Masterproef
Augmented video
3D virtuele wereld-gekoppelde en interactieve video playback

Professor Wim Lamotte

Professor Peter Quax

Dr. Maarten Wijnants

Jeroen Leën

Voorwoord

Na het eerste gesprek over de mogelijke masterthesisonderwerpen wist ik dat ik deze masterthesis wou onderzoeken. Het onderwerp “Augmented video” sprak me aan omdat ik een interesse heb in televisie en video. Onderzoeken hoe een interactieve video werkt, gecombineerd met een virtuele 3D-wereld, leek mij boeiend. Tijdens mijn onderzoek heb ik in samenspraak met dr. Wijnants de opzet verbreed naar het definiëren van overlay-elementen aan de hand van motiontrack data. Ook het registreren van gebruikeracties aan de hand van web analytics tools hoorde bij deze uitbreiding. Als laatste werd het augmented video systeem toegepast op een omnidirectionele video. Het doel van deze thesis werd dan ook om onderzoek te verrichten naar augmented video. Ik wil alvast dr. Maarten Wijnants, professor Wim Lamotte en professor Peter Quax bedanken voor hun nuttige suggesties en hulp tijdens het schrijven van deze thesis. Ik wens u veel plezier bij het lezen.

Inhoudstafel

Inhoudstafel Afbeeldingen

Inleiding

Het onderwerp van deze masterproef is “augmented video”. Daarmee wordt bedoeld dat er een laag toegevoegd wordt aan een video. Dat maakt de video interactiever of zorgt ervoor dat die meer informatie bevat. In deze masterproef gaan we daarom een video uitbreiden op twee manieren. De eerste manier is het koppelen van een virtuele 3D-wereld aan een externe videoplayer. Hierdoor kunnen gebeurtenissen in de video een effect uitvoeren op de 3D-wereld, maar het omgekeerde is ook mogelijk. De tweede manier waarop we video gaan uitbreiden is aan de hand van interactieve video. Met interactieve video wordt een video bedoeld waarbij de kijker niet enkel passief moet kijken naar de video, maar ook de mogelijkheid krijgt om acties op deze video uit te voeren. Deze acties kunnen gevolgen hebben op de site waar de video staat of in de video zelf. In deze masterthesis gaan we overlay-elementen definiëren waar de gebruiker kan mee interageren. Het kan hier gaan over overlay-elementen die extra informatie tonen op een bepaald moment of aanklikbare overlay-elementen waarbij er een actie zal worden uitgevoerd na het aanklikken.

Deze masterproef is een gevalsanalyse. Het bekijken van een video kan te passief zijn. De kijker wordt niet geprikkeld om de video volledig te kijken of hem nog een tweede keer te bekijken. In deze masterthesis worden suggesties gegeven om het kijken interactiever te maken.

In de masterthesis komen drie onderzoeksvragen aan bod. Ze zijn hieronder opgesomd:

- Welke soorten interactie kunnen we toevoegen aan een video om deze video interessanter te maken?
- Welke soorten interacties zijn er allemaal mogelijk tussen een video en een virtuele 3D-wereld?
- Hoe kunnen we overlay-elementen op een zinvolle imersieve manier toevoegen aan een omnidirectionele video?

Deze onderzoeksvragen worden op het einde van de masterproef beantwoord. De thesis bestaat uit zes hoofdstukken waarin vier verschillende onderzoeken wat betreft interactieve video besproken worden.

Het eerste hoofdstuk bestaat uit een literatuurstudie rond interactieve video. Hiervoor heb ik verschillende wetenschappelijke papers geraadpleegd betreffende interactieve video en hypervideo

In het tweede hoofdstuk bespreek ik de zelfgemaakte HTML-videoplayer. Deze videoplayer heeft de eigenschap dat hij in alle browsers dezelfde functionaliteit en hetzelfde uiterlijk aanbiedt. Dat is iets wat het HTML5 video object niet doet.

In het derde hoofdstuk geef ik een overzicht van de eerste testen die ik ontworpen heb rond interactieve video in combinatie met een virtuele 3D-wereld. Vervolgens bespreek ik de use case die ik gemaakt heb rond de combinatie van interactieve video en een virtuele 3D-wereld

Het vierde hoofdstuk gaat over de combinatie van object tracking en interactieve video. Het doel van dit onderdeel is het maken van een systeem waarbij overlay-elementen kunnen gedefinieerd worden aan de hand van object tracking data. Het hoofdstuk begint met een literatuurstudie over object

tracking programma's. Het gaat hier over programma's waarbij een gebruiker een object in een video kan aanduiden waarna het programma het object trackt doorheen de video. Vervolgens geef ik een beschrijving van het framework van dr. Maarten Wijnants. Hij heeft een systeem ontworpen waarmee overlay-elementen kunnen gedefinieerd worden en weergegeven op een HTML5 video object. Hierna volgen mijn uitbreidingen aan dit framework zodat overlay-elementen met motion tracking data worden ondersteund. Tot slot is er een evaluatie van het gemaakte systeem en mogelijke uitbreidingen en restricties van dit systeem.

Het vijfde hoofdstuk handelt over het bijhouden van acties die een gebruiker uitvoert op een site. Deze acties gaan we bijhouden op verschillende plaatsten zodat later de resultaten kunnen geëvalueerd worden en er mogelijke conclusies kunnen gemaakt worden. Het hoofdstuk begint met een literatuurstudie over web analytic programma's. Dit zijn programma's waarmee de eigenaar van een website informatie over de bezoekers kan opvragen. Daarna volgt er een implementatie van zo een web analytics tool in het huidige systeem. Ten slotte geef ik een voorbeeld van het systeem in werking.

Het zesde hoofdstuk gaat over interactieve omnidirectionele video. Hierbij worden er overlay-elementen geplaatst op een omnidirectionele video. In dit deel beschrijf ik de implementatie van de omnidirectionele videoplayer. Vervolgens bespreek ik de aanpassingen die ik heb gemaakt aan het framework om omnidirectionele video te ondersteunen. Tot slot toon ik de resultaten en bespreek ik een use case.

In het besluit worden de gestelde onderzoeksvragen beantwoord.

1 Gerelateerd werk

Interactieve video kan voor verschillende doeleinden gebruikt worden. Het hoofddoel van interactieve video is het toevoegen van extra informatie aan een video. Vaak kan de gebruiker deze extra informatie zien door te klikken op een regio in de video waardoor de extra informatie verschijnt of waardoor de gebruiker wordt omgeleid naar een plaats waar de extra informatie staat. Hierdoor wordt interactieve video ook vaak hypervideo genoemd. Eén van de toepassingen waarvoor interactieve video gebruikt kan worden is niet-lineaire video. Het verschil met een niet-interactieve video is dat bij niet-lineaire video de gebruiker acties kan uitvoeren op de video en deze acties resulteren in een verandering in het verhaal van de video of die bepalen welke video's er nog getoond zullen worden. Hieronder volgt een bespreking van interactieve videosystemen. Eerst bespreek ik systemen die interactieve video toelaten. Vervolgens komen systemen aan bod die voornamelijk gericht zijn op niet-lineaire video.

1.1 Algemene interactieve videosystemen

In het werk van Bove [1] worden interactieve videosystemen opgedeeld in vijf architecturen. Deze architecturen zijn:

- Stand-alone, local video storage: er wordt bijvoorbeeld een interactieve video getoond op een touchscreen in een winkel die klanten kan helpen.
- Stand-alone, local video storage with back channel: het bovenstaand systeem wordt bijvoorbeeld gebruikt door klanten om informatie op te vragen die is opgeslagen op een server.
- Broadcast video: dit is een uitgezonden programma waarbij men over bepaalde objecten in de video meer informatie kan opvragen.
- Broadcast video with back channel streaming: dit is een uitgezonden programma waarbij men over bepaalde objecten in de video meer informatie kan opvragen. De uitzenders van het programma kunnen bijvoorbeeld via de backchannel kijken hoeveel mensen er over een bepaald object meer informatie hebben opgevraagd.
- Video-on-demand with back channel: een systeem waarbij de gebruikers zelf kunnen bepalen welke interactieve video ze zien. De backchannel kan bijvoorbeeld gebruikt worden om gebruikers meer acties aan te bieden na de video of om informatie over acties op te slaan.

In zijn paper stelt Bove [1] naast zijn object segmentatie systeem voor video's ook een demo voor waarbij gebruikers in een soapaflevering kledingstukken kunnen aanklikken. Na het bekijken van de video wordt het back channel gebruikt om de gebruiker een aantal acties aan te bieden. Zo kan de gebruiker bijvoorbeeld doorverwezen worden naar de website van een winkel waar deze producten verkocht worden.

Een ander systeem waarbij gebruikers meta-data kunnen opstellen voor objecten in een video wordt beschreven door Guang-Ho Cha[2]. In dit systeem wordt een video opgesplitst in camerashots. Gebruikers duiden in elke shot objecten aan waaraan ze extra informatie willen toevoegen aan de hand van een boundingbox. De informatie die ze aan de objecten toevoegen worden in xml-formaat bewaard. Deze xml bevat ook de spatiotemporele eigenschappen van het object. Aangezien de metadata van een object in meerdere video's kan voorkomen, stelt het systeem de gebruiker in staat om de metadata aan meerdere objecten te koppelen. De interactieve videoplayer bestaat uit een

videoplayer en een “user interaction kernel”. De user interaction kernel krijgt de metadata binnen en gaat het positioneren en tonen van de metadata afhandelen. In de video zijn de klikbare delen niet zichtbaar, maar wanneer een kijker met zijn muis over een klikbaar deel beweegt dan verandert de muis van cursor. Voor deze masterthesis willen we een systeem creëren waarbij we ook overlay-elementen aan kijkers kunnen laten zien. Dit is echter niet mogelijk met het voorgesteld systeem van Guang-Ho Cha.

In het werk van Sadallah et al. [3] wordt het Component-based Hypervideo Model (CHM) uitgelegd. Dit high level abstract model is component-based en anotation-driven. Aangezien het anotation-driven is, is er een duidelijke scheiding tussen de video content en metadata versus de mogelijke visualisaties. Sadallah et al. bespreekt ook een implementatie van dit model, genaamd WebCHM. Dit werd gemaakt met standaard HTML-functionaliteiten.

