Testrakendus oli mälukasutuse poolest üks parimatest. Kuigi standard *SVG* pakub palju tööd lihtsustavaid omadusi (vt Tabel 5), on nende kasutamine keerulisem kui teekide puhul. Kasutades standardit *SVG* ei sega rakenduse arendajat litsentsi tingimused.

3.10 Võrdluste tulemused

Selles peatükis analüüsiti eelnevalt kümnest kriteeriumist lähtuvalt kuut teeki ja kahte standardit vastavalt võrdluse metoodikale. Tulemused on esitatud tabeli kujul (vt Tabel 5).

Tabel 5: Teekide võrdluse kokkuvõte

Teek	Objektid	Objektide töötlemine	Tekstitöötlus	Pilditöötlus	Animatsioon	Interaktiivsus	Kasutajatugi	Kogukonna aktiivsus	Litsents	Jõudlus ja ressursinõudlikkus	Kokku
Raphaël	5	4	2	3	5	8	10	2,5	8	4	51,5
Snap.svg	7	3	4	6	5	9	8	6	5	5	58
Svg.js	5	3	6	3	4	8	10	1	8	6	54
Fabric.js	10	4	8	10	8	3	8	10	8	1	70
Paper.js	8	10	2	9	2	8	4	7,5	8	5	63,5
Pixi.js	8	6	2	6	1	7	6	7,5	8	4	55,5
Canvas 2D Context	7	0	0	8	1	3	8	2,5	10	5	44,5
SVG	10	2	4	8	4	9	8	5	10	6	66

Suurimad erinevused teekide vahel olid objektide töötlemise, kogukonna aktiivsuse ja tekstitöötluse osas (vt Tabel 5). Kõige kehvema üldtulemuse sai *Canvas 2D Context* standard ja parima teek *Fabric.js*. Standard *SVG* tuli üldkokkuvõttes teisele kohale. Kuna erinevatel rakendustel on erinevad nõuded, siis tuleks teeki valides kokkuvõtva punktisumma asemel vaadata konkreetseid kriteeriume.

Kõik standardit *SVG* kasutavad teegid ja standardid *SVG* ning *Canvas 2D Context* kõlbavad reklaamitööstuses kasutamiseks. Ükski *canvas* elementi kasutavatest teekidest ei olnud mahult piisavalt väike, et reklaamitööstuses kasutada. Ilmselt sobivad standardid *SVG* ja *Canvas 2D Context* reklaamitööstuses kasutamiseks kõige paremini, kuna nende puhul jääb reklaami enda jaoks kõige rohkem ruumi ja litsentsitingimused ei piira nende

kasutamist. Standardis *SVG* on standardis *Canvas 2D Context* rohkem tööd lihtsustavaid omadusi, kuid on aeglasem ja ressursinõudlikum.

Vastupidiselt reklaamitööstusele sobivad mängutööstuse jaoks paremini *canvas* elementi kasutavad teegid. Kuigi lihtsamaid arvutimängud saab luua ka *SVG* standardit kasutavate teekidega, siis keerulisemate mängude puhul nendest ei piisa. Testrakenduses oli 2500 lumehelbe korral standard *Canvas 2D Context* standardist *SVG* Firefox veebilehitsejas rohkem kui 8 korda kiirem (vt Lisa 5). Veebilehitsejas Google Chrome oli 2500 lumehelbe korral standard *Canvas 2D Context* standardist *SVG* rohkem kui 4,5 korda kiirem (vt Lisa 5). Lisaks oli ka protsessori ja mälu kasutus standardil *Canvas 2D Context* standardist *SVG* parem (vt Lisa 5). *Pixi.js* on *WebGL* standardit kasutades standardist *Canvas 2D Context* kiirem. Kuna standardit *WebGL* toetab vähem veebilehitsejaid kui standardit *Canvas 2D Context*, siis oleneb rakenduse nõuetest, kas see on eelis või mitte.

Kokkuvõte

Töö eesmärgiks oli analüüsida kuut populaarset JavaScript 2D graafika teeki ning standardeid *SVG* ja *Canvas 2D Context*. Võrdlemiseks kasutati autori poolt välja töötatud võrdluse kriteeriumit ja metoodikat. Lisaks tehti iga teegi ja standardiga läbi praktiline ülesanne.

Võrdluse kriteerium ja metoodika loodi keskendudes arvutimängutööstusele ja reklaamitööstusele. Võrdluse metoodika eesmärk ei olnud luua paremusjärjekorda, kuna igal rakendusel on siiski eraldi nõuded. Võrdluse metoodikasse valiti omadusi, mis vähendavad rakenduse arendaja poolt tehtavat tööd. Lisaks mainiti eraldi omadust, mis välistab teekide kasutamist reklaamitööstuses. Testrakendust kasutades näidati kui palju on standard seda kasutavatest teekidest kiirem ja mälusäästlikum. Järelduste loomiseks tugineti nii võrdluse metoodikale kui ka praktilisele ülesandele.

