Onderzoek

Versie 1

Gemaakt door:
David Kerkkamp (521142)
Thomas Peters (537702)
Joeri Smits (524292)
Serhat Tunç (491186)
Leroy Witteveen (523896)

Course:
DWA Project

Begeleiders:
Theo Theunissen
Lars Tijsma

Datum:
14 november 2014

1 Probleemstelling

1.1 De huidige situatie

Op dit moment heeft de projectgroep nog geen duidelijk inzicht in welke libraries er beschikbaar zijn voor onze functionaliteiten binnen de Percolator. De projectgroep weet wel dat er libraries beschikbaar zijn voor de volgende functionaliteiten:

- Geheugenmodellen tekenen
- Een teksteditor voor het weergeven van code in de browser
- Real-time (hagerAly, 2013) communicatie tussen gebruikers.

1.2 De gewenste situatie

De projectgroep wil weten welke librarie het meeste geschikt is voor de bovenstaande functionaliteiten. Daarnaast wil de groep ook graag weten of er wijzigingen kunnen worden aangebracht aan de bestaande libraries.
Sommige projectleden hebben al ervaring met bepaalde libraries. Deze ervaring komt goed van pas tijdens dit onderzoek. Waar de projectgroep voor moet uitkijken is dat er geen subjectieve voorkeur komt voor een bepaalde library.

1.3 Het verschil tussen de huidige en gewenste situatie

Op het huidige moment is het probleem dat de projectgroep nog geen kennis heeft van de benodigde libraries voor de Percolator. In dit onderzoek gaat de projectgroep onderzoek doen naar de verschillende libraries.
De gewenste situatie is dan ook dat de projectgroep kennis heeft over welke libraries geschikt zijn voor de bovenstaande functionaliteiten.

Het verschil tussen deze twee situaties is dus de kennis van de projectgroep over de verschillende libraries. De projectgroep wil deze kennis op doen door middel van onderzoek te doen.

2 Doelstelling

De projectgroep zoekt drie geschikte libraries om geheugenmodellen te tekenen, real-time samen te werken, en code weer te geven.

Voor een geschikte library zoekt de projectgroep kennis voor een aantal functionaliteiten die bij onze vraagstelling worden genoemd. Om deze kennis te krijgen moet de projectgroep onderzoek doen naar de verschillende functionaliteiten per library. Goede manieren om deze kennis te verkrijgen zijn:

- De officiële documentatie (indien aanwezig)
- Andere gerelateerde bronnen over de library
- Zelf experimenteren
- Voorbeelden bekijken waarin de library is gebruikt.

Wanneer is een library een goede library?
De projectgroep beoordeeld de libraries op basis van de hoeveelheid functionaliteiten die mogelijk zijn met die library. Deze functionaliteiten staan beschreven onder de vraagstelling. De library met de meeste functionaliteiten komt voor het project het beste uit de test en wordt in gebruik genomen.

Vraagstellingen

Vraagstelling 1:
Welke grafische library kan de projectgroep het best gebruiken om geheugenmodellen te kunnen tekenen?

De eisen aan een grafische tekenlibrary zijn:

1. Figuren tekenen (vierkanten, rechthoeken, etc)
2. Tekst plaatsen in de figuren
3. Positie en grootte van figuren kunnen aanpassen
4. Figuren verbinden door middel van lijnen en pijlen
5. Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON)
6. Makkelijk te implementeren

Vraagstelling 2:
Welke teksteditor kan de projectgroep het best gebruiken om de code bij de geheugenmodellen goed weer te geven?

De eisen aan een teksteditor zijn:

1. Ondersteuning van meerdere programmeertalen (in ieder geval Java en C#)
2. Syntax highlighting en indentatie
3. Regel nummering
4. Geen merkbare performance impact
5. Makkelijk te implementeren
6. Goede documentatie

Vraagstelling 3:
Welke technologie gebruikt de projectgroep om de realtime interactie tussen verschillende gebruikers te faciliteren?