Een JavaScript library die de video op een site toelaat om elementen te besturen is Popcorn.js [4]. Het omgekeerde is ook mogelijk. De library biedt bovendien enkele demo’s aan zoals het weergeven van het adres van locaties getoond in een video, weergegeven in een Google Maps map. Het gaat hier bijvoorbeeld om een film waar een reporter naast het Atomium staat. Naast de video zien gebruikers een Google Maps map met de locatie van die plaats. Een andere demo toont de Twitter tweets van de persoon besproken in een video. De library kan overweg met verschillende soorten videoplayers, waaronder de HTML5 video, YouTube, Vimeo, enz. Popcorn.js biedt echter niet rechtstreeks een systeem aan om overlay-elementen op een video te plaatsen. Dat is wel mogelijk met het systeem dat voorgesteld wordt in deze thesis.

Tot slot zijn er op het internet enkele systemen te vinden die interactieve video aanbieden voor de alledaagse surfer. Zo biedt bijvoorbeeld YouTube een annotatiesysteem[5] aan waarin gebruikers overlay-elementen kunnen plaatsen op hun video’s. De gebruiker heeft hier de keuze tussen vijf soorten uiterlijk wat betreft de annotaties. Deze overlay-elementen kunnen ook een link zijn naar een bepaald moment in een andere video. De overlay-elementen zijn echter statisch en kunnen niet bewegen over de video heen. Deze overlay-elementen kunnen alleen tekst bevatten. In het systeem voorgesteld in deze thesis heeft de gebruiker complete vrijheid over de inhoud van een overlay-element.

Een ander systeem is WireWax [6]. Met dit systeem kunnen gebruikers een video uploaden naar de site van WireWax of een YouTubevideo linken en deze interactief maken. In de editor kan men gebieden in de video aanduiden die automatisch getrackt worden voor de gebruiker en omgezet worden naar overlay-elementen. Het is ook mogelijk om statische overlay-elementen te maken. De gebruiker krijgt de optie de overlay-elementen aan te passen wat betreft de grootte, de kleur en de inhoud (afbeeldingen, tekst of widget). De gebruiker heeft echter geen controle over de motiontracking: dat gebeurt automatisch voor hem. Hoewel dit een voordeel kan zijn voor de alledaagse gebruiker is dit een nadeel voor personen die meer controle over het maken van hun video willen. Ook biedt wireWax niet de mogelijkheid om interacties uit te voeren tussen de video en de site waarop de video staat. Dat is iets wat met het systeem in deze thesis wel kan.

1.2 Specifieke toepassingen

Live televisie

Een specifieke toepassing van interactieve video vinden we in het werk van Hoffmann et al. [7]. Daar wordt een systeem beschreven om een live programma interactief te maken. Het gaat hier echter alleen over interactieve video op computers. In het systeem worden hyperlinks in de video opgeslagen in een RSS feed. Een menselijke editor kijkt naar het live programma en activeert of deactiveert handmatig overlay-elementen die op het scherm worden getoond, zoals te zien is in "Figuur 1.1". Hoffmann stelt ook een onderverdeling voor hypervideo's voor. Hij maakt hier het onderscheid tussen hypervideo's met een open en een gesloten structuur. Bij een gesloten structuur is er slechts één video waaraan alle extra informatie gekoppeld is. Bij een open structuur kan een hypervideo hyperlinks bevatten naar andere hypervideo's.

Figuur 1.5: HyLive video

Niet-lineaire video

Niet-lineaire video is een video waarbij de interacties van de gebruiker op de video bepalen welke delen van de video de gebruiker te zien krijgt. Een voorbeeld hiervan is een video waarbij de protagonist een keuze moet maken tussen naar links gaan of naar rechts. De gebruiker maakt deze keuze en de gepaste video wordt dan afgespeeld met de nodige plotveranderingen.

Voordelen

Eén van de grootste voordelen van niet-lineaire video is dat dit het herbekijken van de video promoot. Als een kijker een keuze moet maken tussen twee paden die het hoofdpersonage kan bewandelen dan zal de kijker vaak ook willen weten wat er zou gebeuren bij de andere keuze. Dit zal de kijker er toe aansporen de video een tweede keer te bekijken en dan de andere optie te kiezen.

Een ander voordeel is dat de gebruiker zelf de controle heeft over het verhaal en het verhaal kan sturen naar zijn smaak. Een voorbeeld hiervan is de romantische film. Vaak moet het hoofdpersonage een keuze maken tussen twee mogelijke partners. Met behulp van niet-lineaire video kan de gebruiker zelf bepalen voor welke partner het hoofdpersonage kiest.

Ook zijn niet-lineaire video's handig voor het maken van tutorials. Ik neem hier het voorbeeld van een tutorial over het assembleren van een kast. De maker van de tutorial kan in de video vragen of de kijker weet hoe hij een schroef correct moet indraaien. Als de gebruiker dit weet klikt hij op 'ja' en gaat de tutorial verder. Als de gebruiker dit niet weet klikt hij op 'nee' en krijgt hij een video te zien over hoe hij dit moet doen. Hierdoor krijgen gebruikers een tutorial die beter op hun noden inspeelt.

Nadelen

Een nadeel van niet-lineaire video is dat het een grotere productie kostprijs heeft dan normale video. Waar men bij een gewone video maar één videopad moet maken, moet men bij niet-lineaire video er meerdere maken. Stel dat een video tien minuten lang duurt. Na vijf minuten zou de gebruiker een keuze kunnen maken tussen twee opties. De video zou dan in totaal vijftien minuten moeten beslaan. Daarom kost het dus veel meer werk om een niet-lineaire film te maken.

Een ander nadeel is dat een deel van het gefilmde materiaal mogelijk niet bekeken zal worden door gebruikers. Het kan zijn dat een gebruiker nooit zal kiezen voor een bepaald pad, waardoor het gefilmde materiaal dat overeen komt met dat pad nooit bekeken zal worden door de gebruiker en dus verloren materiaal is.

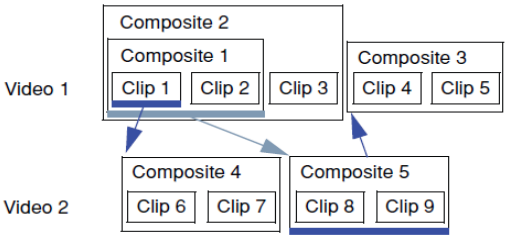
Bestaande technologieën

Er zijn al bestaande technologieën voor niet-lineaire video voor handen. Hieronder geef ik een voorbeeld van drie dergelijke systemen. Het eerste systeem is relatief eenvoudig, de twee daarop volgende zijn complexer.

Eén van de oudste niet-lineaire videosystemen is HyperCafe[8]. Dit is een hypervideo use case waarin gebruikers zich in een café bevinden. Wanneer de gebruikers HyperCafe opstarten dan krijgen ze een video te zien van een café. Hierin ziet de gebruiker de hele scène met daarin verschillende tafels waaraan mensen zitten te praten. De gebruiker kan een tafel selecteren en zal dan de conversatie die de mensen aan die tafel hebben kunnen volgen. Tijdens de conversatie heeft de gebruiker de mogelijkheid om andere conversaties te volgen. Deze mogelijkheden komen op bepaalde momenten tevoorschijn in de vorm van een preview van een andere conversatie. Indien een gebruiker ervoor kiest een andere conversatie te volgen, dan wordt de huidige conversatie gestopt en volgt de gebruiker de nieuwe conversatie. Een voorbeeld hiervan is een conversatie waarin een persoon tegen iemand anders het volgende zegt: “Als ik heel hard ‘Vuur’ roep, zou ik dan de mensen aan die tafel daar storen?”. Bij het uitspreken van deze zin komt er een preview tevoorschijn van de conversatie van de mensen aan de andere tafel. Dit voorbeeld is uitgewerkt in “Figuur 1.2”. Links is de conversatie tussen twee personen te zien die de kijker momenteel aan het volgen is. Rechts in de figuur staat een preview van een andere conversatie. Als de gebruiker op de rechtse video klikt dan gaat hij die conversatie volgen. Tijdens het ontwerpen van deze use case heeft men er op gelet dat de gebruiker het gevoel krijgt dat hij zich in een café bevindt. De video zal bijvoorbeeld nooit op pauze kunnen gezet worden aangezien conversaties in een echte leven ook niet op zo een abrupte manier kunnen worden gepauzeerd.

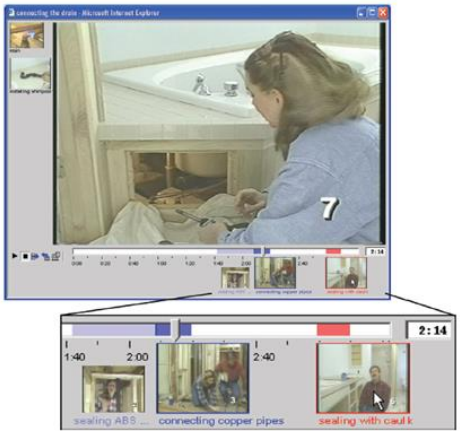
Figuur 1.2: Hypercafe

Een meer uitgebreid systeem van niet-lineaire video wordt beschreven in het werk van Shipman et al. [9]. Hierin stelt men het Hyper-Hitchcock systeem voor. In dit hypervideo systeem kunnen ’clip’s gegroepeerd worden in composits (zie “Figuur 1.3”). Een video bestaat uit een combinatie van composits. De gebruiker kan hyperlinks aanmaken waarmee de kijker naar een andere composit kan gaan. Slechts één hyperlinks kan per tijd zichtbaar zijn. Indien er meerdere zichtbaar zijn, zoals in composite 1 (zie “Figuur 1.3”), dan wordt de meest specifieke gekozen. Wanneer de kijker naar Clip 1 kijkt, zal de donkerblauwe hyperlink getoond worden die



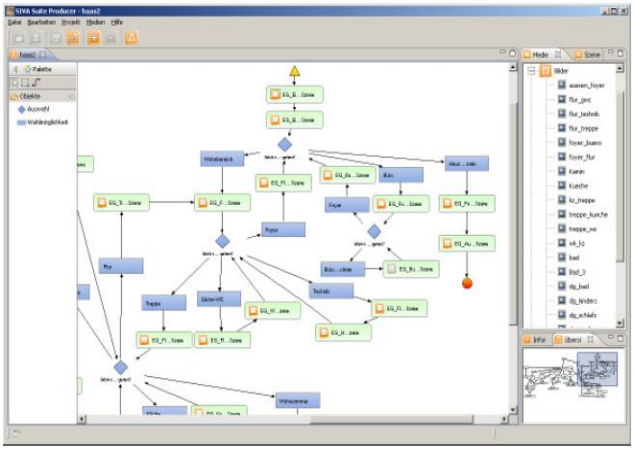
Figuur 1.3: Hyper-Hitcock opbouw video

naar composite 4 verwijst. Als de kijker naar Clip 2 kijkt wordt de lichtblauwe hyperlink naar composite 5 getoond. Voor elke hyperlink kunnen drie eigenschappen ingesteld worden. Ten eerste kan de startpositie in de target video bepaald worden. Vervolgens kan men een tekst label geven aan de hyperlink. Tot slot kan men het terugkeergedrag van de hyperlink instellen. Een voorbeeld is wanneer een gebruiker op een link klikt. Hij kan dan instellen wat er moet gebeuren als de gebruiker de nieuwe video afkijkt, of wat er moet gebeuren als hij de nieuwe video halverwege verlaat. "Figuur 1.4" geeft de player weer. Hierin zijn onderaan in de tijdlijn de hyperlinks te zien. Een hyperlink heeft als visualisatie: een gekleurde sectie op de tijdlijn, een keyframe van de target video en een label. Links in de player (zie "Figuur 1.4") ziet de kijker de geschiedenis van hyperlinks en video's die hij tot nu toe bekeken heeft. Deze zijn klikbaar waardoor de gebruiker naar oudere video's kan terugkeren.