Võrdluse tulemusena selgus, et standardit *SVG* kasutavad teegid ja standardid *Canvas 2D Context* ning *SVG* sobivad reklaamitööstuses kasutamiseks. Kuna *canvas* elementi kasutavad teegin on liiga suured, ei saa neid reklaamitööstuses kasutada.

Testrakenduseks loodi lumesaju animatsioon, kus lumehelveste arv oli muutuja. Lumesaju animatsioon valiti, kuna seda on talvehooajal Internetis tihti kasutatud. Lisaks sai lumesaju animatsiooni abil sai mõõta objektide arvu mõju kaadrisagedusele, mälu kasutusele ja protsessori koormusele. Testrakenduse mälu kasutamist, protsessori koormust ja kaadrisagedust mõõdeti kahe veebilehitsejaga.

Teekidest ja standarditest kiireim oli *Pixi.js*, kui see kasutas joonistamiseks *WebGL* standardit. Kiiruselt teine oli *Canvas 2D Context*, mis oli kiirem kõikidest standardit *Canvas 2D Context* kasutavatest testrakendustest. *Pixi.js*, kasutades joonistamiseks *Canvas 2D Context* standardit, ja *Fabric.js* olid sarnase kiirusega ning nende täpne järjestus sõltus veebilehitsejast. Kõige vähem mälu kasutasid standarditega *SVG* ja *Canvas 2D Context* ning teegiga *Pixi.js* loodud testrakendused. Protsessorit kasutasid kõige säästlikumalt teegid *Pixi.js*, *Fabric.js* ja standard *Canvas 2D Context*.

Bakalaureusetöös läbi viidud analüüsi tulemusi sobib kasutada võrdluse all olevate teekide ning standardite hulgast sobivaima valimiseks. Analüüsi tulemuste hulgas on soovituslik vaadata konkreetsete kriteeriumite hinnanguid, mitte kokkuvõtvat punktisummat.

Viited

[1] "HTML5 for Digital Advertising." [WWW] http://www.iab.net/media/file/HTML5DAv101.pdf, 2013. (10.03.2015).

[2] Adobe Systems Software Ireland Ltd., "Adobe Flash Professional CC." [WWW] http://www.adobe.com/products/flash.html. (25.04.2015).

[3] J. Percival, *HTML5 Advertising*. Apress, detsember 2012.

[4] W3C, "HTML5." [WWW] http://www.w3.org/TR/html5/. (12.04.2015).

[5] W3C, "HTML Canvas 2D Context." [WWW] http://www.w3.org/TR/2dcontext/. (24.03.2015).

[6] Khronos Group, "WebGL Specification." [WWW] https://www.khronos.org/registry/webgl/specs/1.0/. (12.04.2015).

[7] A. Rinde, "Multimeedium, sissejuhatus, meedia." [WWW] http://www.cs.tlu.ee/~rinde/mm_materjal/pdf/mm_algus_meedia.pdf. (13.03.2015).

[8] Mark Pilgrim, "Dive Into HTML5." [WWW] http://diveintohtml5.info/canvas.html. (25.04.2015).

[9] W3C, "Coordinate Systems, Transformations and Units – SVG 1.1 (Second Edition)." [WWW] http://www.w3.org/TR/SVG/coords.html. (24.03.2015).

[10] W3C, "Painting: Filling, Stroking and Marker Symbols – SVG 1.1 (Second Edition)." [WWW] http://www.w3.org/TR/SVG/painting.html. (24.03.2015).

[11] "Sub-pixel offset inconsistent across browsers and OSes. #175." [WWW] https://github.com/DmitryBaranovskiy/raphael/issues/175. (24.03.2015).

[12] W3C, "World Wide Web Consortium Process Document." [WWW] http://www.w3.org/2014/Process-20140801/. (13.03.2015).

[13] "FAQ - WHATWG Wiki." [WWW] https://wiki.whatwg.org/wiki/FAQ#What_does_.22Living_Standard.22_mean.3F. (30.03.2015).

[14] D. Baranovskiy, "Raphaël." [WWW] http://raphaeljs.com/. (09.03.2015).

[15] D. Baranovskiy, "Snap.svg." [WWW] http://snapsvg.io/. (09.03.2015).

[16] W. Fierens, "svg.js." [WWW] http://svgjs.com/. (09.03.2015).

[17] Juriy Zaytsev, Stefan Kienzle, Andrea Bogazzi, "Fabric.js." [WWW] http://fabricjs.com/. (09.03.2015).

[18] "Paper.js." [WWW] http://paperjs.org/. (09.03.2015).

[19] Goodboy Digital Ltd., "Pixi.js - 2D webGL renderer with canvas fallback." [WWW] http://www.pixijs.com/. (09.03.2015).

[20] A. Rinde, "Multimeediumi elementide kasutamine." [WWW] http://www.cs.tlu.ee/~rinde/mm_materjal/pdf/mm_elementide_kasutamine.pdf. (10.03.2015).

[21] "SVG Shapes." [WWW] https://github.com/mbostock/d3/wiki/SVG-Shapes. (25.04.2015).

[22] Microsoft, "Easing Functions." [WWW] https://msdn.microsoft.com/en-us/library/ee308751%28v=vs.110%29.aspx. (25.04.2015).