De eisen aan een implementatie van real-time interactie zijn:

1. Zo min mogelijk merkbare latentie
2. Makkelijk te implementeren
3. Zo veel mogelijk browser compatibiliteit

Data verzamelen

Mogelijke grafische libraries:

- RaphaëlJS - http://raphaeljs.com/
- Draw2D - http://www.draw2d.org/
- D3.JS - http://d3js.org/
- BonsaiJS - http://bonsaijs.org/
- ThreeJS - http://mrdoob.github.io/three.js/
- FabricJS - http://fabricjs.com/
- EaselJS - http://www.createjs.com/#!/EaselJS
- PaperJS - http://paperjs.org/
- OCanvas - http://ocanvas.org/

Mogelijke teksteditors:

- Ace - http://ace.c9.io
- CodeMirror - http://codemirror.net
- SyntaxHighlighter - https://code.google.com/p/syntaxhighlighter/
- Edit area - http://www.cdolivet.com/editarea/
- CodePress - http://codepress.sourceforge.net
- LDT - https://github.com/kueblc/LDT/

Mogelijke technologiëen voor realtime samenwerken:

- Zelf implementeren bovenop socket.io
- Zelf implementeren bovenop WebRTC
- TogetherJS gebruiken - https://togetherjs.com

Data analyseren

Mogelijke grafische libraries:

Library	Vierkanten tekenen	Tekst in vierkant	Versleepbaar	Lijnen kunnen trekken	Exporteren naar bv. JSON	Documentatie	Rang
RaphaëlJS	ja	ja	ja	ja	nee	goed	**4**
BonsaiJS	ja	nee	ja	nee	nee	matig	**3**
FabricJS	ja	ja	ja	ja	ja	zeer goed	**5**
PaperJS	ja	nee	nee	deels	ja	matig	**2**
OCanvas	ja	nee	nee	deels	nee	goed	**1**

De volgende libraries zijn uit het onderzoek verwijderd, omdat ze niet voldoen aan de eisen.
Dit zijn 3D grafische libraries:

- Draw2D - http://www.draw2d.org/
- D3.JS - http://d3js.org/
- ThreeJS - http://mrdoob.github.io/three.js/
- EaselJS - http://www.createjs.com/#!/EaselJS

Onderzoek:

RaphaëlJS

1. Vierkanten tekenen: ja
    a. Deze library heeft eenvoudige functies met attributen waarmee je verschillende vormen kunt tekenen.
    b. Voorbeeld om een cirkel te tekenen:

```
// Creates canvas 320 × 200 at 10, 50
var paper = Raphael(10, 50, 320, 200);

// Creates circle at x = 50, y = 40, with radius 10
var circle = paper.circle(50, 40, 10);
// Sets the fill attribute of the circle to red (#f00)
circle.attr("fill", "#f00");

// Sets the stroke attribute of the circle to white
circle.attr("stroke", "#fff");
```

2. Tekst in vierkant plaatsen: ja
    a. Met deze functie kun je tekst schrijven op een bepaald x- en y-positie.
    b. Voorbeeld om tekst te plaatsen:

```
var t = paper.text(50, 50, "Raphaël\nkicks\nbutt!");
```

Paper.text(x, y, text)

Draws a text string. If you need line breaks, put "\n" in the string.

3. Versleepbaar: ja
    a. Met deze functie kun je aangemaakte vormen verslepen met de muis.

Element.drag(onmove, onstart, onend, [mcontext], [scontext], [econtext]) ⚓ ➪

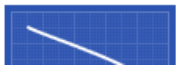
Adds event handlers for drag of the element.

4. Lijnen kunnen trekken: twijfel
    a. Met deze functie kun je lijnen tekenen op een bepaald x- en y-positie. Je zou dit kunnen gebruiken om handmatig vormen te verbinden.
    b. Voorbeeld om een lijn te tekenen:

```
var c = paper.path("M10 10L90 90");
// draw a diagonal line:
// move to 10,10, line to 90,90
```

Paper.path([pathString])

Creates a path element by given path data string.



5. Exporteren naar JSON: nee
6. Documentatie: zeer goed
    a. Alle functies inclusief de attributen zijn goed gedocumenteerd en uitgewerkt.
    b. Voor bijna elke functie is een voorbeeld code geschreven om je op weg te helpen.

BonsaiJS

1. Vierkanten tekenen: ja
    a. Deze library heeft verschillende functies met attributen waarmee je allerlei vormen mee kunt tekenen.
    b. Voorbeeld om een vierkant te tekenen:

```
new Rect(x, y, width, height, [cornerRadius])
```

2. Tekst in vierkant plaatsen: nee
3. Versleepbaar: ja
    a. Met deze functie kun je aangemaakte vormen verslepen met de muis.