Figuur 1.4: Hyper-Hitchcock systeem player

Een programma dat niet-lineaire video's overzichtelijk maakt is Siva Suite[10]. Hiermee kunnen gebruikers een "scene graph" opstellen waarmee men een duidelijk overzicht kan genereren van welke video's naar elkaar verwijzen. Het systeem laat ook toe om annotaties te tonen naast de video in de vorm van tekst, richtext, ondertitels, afbeeldingen, video's en audiobestanden. Er kunnen ook annotaties op de video zelf geplaatst worden in de vorm van een bounding box.



Figuur 1.5: Siva Suite scene graph

1.3 Motion tracking

In een video blijven de objecten die we interactief willen maken vaak niet op dezelfde plaats staan. Indien ze stil blijven staan in de scène is er nog steeds een grote kans dat de camera beweegt en dus ook de locatie van het object in de video. Dit houdt in dat motion tracking technieken die de positie van objecten in een video kunnen tracken, een belangrijke rol spelen in augmented video. In het werk van Park et al. [11] worden drie manieren geïdentificeerd om een trigger te plaatsen in een video. Zij gebruiken de benaming “een aanklikbaar deel in de video”. De eerste methode bestaat er uit manueel frame-per-frame een bounding box te definiëren voor de trigger. Methode twee is door een objectsegmentatieprogramma te gebruiken om de randen van objecten in video’s te exporteren. Deze techniek is echter moeilijk te gebruiken in de praktijk aangezien er veel veranderende factoren zijn in een video (zoals belichting, camera standpunt, enz.). In de derde methode schetst men de vorm van een trigger. Waar men dus in methode twee een opdeling krijgt van de video en men gebieden kan aanduiden als trigger kan men in de derde methode zelf met de hand aanduiden welk deel van de video de trigger is. Deze trigger wordt getrackt over de scène. Deze laatste methode gebruiken Park et al. in hun werk. Later in deze thesis zullen we een gelijkaardige techniek bespreken waar men een vierhoek kan tekenen in een trackingprogramma, die dan wordt getoond als overlay-element (zie de ‘*outlined trackingpoints’* techniek in sectie 4.3).

In het werk van Sugawara et al. [12] stelt men een alternatieve manier voor om de spatiale en temporele positie van een object in een video waar te nemen. In het object dat men interactief wil maken gaat men infrarood led-lampen verstoppen die niet zichtbaar zijn voor het blote oog. Naast de normale camera om de scène te filmen gaat men ook een infraroodcamera plaatsen. Deze gaat via de positie van de infraroodled-lampen de positie bepalen van het object dat men wil interactief maken. Het nadeel van deze techniek is dat reeds bestaande films niet interactief kunnen gemaakt worden. Ook zal het vaak niet makkelijk zijn om de infraroodled-lampen in een object te verstoppen. Ook heeft men voor deze techniek specifieke apparatuur nodig waardoor de productiekost van de video omhoog gaat.

In deze masterthesis gaan we objecttracking gebruiken om objecten in een scène te detecteren. Welke trackingsoftware hiervoor gebruik wordt kan gelezen worden in sectie 4.1 .

1.4 Virtuele 3D-wereld

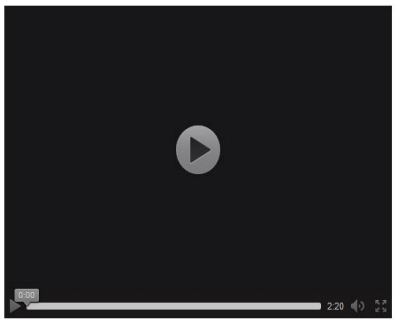
In deze masterthesis ga ik werken met de Unity3D [13]. Unity3D geeft gebruikers de mogelijkheid om op een gemakkelijke manier via een editor virtuele 3D-werelden aan te maken. Door de Unity Web Player kunnen gebruikers deze virtuele 3D-werelden betreden. De Unity Web player laat developers ook toe om te communiceren tussen virtuele 3D-wereld en de website waar de wereld zich op bevindt. Hoe dit gebeurt, wordt in sectie 2.3 uitgelegd. Het nadeel aan Unity Webplayer is dat het een plugin is.

Een andere mogelijkheid voor virtuele 3D-werelden op een website te plaatsen is three.js [14]. Deze JavaScript library Three.js kan samenwerken met het html5 canvas element, SVG [15] en webGL [16]. Virtuele 3D-werelden moeten met three.js via een tekst based editor gemaakt worden. Three.js biedt ook een editor aan, maar deze is nog in beta. Gebruikers moeten geen plugin installeren om een virtuele 3D-wereld gemaakt in Three.js te betreden. Aangezien de three.js code geëmbeded is in de website, is communicatie tussen de 3D-wereld en de site eenvoudig dankzij JavaScript.

2 Videoplayer

Eén van de eerste componenten die ik voor mijn masterproef heb ontwikkeld, is een ‘custom HTML5’-videospeler. De reden hiervoor is dat we een videospeler willen hebben in het masterproefproject die constant blijft over verschillende browsers, zowel op het gebied van uiterlijk als functionaliteit. Het standaard HTML5 video object biedt dat niet aan. “Figuur 2.1”, “Figuur 2.2” en “Figuur 2.3” geven het video object weer in rusttoestand in de verschillende browsers. We zien duidelijk dat in iedere browser de videospeler er verschillend uitziet. Een ander voorbeeld is dat wanneer men in Firefox op een video klikt, de video playback dan start of pauzeert. Dat gebeurt niet in andere browsers, zoals Chrome en IE. Het constant blijven van de videospeler over verschillende browsers, is belangrijk omdat we code willen schrijven die voor elke browser hetzelfde werkt. Bijvoorbeeld als we een interactieve video maken die klikbaar is, dan willen we niet dat we tijdens het ontwerpen van de interactiviteit moeten rekening houden met het feit dat Firefox zijn video pauzeert wanneer we klikken op een object in de video.

Figuur 2.1:HTML5 video in IE



Figuur 2.2:HTML5 video in FireFox

Figuur 2.3:HTML5 video in Chrome

2.1 Opbouw

De zelfgemaakte videoplayer ziet er standaard als volgt uit:

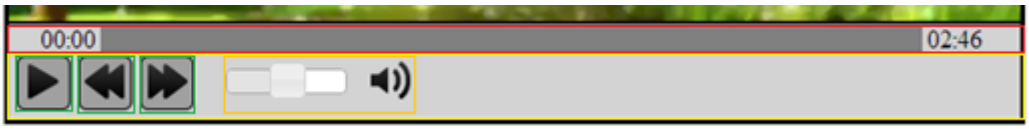
Figuur 2.6:Zelf gemaakte videoplayer

De basis html code ziet er als volgt uit:

```
<div class='overalContainer’>
	<div class='videoContainer'>
		//video element code
	</div>
	<div class='videobarContainer'>
		<div class= ‘topVideobarContainer’>
		//progressbar code
		</div>
		<div class= ‘bottomVideobarContainer’>
		//code voor knoppen en volume slider
		</div>
	</div>
</div>
```

De videospeler is opgedeeld in twee delen, namelijk de ‘*videoContainer*’ en een ‘*videobarContainer*’. In het ‘*videoContainer*’ element bevindt zich een HTML5 video-element. Deze HTML5-videoplayer heeft geen controlbar omdat we deze uitschakelen door de ‘*controls*’ attribute niet mee te geven. We kunnen door middel van JavaScript dit video-element opvragen en er dan functies op uitvoeren.

De videocontrolbar bevindt zich in de *'videobarContainer'* div. Deze div heeft twee child div’s: één voor de progressbar (‘*topVideobarContainer*’) en één die de knoppen groepeert (‘*bottomVideobarContainer*’). Deze twee div’s hebben geen opmaak behalve een grootte. Hierdoor valt het niet op dat de controlbar uit twee delen bestaat.



Figuur 2.7: Div lay-out van controlbar

In bovenstaande afbeelding zien we de verschillende div's. De rode vierhoek is de '*topVideobarContainer'* div*,* de gele vierhoek is de '*bottomVideobarContainer'* div , de groene vierhoeken zijn de div's voor de knoppen en de oranje vierhoek is de div waar de volume slider en mute knop in steken.

Progressbar

De '*topVideobarContainer'* bevat drie div's die naast elkaar aligned worden. Dat gebeurt door middel van float:left. In de meest linkse div zit de timer die de huidige playtime van de video aanduidt. De meest rechtse div bevat de maximum playtime van de video. De middelste div bevat een progresselement; dit element bevat een spanelement. Deze elementen werken samen om een progressbar te genereren. Deze progressbar ziet er echter verschillend uit in verschillende browsers. Daarom gaan we in de standaard css code van de custom player de css van het progresselement en spanelement volledig uitschakelen. Hierdoor gaat de progressbar er in alle browsers, hetzelfde uitzien. De progressbar is klikbaar: wanneer de gebruiker op de progressbar klikt, dan wordt de videoplaytime gezet op het overeenstemmend tijdstip.