[23] Craig Buckler, "How to Use the Mouse Wheel Event in HTML5 Pages." [WWW] http://www.sitepoint.com/html5-javascript-mouse-wheel/. (07.05.2015).

[24] Microsoft, "SVG vs canvas: how to choose." [WWW] https://msdn.microsoft.com/en-us/library/ie/gg193983%28v=vs.85%29.aspx. (10.03.2015).

[25] "HTML5 Ad Specifications." [WWW] https://nz.adspecs.yahoo.com/pages/html5-ad-sepcs/. (10.03.2015).

[26] "Closure Compiler." [WWW] https://developers.google.com/closure/compiler/. (10.03.2015).

[27] "Performance - Firefox Developer Tools | MDN." [WWW] https://developer.mozilla.org/en-US/docs/Tools/Performance. (13.03.2015).

[28] Refsnes Data, "Browser Statistics." [WWW] http://www.w3schools.com/browsers/browsers_stats.asp. (25.04.2015).

[29] "about:memory - Mozilla | MDN." [WWW] https://developer.mozilla.org/en-US/docs/Mozilla/Performance/about:memory. (25.04.2015).

[30] "Force browser processes to close - Chrome Help." [WWW] https://support.google.com/chrome/answer/95672?hl=en. (25.04.2015).

[31] "htop - an interactive process viewer for Linux." [WWW] http://hisham.hm/htop/. (25.04.2015).

[32] "Kubuntu | Friendly Computing." [WWW] http://www.kubuntu.org/. (07.04.2015).

[33] Intel Corporation, "ARK | Intel®Core™ i7-3630QM Processor (6M Cache, up to 3.40 GHz)." [WWW] http://ark.intel.com/products/71459/Intel-Core-i7-3630QM-Processor-6M-Cache-up-to-3_40-GHz. (07.04.2015).

[34] D. Baranovskiy, "Raphaël." [WWW] http://raphaeljs.com/reference.html. (20.03.2015).

[35] "Snap.svg - Why Snap." [WWW] http://snapsvg.io/about/. (26.03.2015).

[36] "svg.js." [WWW] http://documentup.com/wout/svg.js. (26.03.2015).

[37] "Javascript 2D Canvas Libraries." [WWW] https://spreadsheets.google.com/pub?hl=en&hl=en&key=0Aqj_mVmuz3Y8dHNhUVFDYlRaaXlyX0xYSTVnalV5ZlE&output=html. (28.03.2015).

[38] "kangax/fabric.js · GitHub." [WWW] https://github.com/kangax/fabric.js/. (28.03.2015).

[39] "JSDoc:Index." [WWW] http://fabricjs.com/docs/. (29.03.2015).

[40] "Paper.js – About." [WWW] http://paperjs.org/about/. (29.03.2015).

[41] "Paper.js – Reference." [WWW] http://paperjs.org/reference/. (29.03.2015).

[42] "GoodBoyDigital/pixi.js · GitHub." [WWW] https://github.com/GoodBoyDigital/pixi.js/. (29.03.2015).

[43] "FAQs · GoodBoyDigital/pixi.js Wiki · GitHub." [WWW] https://github.com/GoodBoyDigital/pixi.js/wiki/FAQs#what-browsers-are-supported. (29.03.2015).

[44] "Is there a skew property coming? #612 ." [WWW] https://github.com/GoodBoyDigital/pixi.js/issues/612. (29.03.2015).

[45] "DisplayObjectContainer - pixi.js." [WWW] http://www.goodboydigital.com/pixijs/docs/classes/DisplayObjectContainer.html. (29.03.2015).

[46] W3C, "SVG Current Status." [WWW] http://www.w3.org/standards/techs/svg. (30.03.2015).

[47] W3C, "Document Structure – SVG 1.1 (Second Edition)." [WWW] www.w3.org/TR/SVG/paths.html. (30.03.2015).

Lisad

Lisa 1

Teekide- ja standarditega loodud testrakendus koos lähtekoodiga on salvestatud tööga kaasas olevas zip failis kaustas nimega testrakendus. Lisaks testrakendusele endale on samas kaustas ka kasutusjuhend.

Lisa 2

Teekidega loodud transformatsioonide testimise lähtekood on salvestatud tööga kaasas olevas zip failis kaustas nimega transformatsioonid.

Lisa 3

Teekide- ja standarditega loodud interaktiivsuse testimise lähtekood on salvestatud tööga kaasas olevas zip failis kaustas nimega interaktiivsus. Testitud on ainult hiire interaktiivsuse omadusi.

Lisa 4

Teekidega loodud testrakenduse mõõtmistulemused tabelite kujul on töö kaasas olevas zip failis oleva faili tulemus.ods sees.

Lisa 5

Teekide hinnangute põhjendused koos vastavate viidetega asuvad tööga kaasas olevas zip failis oleva faili lisa5.pdf sees.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks.

Mina **Priit Pihlamägi** (sünnikuupäev: 19.11.1989)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
"**JavaScript 2D graafika teekide võrdlus**",
mille juhendaja on Siim Karus,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus **07.05.2015**