TOUCH EVENTS

By default, you can bind the above pointer events and they should work as you would for single-finger interactions.

For multitiple-finger interactions, you should bind to the following prefixxed events:

- `multi:pointerup`
- `multi:pointerdown`
- `multi:pointermove`
- `multi:drag`

All `multi:` events will have a `touchId` in the passed event object, which uniquely identifies the finger across all touch sessions, so you can keep track of the fingers currently on screen. i.e.

4. Lijnen kunnen trekken: nee
5. Exporteren naar JSON: nee
6. Documentatie: matig

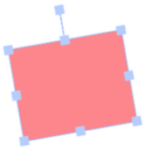
FabricJS

1. Figuren tekenen ja
    a. Deze library heeft functies om verschillende figuren als objecten aan te maken
    b. Het onderstaande stuk code is een simpel voorbeeld van een manier om een figuur, in dit geval een vierkant, aan te maken in FabricJS. Hierin kunnen eigenschappen zoals de positie, grootte, hoek en kleur worden meegegeven, zie afbeelding f1 (FabricJS, Controls customization demo).

```
var rect = new fabric.Rect({
  left: 150,
  top: 200,
  originX: 'left',
  originY: 'top',
  width: 150,
  height: 120,
  angle: -10,
  fill: 'rgba(255,0,0,0.5)',
  transparentCorners: false
});
```

Afbeelding f1

2. Tekst plaatsen in de figuren ja
    a. FabricJS beschikt over een aantal functies om tekst aan het canvas toe te voegen en deze vorm te geven doormiddel van eigenschappen
    b. In FabricJS is het ook mogelijk een groep van FabricJS objecten aan te maken als één entiteit. Op die manier is het mogelijk om een figuur met tekst erin te behandelen als één object (FabricJS, Introduction to Fabric.js).
3. Positie en grootte van figuren kunnen aanpassen ja
    a. Getekende figuren kunnen worden versleept en geroteerd en de grootte is aan te passen
    b. Door met de muis in het figuur te klikken en vervolgens te slepen, kan het figuur worden verplaatst. Door in een hoek te klikken en te slepen, wordt de grootte aangepast. Als laatste kan de hoek worden aangepast zoals te zien is in afbeelding f2.



Afbeelding f3

4. Figuren verbinden door middel van lijnen en pijlen ja
    a. Het is mogelijk om in FabricJS lijnen te tekenen en deze vast te maken aan andere figuren.
    b. Door eerst een lijn aan te maken en vervolgens figuren te tekenen op de x en y coördinaten van die lijn kunnen figuren worden verbonden zoals in afbeeldingen f3 en f4 (FabricJS, Stickman demo) te zien is.

```
var line = makeLine([ 250, 125, 250, 175 ])
```

Afbeelding f3

```
canvas.add(
  makeCircle(line.get('x1'), line.get('y1'), null, line),
  makeCircle(line.get('x2'), line.get('y2'), line, line2, line5, line6),
```

Afbeelding f4

5. Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) ja

a. De data van alle getekende elementen is te exporteren naar JSON.

b. Met de methode canvas.toJSON() worden alle elementen op het canvas in een JSON object gestopt. Met de methode canvas.loadFromJSON() is deze data vervolgens weer in te laden (FabricJS, Introduction to Fabric.js).

1. Goede documentatie uitstekend
    1. FabricJS heeft een uitgebreide documentatie met voorbeeldcode
    2. Er zijn veel functies om eigenschappen van getekende figuren op te vragen en bij te werken.

PaperJS

1. Figuren tekenen ja
    a. Het is mogelijk om verschillende soorten figuren, zoals cirkels te tekenen
    b. Onderstaand stukje code (afbeelding p1) geeft een voorbeeld van het tekenen van een cirkel

```
var circle2 = new Path.Circle(new Point(120, 50), 35);
circle2.style = {
    fillColor: 'blue',
    strokeColor: 'green',
    strokeWidth: 10
};
```

Afbeelding p1

2. Tekst plaatsen in de figuren nee
    a. Het plaatsen van tekst in het canvas.loadFromJSON is mogelijk (volgens PaperJS documentatie), maar de tekst wordt als afbeelding op het scherm weergegeven
    b. In afbeelding p2 wordt een PointText aangemaakt

```
var text = new PointText({
    point: [50, 50],
    content: 'The contents of the point text',
    fillColor: 'black',
    fontFamily: 'Courier New',
    fontWeight: 'bold',
    fontSize: 25
});
```

Afbeelding p2

3. Positie en grootte van figuren kunnen aanpassen gedeeltelijk

a. Het is mogelijk figuren te verplaatsen, maar dat moet gebeuren via functies. Er is geen ingebouwde functionaliteit voor het verslepen van figuren (PaperJS, Transforming Items).