Buttonbar

In de '*bottomVideobarContainer'* bevindt zich de HTML code voor de knoppen. Elke knop is een div met een IMG-element erin. Deze knoppen krijgen een onclickevent toegewezen die de gepaste functies zal aanroepen op het video-element. Naast de drie knoppen (play/pause, vertragen en versnellen) staat er ook nog een volumeslider in de buttonbar. Deze is aangemaakt door gebruik te maken van de slider van jQuery. Het icoon naast de slider verandert naargelang het volume. Indien het volume op 0 staat, wordt er een mute icoon getoond. Het icoon is ook klikbaar. Door er op te klikken mute de gebruiker de video, door er weer op te klikken keert de video terug naar het vorige volume.

2.2 Functionaliteit

Integratie

De videoplayercode is zo geschreven dat hij makkelijk te integreren is in een bestaand webproject. Alle code bevindt zich in het bestand VideoPlayer.js. Om een videospeler aan te maken moet een gebruiker enkel een div declareren in zijn HTML-code en de volgende regels code aanroepen:

```
//HTML code

<div id="videoPlayer1" style="float:left;"> </div>

// HTML code

<script src="VideoPlayer.js"></script>
<link rel="stylesheet"
href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
```

```
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
<script>
   //declaratie van de variabelen

   var player = new videoPlayer("videoPlayer1", true);
   player.setSource("Maya.mp4", false, true);

</script>
```

Aan de constructor van de videoplayer moet de gebruiker twee parameters meegeven. De eerste parameter is de naam van de div waar de player in terecht moet komen. De tweede parameter duidt aan of de gebruiker wilt dat de speler een controlbar heeft. De *'setSource'* functie wordt later uitgelegd.

Aanpasbare layout

De gebruiker kan op het videoplayerobject verschillende functies aanroepen om de layout van de speler te beïnvloeden. Bij het instellen van de source van een videoplayer, met de *'setSource'* functie, kan de gebruiker ook als parameter meegeven of hij wilt dat de videoplayer zijn grootte automatisch aanpast aan de grootte van de video. Indien de gebruiker hier false meegeeft, dan behoudt de speler zijn huidige grootte. De gebruiker heeft ook de optie om de grootte van de speler en het video-element handmatig aan te passen. Met de laatste parameter geven gebruikers aan of ze autoplay willen aanzetten.

Naast grootte kan de gebruiker ook via JavaScript zijn eigen css style aan de speler meegeven. De elementen waarvan de gebruiker de css kan veranderen zijn: de algemene div, de div van de videospeler, de div van de controlbar, de div's van de knoppen, de slider en de div's van de afspeeltijden. Naast de div's kan de gebruiker ook de afbeeldingen die op de knoppen staan aanpassen.

Welke functionaliteit de controlbar aanbiedt kan de gebruiker ook zelf kiezen. Zo is er voor elke knop/slider een functie die de knop/slider kan doen verdwijnen van de controlbar. Ook de progressbar kan men doen verdwijnen of men kan ervoor kiezen om de progressbar te tonen maar de klikfunctionaliteit er van uit te schakelen.

Er is ook een '*resetLook*' functie aanwezig. Als deze functie wordt aangeroepen, dan wordt al de css opmaak, afbeeldingen en grootte van de video terug ingesteld op hun standaardwaarden. Als de videoplayer een controlbar heeft, dan worden al de knoppen en functionaliteiten hierop ook terug

enabled.

Figuur 2.8: Videoplayer zonder en met controlbar

Videofunctionaliteit en events

Naast de layout kan de gebruiker ook makkelijk aan al de basisfuncties van het video-element. Voorbeelden van zo functies zijn: play, pause, afspeeltijd instellen, afspeelsnelheid instellen, volume regelen,...

Naast deze functies kan de gebruiker ook enkele eventhandlers koppelen aan het videoplayerobject. Zo kan de gebruiker zijn eigen functie schrijven die aangeroepen zal worden wanneer een video kan afgespeeld worden en wanneer een video gedaan heeft met spelen.

2.3 Implementatie

Hieronder worden enkele implementatie keuzes verder uitgelicht.

Volume slider

De volumeslider is geïmplementeerd met behulp van jQuery. JQuery biedt een eenvoudige manier aan om sliders in een website te implementeren. Aan de slider moet een functie worden meegegeven die wordt afgevuurd telkens als de slider van waarde verandert. In dit geval gaat er gekeken worden naar de waarde van de slider en gaat een overeenkomstige volumewaarde gegeven worden aan de video. Ook gaat de functie kijken of de waarde kleiner of groter dan 0.5 is en dan het volume icoon aanpassen.

Laag volume:  hoog volume:  mute: 

Css opmaak

Voor het veranderen van de css style van bepaalde elementen wordt een eenvoudige functie aangeroepen die de cssText property aanpast van het element:

```
function setOverallDivStyle(style) {
        overalContainer.style.cssText = style;
}
```

Hierbij is echter de slider een uitzondering. Deze slider wordt zoals eerder vermeld aangemaakt door jQuery. De slider heeft een specifieke opmaak. Het is hier moeilijk om bepaalde elementen van de slider op te vragen. Daarom heb ik er voor gekozen om een style object in de HTML-code te verwerken. Indien de gebruiker dan de setSliderStyle functie aanroept, gaat het style object integraal vervangen worden met de css-code die de gebruiker doorgeeft.

```
function setSliderStyle(style) {

        try {
            if (controlBar && _enableVolumeSlider) {
                sliderStyle.innerHTML = style;
            } else {
                throw "Video does not have a controlbar or volumeslider ";
            }
        } catch (err) {
            alert(err);
        }
}
```

De style van de progressbar kan niet worden veranderd. Dit is omdat de style gedisabled wordt om de progressbar er in al de browsers hetzelfde te doen uitzien. Er zijn sites waarop wordt uitgelegd hoe men een progressbar kan stylen[17]. We zien dat de gebruiker zeer ver kan gaan met het stylen van de progressbar maar dat de css-code redelijk complex is en verschillend voor alle browsers.

Voor het implementeren van de '*resetLook*' functie gaan we eerst alle css-code van alle elementen in de videoplayer opslaan in JavaScript variabelen. Het HTML5 video-element zijn grootte wordt niet bepaald door css-opmaak dus deze zullen we ook apart moeten opslaan. Bij het aanroepen van de '*resetLook*' functie wordt de css-code van al de elementen terug ingesteld op de opgeslagen variabelen.

Progressbar

De progressbar wordt verwezelijkt door de volgende HTML-code:

```
<progress class='progressBar' max='100' value='0'><span>0</span>% played</progress>
```

Deze HTML-code wordt gecombineerd met de JavaScript functie '*barUpdate'*. De functie is gekoppeld aan het '*timeupdate'* event van het video-element. Telkens als de video geupdatet wordt (dus de playtime verandert) dan wordt de '*barUpdate'* functie aangeroepen. In deze functie gaan we het percentage berekenen van hoever de video is. De progressbar updaten we door de innerHTML eigenschap van de span in te stellen op het berekend percentage. In de '*barUpdate'* functie wordt ook de currenttime links van de progressbar geupdatet.

Aan de progressbar wordt ook een '*onclick'* event toegevoegd. Wanneer dit event wordt afgevuurd, gaat er gekeken worden naar de clickpositie ten opzichte van de progressbar. Deze plaats wordt omgezet naar een tijdstip in de video. De video wordt dan ingesteld op dit tijdstip.

Enablen/disablen van knoppen

Inserten van HTML-children aan een div tussen de bestaande children van een div is niet mogelijk in HTML. Via de innerHTML-variabelen van een element kunnen we de children bepalen. Maar wanneer we deze variabelen overschrijven, dan overschrijven we alle HTML-code in het HTML-element. Bijvoorbeeld:

```
<div id="test" style="float:left;">A</div>
<script>
	var div = document.getElementById(“test”);

	div.innerHTML= “B”;

</script>
```

Resulteert in praktijk in:

```
<div id="test" style="float:left;">B</div>
```

Daarom heb ik een hertekenfunctie geschreven die de controlbar hertekent naargelang welke knoppen/functionaliteit enabled is. In deze hertekenfunctie gaan we de innerHTML-string opstellen naargelang welke knoppen enabled zijn. We moeten hierbij wel met enkele dingen rekening houden.

Wanneer we een element hertekenen, dan bestaat het oude element niet meer. Dat wil zeggen dat wanneer we een element in een JavaScript variabele hebben gestoken en we hertekenen dit element, dan is de inhoud van de variabele niet langer geldig. We moeten dus na het hertekenen van de elementen al de variabelen die hertekend zijn weer correct instellen. Een ander probleem is dat de custom css-opmaak van de elementen verdwijnt wanneer we ze hertekenen. Om dit tegen te gaan, vragen we de css van al de elementen eerst op en slaan we deze daarna op in variabelen. Na het hertekenen, stellen we voor de hertekende elementen terug de correcte css in. Als laatste probleem moeten we ook weer alle eventlisteners ontkoppelen voor we een element disablen en terug correct koppelen aan elementen die enabled zijn.

3 Combinatie interactieve video en virtuele 3D-omgeving

In dit deel van de masterproef beschrijf ik de testen die ik heb uitgevoerd rond interactieve video en de combinatie met een virtuele 3D-wereld. Eerst is er een literatuurstudie rond interactieve video. Daarna geef ik een overzicht van testen en een use case die ik ontworpen heb.

3.1 Test 1: Canvas overlay

Mijn eerste test rond interactieve video maakt gebruik van een canvas overlay op een video. De bedoeling van de test is om te zien of we het canvas element kunnen gebruiken als een overlay op de video. In deze canvas gaan we overlay-elementen tekenen. In deze test word er nog gebruik gemaakt van een oudere versie van de zelfgemaakte videoplayer besproken in hoofdstuk twee.

In de test krijgt de gebruiker een film te zien waarop drie blauwe overlay-elementen worden getoond (zie “Figuur 3.1”). Deze overlay-elementen zijn getekend in hetzelfde canvaselement. Naargelang er geklikt wordt op een bepaald overlay-element, verschijnt er een pop-up met de tekst: “Geklikt op de zoveelste rechthoek”. Na een aantal seconden verdwijnen de overlay-elementen.