4. Figuren verbinden door middel van lijnen en pijlen gedeeltelijk
   a. Het is in PaperJS mogelijk om lijnen/pijlen te tekenen, maar dit moet op een ingewikkelde manier en er is geen ingebouwde functionaliteit voor het vastmaken van pijlen aan andere figuren. Het tekenen van deze lijnen is ook eigenlijk voor wiskundige doeleinden (PaperJS, Vector Geometry).
   b. In afbeelding p3 wordt een voorbeeld getoond

```
var point1 = new Point(50, 0);
```

Afbeelding p3

5. Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) ja
   a. Via de functies exportJSON() en importJSON() is het mogelijk de data van alle objecten op het canvas te exporteren of te importeren (PaperJS, Importing / Exporting JSON and SVG).
6. Goede documentatie matig
   a. Er is documentatie aanwezig, er worden echter maar weinig functies in genoemd
   b. Niet bij alle beschreven functies is voorbeeldcode aanwezig

oCanvas

1. Figuren tekenen ja
   a. In oCanvas is het mogelijk figuren zoals vierkanten te tekenen (volgens oCanvas documentatie)
   b. Afbeelding o1 toont een voorbeeld waarin een rechthoek wordt aangemaakt

```
var rectangle = canvas.display.rectangle({
    x: 77,
    y: 77,
    width: 200,
    height: 100,
    fill: "#0aa"
});
```

Afbeelding o1

2. Tekst plaatsen in de figuren nee
    a. Er zijn functies aanwezig om tekst op het canvas te plaatsen, maar niet om deze in of op een ander object te plaatsen (volgens oCanvas documentatie).
    b. In afbeelding o2 wordt een nieuw text object aangemaakt en er worden een aantal eigenschappen meegegeven

```
var text = canvas.display.text({
    x: 177,
    y: 107,
    origin: { x: "center", y: "top" },
    font: "bold 30px sans-serif",
    text: "Hello World!",
    fill: "#0aa"
});
```

Afbeelding o2

3. Positie en grootte van figuren kunnen aanpassen gedeeltelijk
    a. Het is mogelijk figuren te verplaatsen, maar dat moet gebeuren via functies. Er is geen ingebouwde functionaliteit voor het verslepen van figuren.

4. Figuren verbinden door middel van lijnen en pijlen gedeeltelijk
    a. Het is via een functie mogelijk om lijnen te tekenen
    b. Onderstaande afbeelding (afbeelding o3), toont een voorbeeld waarin een lijn wordt getekend

```
var text = canvas.display.text({
    x: 177,
    y: 107,
    origin: { x: "center", y: "top" },
    font: "bold 30px sans-serif",
    text: "Hello World!",
    fill: "#0aa"
});
```

Afbeelding o3

5. Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) Nee
    a. Er is geen functionaliteit gevonden voor het exporteren naar een bestandstype dat makkelijk in een database kan worden opgeslagen
6. Goede documentatie Ja
    a. De functies die er zijn, zijn gedocumenteerd met voorbeeldcode

Teksteditors

In de percolator moet er de mogelijkheid zijn om code op de webpagina te schrijven. Deze code hoeft natuurlijk niet uitgevoerd te worden, maar het moet wel weer kunnen geven welke regel code bij het huidige geheugenmodel hoort. Hier zal dus een soort highlighting beschikbaar moeten zijn. Uiteraard moet de code ook duidelijk leesbaar zijn. Een stuk code zonder opmaak en standaard letterkleur is niet te onderscheiden van een normaal stukje tekst. De teksteditor zal dus functies, statements en variabelen zichtbaar moeten onderscheiden van elkaar. Dit kan gebeuren door ze een aparte kleur of opmaak te geven.