Figuur 3.1: Video met drie overlay-elementen getekend op een canvas

De HTML-code ziet er als volgt uit:

```
<video id="video1" width="640" height="264"
style="background-color:green">
     <source src="test4.mp4" type="video/mp4" >
     <source src="test4.ogv" type="video/ogg" >
     Your browser does not support HTML5 video.
</video>
<canvas id="drawCanvas" style="background-color:transparent; position:absolute">
</canvas>
```

Het canvaselement wordt over het video-element geplaatst. De test heeft een updatefunctie die om de 100 milliseconden afgaat. Deze functie kijkt naar de videotijd en gaat op een bepaald moment in de video de eerste twee overlay-elementen tekenen en iets later het derde. De overlay-elementen krijgen telkens een andere alphawaarde wanneer ze getekend worden. Het eerste overlay-element krijgt de maximale alphawaarde. Het tweede overlay-element krijgt een iets lagere alphawaarde; het derde krijgt een nog lagere alphawaarde. De waarden zijn nog hoog genoeg gehouden zodat de gebruiker het bijna niet kan zien. In “Figuur 3.1” zien we dat het niet opvalt dat de eerste twee overlay-elementen verschillende alphawaarden hebben.

In het ‘onclick’ event van de canvas wordt er gekeken naar de pixel in de canvas waar de gebruiker heeft op geklikt. De waarde van een pixel (x,y) van een canvas wordt opgevraagd aan de hand van de volgende functie:

```
var drawCanvas = document.getElementById('drawCanvas');
var context = drawCanvas.getContext('2d');
var pixel = context.getImageData(x, y, 1, 1);
```

De variabele ‘pixel’ bevat nu een array van vier waarden: Rood, Groen, Blauw, Alpha. De alpha-waarde is een waarde tussen 0 en 255. Aan de hand van de alphawaarde kan hier nu gekeken

worden welk overlay-element er is aangeklikt (aangezien elk element een unieke alphawaarde heeft). Indien de gebruiker op een plaats in het canvas tekent waar niet op getekend is, dan is de alphawaarde 0.

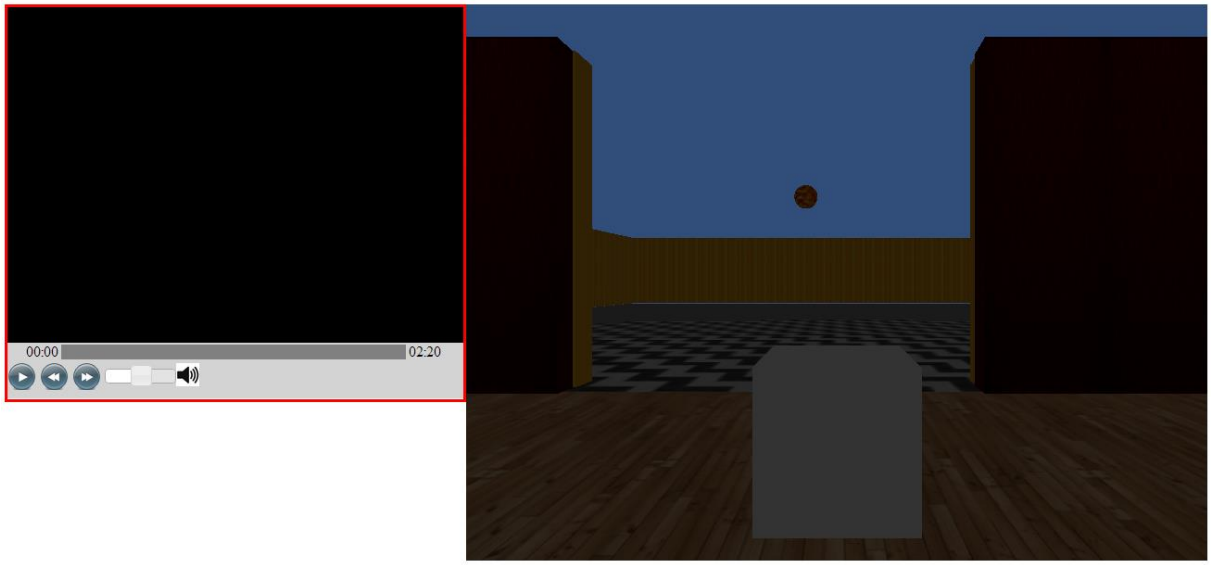
Deze techniek heeft echter enkele nadelen:

- Indien er veel overlay-elementen tegelijk getekend moeten worden, wordt het verschil in de alphawaarden tussen de overlay-elementen zichtbaar. Sommige overlay-elementen zouden doorzichtig worden en dat is onaanvaardbaar.
- Op een canvas kan tekst getekend worden maar de gebieden tussen de letters worden niet opgevuld. Dat betekent dat wanneer een overlay-element alleen uit tekst bestaat de gebruiker op een letter moet klikken voordat het systeem zal herkennen dat er op iets geklikt is geweest.
- Een canvaselement is bedoeld om op te tekenen; het biedt functies aan waarmee gebruikers afbeeldingen, lijnen en zelfs videoframes kunnen tekenen op het canvas. Het is echter niet eenvoudig om complexere elementen mee te ontwerpen. Als men een alinea aan tekst in het canvas wil schrijven, moet men per regel de positie van elke regel aanduiden in het canvas.

Deze nadelen zorgen ervoor dat deze techniek niet geschikt is om overlay-elementen te tekenen op een video.

3.2 Test 2: Combinatie video en Unity webplayer

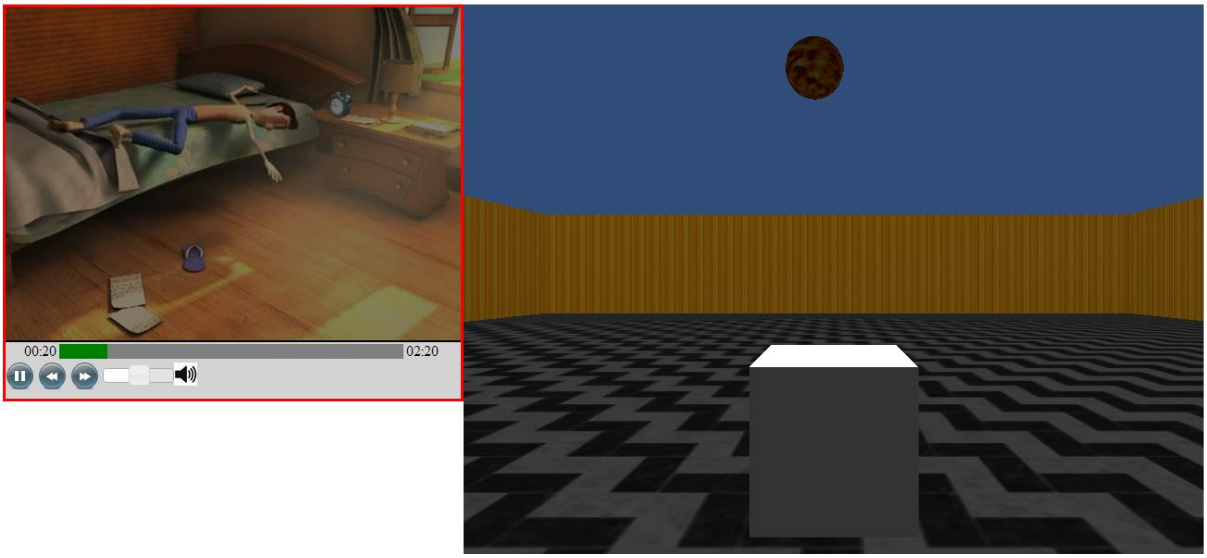
Het doel van deze test was om uit te zoeken hoe de interactiviteit tussen een externe video en een virtuele 3D-wereld in Unity werkt.



Figuur 3.2: Video gecombineert met een virtuele 3D-wereld

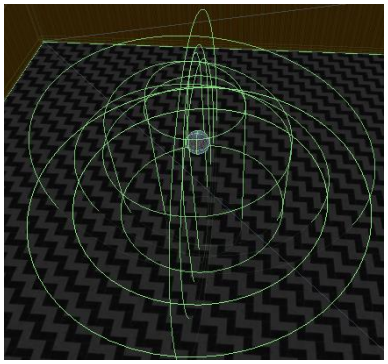
In bovenstaande figuur is de test te zien. Links staat de videoplayer en rechts staat de virtuele 3D-wereld. De gebruiker bestuurt de kubus is de virtuele 3D-wereld. Indien de gebruiker de kubus naar kamer met de bol beweegt, start de video. Zodra de video start wordt er een overlay-element op geplaatst met de techniek besproken in hoofdstuk 3.1. Wanneer een gebruiker hier op klikt gaat de

lamp (de bol in de kamer) aan en wordt de video donker. Naarmate de gebruiker dichter bij de lamp komt gaat hij weer meer te zien krijgen in de video.



Figuur 3.2: Kubus in kamer 2

In bovenstaande afbeelding zien we dat de kubus is de tweede kamer is. De lamp geeft licht op de kubus en de video is donkerder gemaakt omdat de kubus niet volledig onder de lamp staat. Hoe dicht de kubus bij de lamp staat wordt bepaald door enkele triggerboxen die rondom de lamp worden geplaatst. (zie “Figuur 3.3”).



Figuur 3.3:Triggerboxen rond lamp

Uit deze test heb ik geconcludeerd dat het mogelijk is om een interactieve video te koppelen aan een virtuele 3D-wereld. Hoe de communicatie werkt tussen de Unity webplayer en de website wordt besproken in hoofdstuk 3.2. In dat hoofdstuk wordt het concept van de video donker maken aan de hand van een gebeurtenis in de virtuele 3D-wereld uitgelegd.

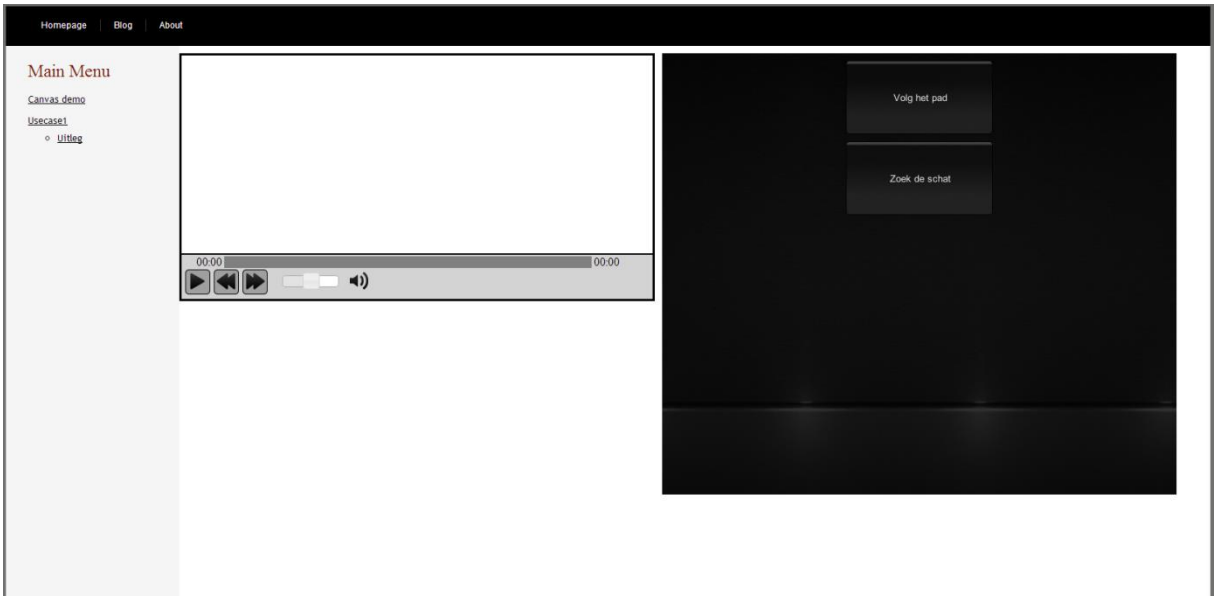
3.2 Use case

Doel van de use case

Het doel van deze use case is om uit te zoeken welke soorten interacties allemaal mogelijk zijn tussen een video en een virtuele wereld en om te onderzoeken hoe we dit in de praktijk zouden kunnen verwezenlijken. In deze use case werken we met Unity3D om de virtuele wereld te maken. De use case bestaat uit twee spelen die beide andere interacties met de video aanbieden. Het eerste spel is een combinatie van een racespel en videoplayback. Het tweede spel is een combinatie van een zoekspel met video playback. De video wordt niet getoond in Unity zelf maar in een externe videoplayer.

Menu

Wanneer de gebruiker naar de pagina van de use case gaat dan krijgt hij een pagina te zien met links de custom made videoplayer en rechts een Unity webplayer. De Unity webplayer is een browserplugin waarmee gebruikers kunnen interageren met virtuele 3D-werelden, gemaakt in Unity. In de Unity 3D-wereld krijgt de gebruiker een menu te zien. Hier heeft hij de optie tussen twee spelen die gekozen kunnen worden. In “Figuur 3.4” zien we links de zelfgemaakte videoplayer en rechts de Unity webplayer. Het menu is zichtbaar in de Unity Webplayer



Figuur 3.4: Use case pagina, menu zichtbaar

Spel1: “Volg het pad”

Doel van het spel

In dit spel speel heeft de speler de rol van ‘Maya de bij’. De bedoeling van het spel is om op het einde van het pad te geraken voordat de tijd op is. Zoals in de meeste hedendaagse racegames, moet de speler voorbij verschillende checkpoints passeren voordat hij de finish kan bereiken**.** Tijdens het spel speelt er een video af van Maya de Bij. Pas wanneer de speler het einde van het pad bereikt, kan hij ongestoord de hele video bekijken. Zolang de speler op het pad blijft en op tijd voorbij de checkpoints komt blijft de video afspelen.

Wanneer een speler voorbij het eerste checkpoint komt, start de video en verschijnen er rechts boven twee tijdsstippen. Deze twee tijden geven weer hoeveel tijd de speler heeft om het volgende checkpoint te bereiken, en hoelang hij tot nu toe al bezig is met het spel. Telkens wanneer een speler voorbij een checkpoint komt, krijgt hij een instelbaar aantal seconden extra om bij het volgende checkpoint te geraken. Indien een speler niet tijdig het volgende checkpoint bereikt, dan wordt hij teruggezet op het laatste checkpoint waar hij voorbij kwam. Hij krijgt dan het ingesteld aantal seconden om bij het volgende checkpoint te geraken. Tegelijkertijd wordt de video teruggespoeld naar het moment dat de speler voorbij dat checkpoint kwam. Zo dwingt het spel dus de speler om voorbij al de checkpoints te gaan. In onderstaande figuur zien we rechts de Unity omgeving waarin de gebruiker de bij bestuurt. Het pad dat de gebruiker moet volgen, is grijs. Wanneer de gebruiker zich op het gras bevindt dan zit hij niet meer op het pad. De twee kegels aan weerskanten van het pad zijn het eerste checkpoint. De video die links , zal starten als de bij hier voorbijkomt.

Figuur 3.5: Volg het pad spel

Een speler mag niet van het pad afwijken. Indien een speler dat toch doet, krijgt hij een instelbaar aantal seconden om weer op het pad te geraken. Indien de speler dat niet doet, is het gameover en krijgt hij de keuze het spel te herstarten of terug naar het menuscherm te gaan. De video zal ook een aanwijzing geven over hoe lang de speler nog heeft om terug op het pad te geraken. Wanneer een speler van het pad afwijkt, wordt de video geleidelijk aan donkerder. Zodra de video helemaal zwart geworden is, weet de speler dat het game over is.

Interacties tussen video en virtuele wereld en praktische uitwerking

Tijdens het spel vinden er verschillende interacties plaats tussen de video en de virtuele wereld. Dat is mogelijk doordat de Unity webplayer kan communiceren met de webpagina waarin hij zich bevindt[18]. De communicatie tussen Unity en de webpagina gebeurt als volgt:

Application.ExternalCall("functie1", "variabele1",...);

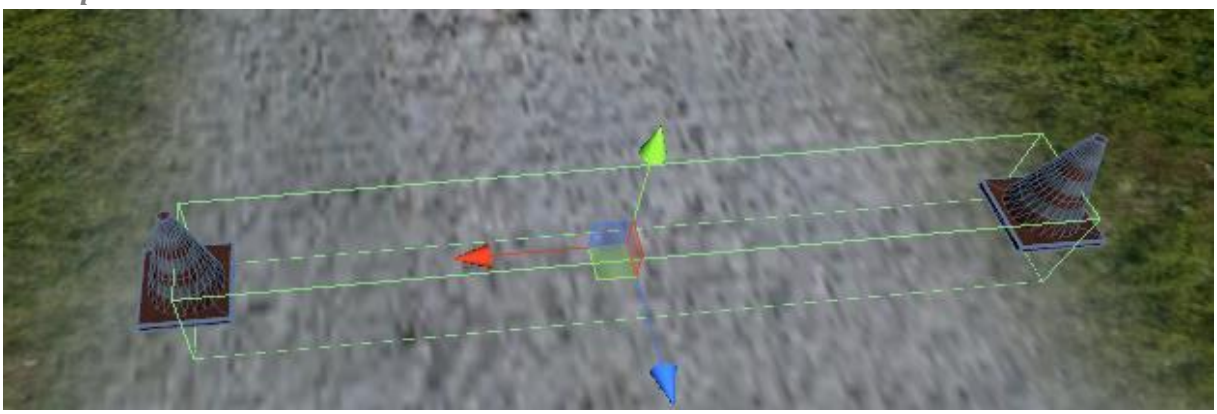
Deze regel code roept een JavaScript functie aan met de naam “functie1” en passeert een aantal parameters aan deze JavaScript functie. In deze use case is er uitsluitend éénrichtingsverkeer van Unity naar de webpagina. Al de gamelogica bevindt zich in Unity en het spel zal dus bepalen wat de speler ziet van de video en hoe de video eruitziet.

Hieronder licht ik toe hoe ik bepaalde aspecten van het spel en de communicatie met de video heb aangepakt.

Opstarten en eindigen spel

Wanneer een speler in het menu het spel “Volg het pad” selecteert, dan wordt er al meteen een functie in de webpagina aangeroepen die de webpagina laat weten welk spel er gestart wordt. Zo weet de webpagina ook welk filmpje er geladen moet worden. Als het spel voorbij is, dan krijgt de speler opnieuw een menu te zien. Kiest hij hier om nog eens te spelen, dan wordt het spel opnieuw geladen. Kiest hij ervoor om terug te gaan naar het hoofdmenu, dan laat Unity de webpagina weten dat de video verwijderd moet worden.

Checkpoints

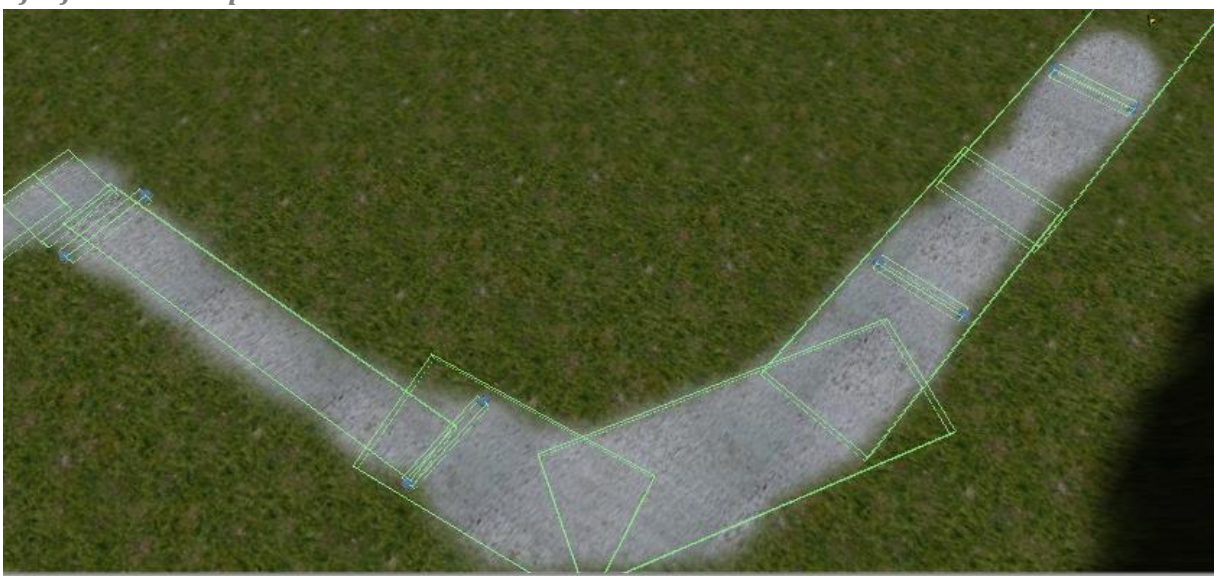


Figuur 3.6: Spel 1 checkpoint opbouw

Op het pad in Unity staan checkpoints. Dat zijn eenvoudige triggerboxen (de groene balk in “Figuur 3.6”) met aan weerskanten een verkeerskegel. Indien de speler hier voorbij komt, krijgt hij extra tijd. Hoeveel tijd de speler heeft, wordt bijgehouden in Unity. Als de speler voorbij een checkpoint komt, wordt er ook een functie aangeroepen op de webpagina. Die functie gaat kijken wat de afspeeltijd van de video op dat moment is. Deze afspeeltijd gaat de webpagina bijhouden totdat hij het signaal krijgt van Unity dat hij een andere afspeeltijd moet bewaren.