Requirements:

- Structuur
    - Dit wil zeggen dat de code leesbaar wordt door gebruik te maken van indentatie.
- Onderscheidt gereserveerde woorden
    - Dit moet worden gedaan door gereserveerde woorden een aparte kleur te geven en/of tekstopmaak toe te passen.
- Programmeertalen
    - De perculator zou voor meerdere programmeertalen gebruikt kunnen worden. Deze talen moeten natuurlijk wel ondersteund worden door de teksteditor.

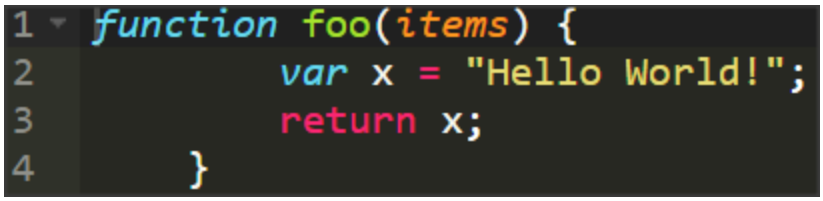
De volgende teksteditors zullen worden onderzocht:

- Ace - http://ace.c9.io
- CodeMirror - http://codemirror.net
- SyntaxHighlighter - https://code.google.com/p/syntaxhighlighter/
- Edit area - http://www.cdolivet.com/editarea/
- CodePress - http://codepress.sourceforge.net
- LDT - https://github.com/kueblc/LDT/

Ace

Samenvatting:

Past structuur toe	**Onderscheidt gereserveerde woorden**	**Aantal programmeertalen**	**Documentatie** **(slecht, matig, goed)**
Ja	Ja	110	matig



Structuur:
Zoals op bovenstaande foto is te zien, zorgt Ace voor een gestructureerde weergave van de code. Er is duidelijk indentatie toegepast.

Onderscheidt gereserveerde woorden:
Zoals op bovenstaande foto is te zien, hebben de woorden verschillende kleuren. Op deze manier is het gemakkelijk om functies, variabelen en gereserveerde woorden van elkaar te onderscheiden.

Aantal programmeertalen:
Ace ondersteunt maar liefst 110 verschillende programmeer talen(Waaronder Java en JavaScript). Echter, in de broncode moet worden aangegeven welke programmeertaal op de webpagina ondersteunt wordt. Er zal dus een functie moeten worden geschreven waarmee je de programmeertaal aan kunt passen. Dit is dus een eventueel risico.

Documentatie:
Alle mogelijke functies zijn gedocumenteerd. Hier staan echter geen voorbeelden bij. Enkel een korte omschrijving(1-2 zinnen) bij elke methode moet voor verduidelijking zorgen.

(APA Reference nog maken?)

CodeMirror

Realtime samenwerken

Zelf implementeren bovenop Socket.IO

1. Zo min mogelijk merkbare latentie:

Latentie is afhankelijk van de kwaliteit van de verbinding tussen client en server. Het is mogelijk zelf kleine verbeteringen hier op aan te brengen, zoals meerdere events bundelen en versturen in één keer.

2. Makkelijk te implementeren:

Het is redelijk lastig om klikken, muis slepen en typen tussen verschillende gebruikers te delen te bouwen vanaf scratch. Ook zul je zelf disconnects moeten afvangen.

3. Zo veel mogelijk browser compatibiliteit:

Socket.IO is gebaseerd op Web Sockets. Web Sockets wordt ondersteund door 84.4% (Can I use websockets, 2014) van alle browsers. Wanneer Web Sockets niet worden ondersteund, valt Socket.IO terug op Flash sockets, JSONP polling of AJAX long polling. Dit maakt dat elke browser die het zou kunnen ondersteunen wordt ondersteund.

Zelf implementeren bovenop WebRTC

1. Zo min mogelijk merkbare latentie:

Omdat WebRTC een peer to peer protocol is, ben je afhankelijk van de internet snelheid van de gebruikers om een snelle ervaring te kunnen leveren. De kans is erg groot dat een gebruiker een minder snelle internetverbinding heeft dan een server in een datacenter. Wanneer een client een verandering aanbrengt zal hij deze moeten sturen naar alle andere clients, wat zorgt voor hogere eisen voor de clients dan in een client-server model.

2. Makkelijk te implementeren:

WebRTC is gemaakt voor peer to peer (client naar client) communicatie, welke lastiger te implementeren is dan een client-server model. Daarnaast moet alles dat bij Socket.IO moet worden geïmplementeerd moet hier ook worden gedaan.