Indien een speler niet op tijd bij het volgende checkpoint is, zal Unity weer een functie aanroepen in de webpagina. Deze functie zal dan de video herzetten naar het laatst opgeslagen tijdstip.

Afwijken van het pad



Figuur 3.7: Spel 1 pad opbouw

Het pad in Unity is gemaakt uit een reeks triggerboxen (de grote vlakke balken in “Figuur 3.7”) die mekaar gedeeltelijk overlappen. Wanneer de speler met de avatar van het pad afwijkt en dus zich in geen enkele triggerbox meer bevindt, weet het spel dat de speler van het pad is afgegaan. Unity gaat dan een timer starten die na het ingesteld aantal seconden afloopt. Als de speler binnen die tijd niet terug binnen één van de triggerboxen is, dan is het spel afgelopen. Tegelijkertijd met het starten van de timer, gaat Unity ook een functie aanroepen in de webpagina. Deze functie gaat in de webpagina zelf ook een timer van een instelbaar aantal seconden starten. Deze timer gaat om de 100 milliseconden de video een beetje donkerder maken zodat hij na na het ingesteld aantal seconden helemaal zwart is.

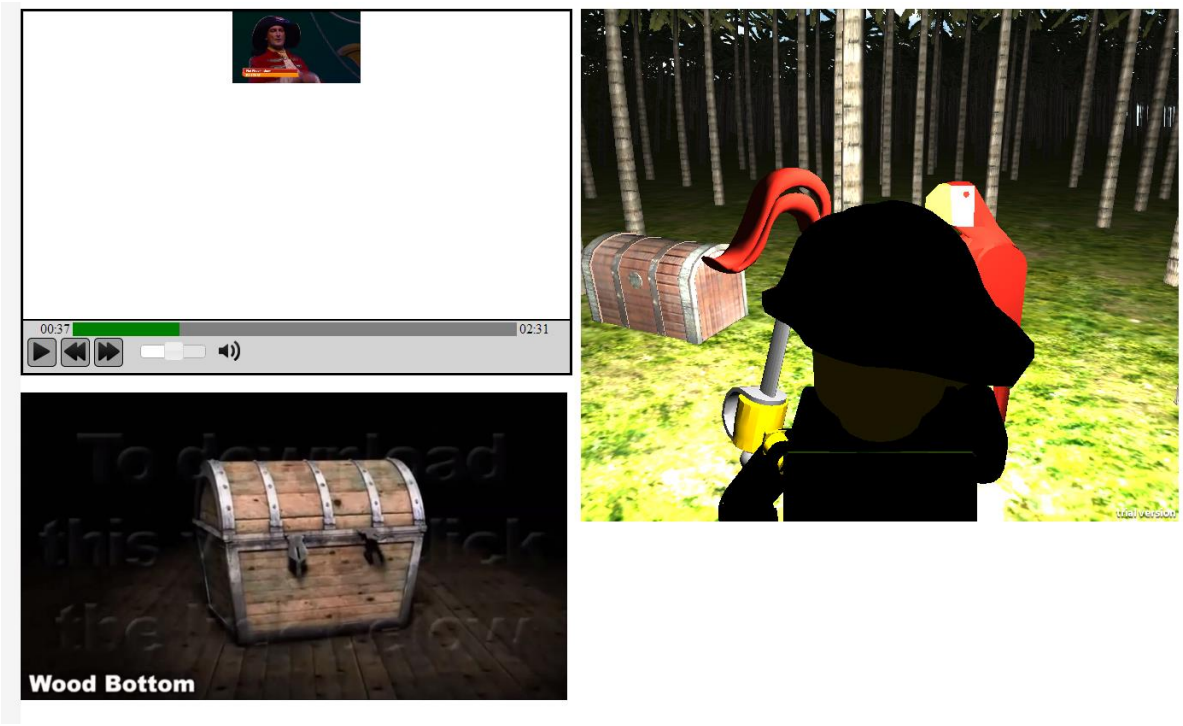
Indien de speler erin slaagt terug te keren naar het pad voordat de tijdlimiet verstreken is, gaat Unity de timer stoppen en de webpagina laten weten dat hij zijn timer ook kan stoppen. De webpagina gaat dan de video terug volledig zichtbaar maken. Indien de speler niet binnen de twee seconden terug op het pad is, gaat er in het spel een gameoverbericht komen en gaat Unity de webpagina laten weten dat het spel voorbij is. De wepagina gaat dan de video stoppen en terugspoelen naar het begin.

Het donker maken van de video gebeurt door een canvasoverlay op de video te plaatsen. Deze canvasoverlay is een HTML5-canvaselement dat we positioneren op het video object. Dit doen we door de left- en topoffset van de video op te vragen en in te stellen als de offset van het canvaselement. Op deze canvasoverlay een volledig zwarte afbeelding getekend. Elke 100 milliseconden wordt de alphawaarde van deze afbeelding aangepast. Zo zal er dus eerst een “doorzichtige” zwarte afbeelding over de video staan, en na het tijdlimiet verstreken is een totaal ondoorzichtige zwarte afbeelding.

Spel 2: "Zoek de schat"

Doel van het spel

In dit spel neemt de speler de rol aan van Piet Piraat. Hij gaat op zoek naar de schat van zijn grootvader. De speler kan de schat vinden door te kijken naar de grootte van de video en te luisteren naar het volume. Des te dichter de speler zijn avatar beweegt naar de schat toe, des te groter en luider de video wordt. De speler moet over de schat heen lopen om hem te openen. Als de speler over een schatkist beweegt, dan speelt er een tweede video af die de speler laat weten of de schatkist een schat bevat of leeg is. Als een schatkist leeg is, dan wordt de video terug klein en stil en moet de speler weer op zoek gaan. De volgorde van de schatten die de speler moet vinden is vast bepaald door het spel.



Figuur 3.8: Zoek-de-schatspel

In bovenstaande afbeelding zien we het zoek-de-schatspel. We zien dat de video in de player (links bovenaan) klein is. Dit betekent dat de volgende schat ver weg is. Rechts zien we de virtuele wereld waar de gebruiker de schatten in moet zoeken. Er staat een schat voor hem waar hij juist over is gewandeld. Links onderaan wordt er een tweede video getoond waaruit de gebruiker kan afleiden dat de schat leeg was.

Interacties tussen video en virtuele wereld en praktische uitwerking

Opstarten en eindigen spel

Het opstarten en beëindigen van het spel gebeurt analoog aan het eerste spel "Volg het pad".

Aanpassen grootte video

In de updatelus van het spelerobject in Unity wordt er om de 100 milliseconden een functie aangeroepen van de webpagina. In Unity wordt er berekend hoe ver de speler verwijderd is van de volgende schatkist. Deze afstand wordt omgerekend naar een percentage. Dat percentage specificeert hoe groot/luid de video moet zijn ten op zichte van zijn maximale grootte/volume. De

minimum grootte/volume is 20% zodat de speler altijd iets kan zien en horen van de video. Indien de speler een lege schatkist heeft gevonden, wordt het percentage opnieuw berekend ten opzichte van de nieuwe schatkist die de speler moet zoeken.

Schatkist openen

Als een speler een schatkist vindt, dan gaat Unity kijken of dit de juiste schatkist is. Unity gaat dan de gepaste functie aanroepen op de webpagina. Indien het een lege schatkist is, gaat de webpagina functie een extra filmpje tonen van een lege schatkist. Indien het de echte schat is, gaat de webpagina functie een filmpje tonen van een volle schatkist.

Conclusie van de use case

Wanneer we een combinatie van virtuele 3D-wereld en video maken, dan kunnen we best de 'game logica' op één plaats bewaren. In deze use case hebben we de logica geplaatst in Unity omdat hier sowieso al game logica steekt. Een voorbeeld hiervan is de detectie of de speler zich op het pad bevindt. Het Unity project zal hier events uitsturen naar de webpagina met acties die moeten worden uitgevoerd op de video. Wanneer men dus een combinatie wil maken van een virtuele 3D-wereld met een externe videoplayer, dan kan men best eerst kiezen waar men de logica gaat steken. De logica wordt het best op de plaats gezet waar de gebruiker het vaakst mee gaat interageren. Dit betekent dat als men een use case wil ontwerpen met een interactieve video die ondersteund wordt door een simpele virtuele 3D-wereld, men dus best de logica in JavaScript op de site plaatst.

Interacties tussen een video en een virtuele 3D-wereld zijn perfect mogelijk met de technologie die momenteel beschikbaar is voor PC's. Wanneer men echter de use case uitprobeert, zal men merken dat het niet mogelijk is om volledig geconcentreerd te zijn op beide elementen, namelijk de video en de 3D-wereld. Voor mobiele apparaten zoals tablets zullen deze use cases echter niet werken. Tot nu toe ondersteunen deze nog niet de webplayer van Unity.

4 Motiontracking data gebruiken voor overlay-elementen

In dit deel van de masterthesis ga ik verder werken op de codebase gegeven door dr. Maarten Wijnants. Het doel van dit deel van de masterthesis is om een systeem te ontwikkelen waarmee overlay-elementen kunnen bewegen via motion/object tracking data.