3. Zo veel mogelijk browser compatibiliteit:

WebRTC wordt ondersteund door 55.44% (Can I use WebRTC, 2014) van alle browsers. Bij desktop gebruikers vallen Internet Explorer en Safari buiten de boot.

TogetherJS gebruiken

1. Zo min mogelijk merkbare latentie:

Net iets minder latentie dan Socket.IO omdat het native Web Sockets gebruikt. Je kunt instellen dat alleen clicks op het canvas verwerkt worden. Dit heeft minder netwerkverkeer als resultaat, wat de latentie ten goede komt.

2. Makkelijk te implementeren:

TogetherJS heeft het realtime samenwerken al ingebouwd. (Voice)chat, klikken en typen wordt gedeeld met andere gebruikers. Wel moet TogetherJS een beetje aangepast worden wil je events als bijvoorbeeld je muis slepen gebruiken.

3. Zo veel mogelijk browser compatibiliteit:

Web Sockets worden ondersteund door 84.4% (Can I use websockets, 2014) van alle browsers. WebRTC wordt ondersteund door 55.44% (Can I use WebRTC, 2014) van alle browsers. Wanneer WebRTC niet ondersteund wordt werkt alleen chatten niet, dit is dus geen groot probleem. Bij desktop gebruikers vallen bij het Web Sockets gedeelte IE9 en lager buiten de boot.

Conclusie:

Socket.IO heeft de beste browser compatibiliteit, TogetherJS is veruit het makkelijkst te implementeren en Socket.IO heeft het minst merkbare latentie.
TogetherJS komt als beste uit het onderzoek want:

- Het verschil in latentie is redelijk klein
- Een browser compatibiliteit van 84.4% is niet slecht, alleen oude en enkele kleine mobiele browsers worden niet ondersteund
- Het feit dat TogetherJS makkelijk te implementeren is, is voor de projectgroep het belangrijkste

Bibliografie

- hagerAly. (2013, november 3). *Real Time Web Development - Brief*. Opgeroepen op 11 november, 2014, van Code project: http://www.codeproject.com/Tips/677819/Real-Time-Web-Development-Brief
- Can I use websockets. (2014, oktober). *Can I use websockets?*. Opgeroepen op 14 november, 2014, van caniuse.com: http://caniuse.com/#search=websockets
- Can I use WebRTC. (2014, oktober). *Can I use webrtc?*. Opgeroepen op 14 november, 2014, van caniuse.com: http://caniuse.com/#search=webrtc
- FabricJS. (geen datum). *Fabric.js demos · Stickman*. Geraadpleegd op 14 november 2014, van *http://fabricjs.com/stickman/*
- FabricJS. (geen datum). *Fabric.js demos · Controls customization*. Geraadpleegd op 14 november 2014, van *http://fabricjs.com/controls-customization/*
- FabricJS. (geen datum). *Introduction to Fabric.js. Part 3.* Geraadpleegd op 14 november 2014, van http://fabricjs.com/fabric-intro-part-3/
- FabricJS. (geen datum). *FabricJS Docs.* Geraadpleegd op 14 november 2014, van http://fabricjs.com/docs/
- PaperJS. (geen datum). *Transforming Items.* Geraadpleegd op 14 november 2014, van *http://paperjs.org/tutorials/project-items/transforming-items/*
- PaperJS. (geen datum). *PointText.* Geraadpleegd op 14 november 2014, van *http://paperjs.org/reference/pointtext/*
- PaperJS. (geen datum). *Vector Geometry.* Geraadpleegd op 14 november 2014, van *http://paperjs.org/tutorials/geometry/vector-geometry/*
- *oCanvas. (geen datum). Text. Geraadpleegd op 14 november 2014, van http://ocanvas.org/docs/Display-Objects/Text*
- *oCanvas. (geen datum). Rectangle. Geraadpleegd op 14 november 2014, van http://ocanvas.org/docs/Display-Objects/Rectangle*
- *oCanvas. (geen datum). Line. Geraadpleegd op 14 november 2014, van http://ocanvas.org/docs/Display-Objects/Line*
- *oCanvas. (geen datum). Documentation. Geraadpleegd op 14 november 2014, van http://ocanvas.org/docs*