4.1 Literatuurstudie object tracking

Wanneer we objecten in een video willen klikbaar maken, dan moeten we hierbij een onderscheid maken tussen twee soorten objecten, namelijk statische en dynamische objecten. Indien we software hebben die ons toestaat objecten klikbaar te maken in een video, dan is het eenvoudig dit toe te passen op statische objecten. De ontwerper moet enkel weten waar, wanneer en hoe groot het object in de video is. Voor dynamische objecten is het minder eenvoudig; hierbij kan het object wat de ontwerper klikbaar wil maken elke frame op een andere positie staan. Het is niet praktisch om de ontwerper voor elke frame te laten definiëren waar het object staat. Daarom gaan we gebruik maken van trackingsoftware die een object of regio doorheen een filmscène trackt en die hiervan de positionele data weergeeft. Deze tracking data zou dan gebruikt kunnen worden door de interactieve videosoftware om de overlay-elementen in elke frame op de juiste plaats te zetten.

Het onderwerp van objecttracking in video's is zeer ruim. In dit deel van de literatuurstudie ga ik proberen een overzicht te geven van welke soorten motion/object tracking algoritmes en software er bestaan en waarom ze al dan niet toepasbaar zijn in deze masterthesis.

Bestaande algoritmes en software

OpenCV (Open Source Computer Vision Library)

OpenCV is een open source library die verschillende computervisie-algoritmes aanbiedt [19]. Ontwikkelaars kunnen deze library gebruiken om zelf hun eigen applicaties te schrijven of de OpenCV algoritmes te integreren in bestaande applicaties. Men kan ook verschillende tutorials vinden van onder andere objecttracking en vormtracking [20].

Het voordeel van gebruik te maken van OpenCV is dat de ontwikkelaar zelf kiest welk algoritme hij implementeert. Hij moet ook niet alles van nul af aan maken aangezien de library veel functionaliteit aanbiedt. Bovendien is deze open source waardoor een ontwikkelaar zelf kan kijken hoe deze functies werken en of ze exact doen wat hij wil dat ze doen.

Een nadeel van het gebruiken van OpenCV is dat een ontwikkelaar voldoende kennis moet hebben van computervisietechnieken en videoanalyse. Ook neemt het implementeren van een trackingprogramma veel tijd in beslag. In deze thesis ligt de nadruk niet op motion/object tracking en willen we eerder een programma gebruiken dat ons deze functionaliteit aanbiedt. We zijn meer geïnteresseerd in welke resultaten er uit komen en hoe we deze kunnen laten interpreteren door onze software.

OpenTLD (predator)

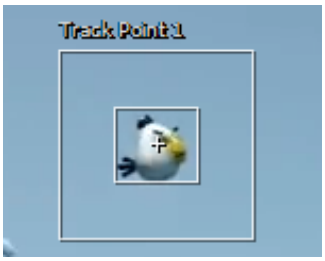
Dit programma is gemaakt door dr. Zdenek Kalal [21]. Met dit programma is het mogelijk een regio aan te duiden in een video en deze te tracken.

Het programma maakt gebruik van een leermodel [22]. De software gaat leren wat een object, namelijk hetgene dat we willen tracken, is en wat geen object is. Het programma gaat rekening houden met hoe het object eruitziet in verschillende groottes en rotaties. Het gaat deze informatie gebruiken om elke frame af te gaan op zoek naar het object. Doordat de software steeds bijleert, gaat de tracking steeds beter worden. Deze software is open source en is dus aanpasbaar door ervaren ontwikkelaars. OpenTLD maakt gebruik van OpenCV.

OpenTLD is ontwikkeld als een doctoraatsstudie [23] en is daardoor niet ontworpen voor commerciële doeleinden. Er is een commerciële versie van het programma maar hier moet men dus voor betalen.

Adobe After Effects (AE)

Adobe After Effects is een grafische tool voor video editing [24]. Het stelt gebruikers in staat om video's aan te passen op verschillende manieren. Eén van de tools die ze ter beschikking stellen is een trackingmechanisme. Dit trackingmechanisme stelt de gebruiker in staat een punt aan te duiden in de video op een bepaald frame. Dit punt stelt de trackcoördinaten voor die worden teruggegeven. Rondom het punt staan twee vierkanten: het binnenste vierkant stelt het gebied voor dat we willen tracken. Het buitenste vierkant stelt het gebied voor waarin er gezocht wordt naar de regio die we willen tracken. Dit vierkant beweegt mee met de getrackte regio.



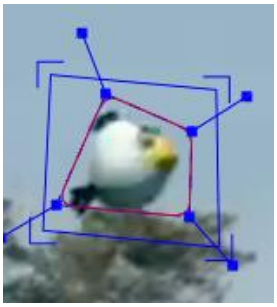
Figuur 4.1: Tracking in Adobe After Effects

Aangezien AE geen open source programma is, kunnen we niet achterhalen welk algoritme AE gebruikt om aan tracking te doen. Het biedt wel een uitgebreide interface aan die gebruikers in staat stelt makkelijk naar de exacte frame te gaan waar men wil beginnen met tracken. AE biedt wel geen mogelijkheid om zijn trackdata makkelijk te exporteren. De gebruiker moet een omweg maken door zelf een bestand aan te maken, de data te kopiëren uit het programma en het dan te plakken in het bestand. Hierdoor is er een kans dat de gebruiker fouten introduceert in de trackdata.

Mocha AE

Mocha AE is een programma dat zich specialiseert in videotracking en rotoscoping (het loskoppelen van een object in een video van de achtergrond zodat een andere achtergrond kan geplaatst worden) [25]. Deze versie van Mocha komt samen met Adobe After Effects. Gebruikers kunnen dan ook makkelijk een object tracken in Mocha AE en de data doorgeven aan AE. Mocha AE voorziet ook de mogelijkheid om trackingdata te exporteren in txt-formaat. Er zijn 3 mogelijkheden om motiontrack data te exporteren:

- Corner pin only
- Corner pin + motion blur data
- Transform data



Figuur 4.2: Tracking in Mocha AE

In Mocha AE kan de gebruiker eerst een veelhoek tekenen rondom het object dat hij wil tracken. Het programma genereert vervolgens vierhoek. De hoekpunten van deze vierhoek zijn de corner pins waarvan Mocha AE de positie per frame gaat exporteren. In “Figuur 4.2: Tracking in Mocha AE” zien we de blauwe punten, met rood verbonden. Deze zijn geplaatst door de gebruiker. De hoekpunten van de blauwe vierhoek zijn de corner pins.

Deze export mogelijkheden zijn specifiek ontworpen voor After Effects en zijn dus niet bedoeld om door andere programma’s ingeladen te worden. Mocha pro biedt echter meerdere export data mogelijkheden. Mocha pro is echter een duur programma waardoor we dit programma niet zullen kunnen gebruiken in de masterthesis.

Keuze

Het tracken van objecten gaat gebeuren met Mocha AE omwille van de volgende redenen:

- Aangezien ik Adobe CS6 ter beschikking heb, heb ik ook Mocha AE ter beschikking.
- Het tracken van objecten in Mocha AE is gedetaileerder en nauwkeuriger dan Adobe After Effects.
- Mocha AE biedt een eenvoudige interface aan om objecten te tracken in een video
- Mocha pro heeft meer mogelijkheden om data te exporteren maar is helaas te duur om aan te schaffen.

4.2 Het framework

In deze masterthesis heb ik verder gewerkt op de code van dr. Maarten Wijnants. In deze sectie van de masterthesis geef ik een beschrijving van de gekregen code en hoe ze werkt. In dit framework gaan er overlay-elementen geplaatst worden op een video door div elementen te gebruiken. Ieder overlay-element is een div element. De gebruiker heeft de mogelijkheid de inhoud, opmaak, positie en tijdstip te bepalen van dit overlay-element. Hieronder wordt beschreven wat de klassen/bestanden in het systeem doen.

wnggav_types

In dit bestand staan verschillende types gedefinieerd, waaronder het '*Position2D*' type, een type dat een 2D-punt voorstelt, en het type '*CustomEventTypes*_t' dat bijhoudt welke soorten custom events er zijn in het systeem.

VideoOverlayelement

Deze klasse is de definitie van het overlay-element. De klasse houdt de volgende waarden bij:

- Starttijd van het overlay-element (tijdstip in de video)
- Stoptijd van het overlay-element (tijdstip in de video)
- Startpositie van het overlay-element (2D-punt van het zelfgemaakte type wnggav_types.Position2D)
- Stoppositie van het overlay-element
- Een ID
- De HTML content van het overlay-element
- De HTML style van het overlay-element
- De interactie handlers van het element

videoOverlayEltFactory

Dit is een factory klasse die twee functies aanbiedt om objecten van het type '*VideoOverlayelement*' aan te maken. Eén functie maakt overlay-elementen aan uit van een array en de andere maakt overlay-elementen aan van een JSON string.

videoOverlayEltsRepo

Dit is een repository klasse waarin overlay-elementen worden in bijgehouden in een JsMap.

animParamsLinInterp

Dit is een klasse die de lineaire interpolatieparameters bijhoudt voor een bepaald overlay-element.

animationLinInterp

Dit is een klasse die een JsMap bijhoudt die een '*VideoOverlayelement*' object linkt aan een '*AnimParamsLinInterp'* object. De klasse gaat luisteren naar bepaalde custom events. Deze custom events laten de klasse weten wanneer nieuwe '*VideoOverlayelement'* objecten worden aangemaakt, gewijzigd of verwijderd. Het gaat dan voor deze objecten een object van het type '*AnimParamsLinInterp'* aanmaken en opslaan in de JsMap. De klasse biedt ook een '*doAnimationStep'* functie aan, deze krijgt een tijdstip en een '*VideoOverlayelement'* object binnen en gaat aan de hand van de '*AnimParamsLinInterp'* van dit overlay-element de positie berekenen.

videoOverlayManager

Dit is de klasse die de overlay-elementen gaat beheren. Deze klasse houdt twee objecten van het type '*videoOverlayEltsRepo*' bij, één voor objecten van het type '*VideoOverlayelement'* en één voor de domrepresentaties van de overlay-elementen. De manager heeft ook een verwijzing naar het video-element en een verwijzing naar een div element waar de overlay-elementen in worden getoond. Gebruikers kunnen objecten van het type '*VideoOverlayelement'* aan deze klasse meegeven, de manager zal dan het dom element van dit overlay-element aanmaken. Dit dom element krijgt de gepaste inhoud, stijl en interactiehandlers. De manager gaat er ook voor zorgen dat het overlay-element op het gepaste tijdstip wordt getoond op de correcte plaats. Dit doet de manager met behulp van een '*AnimationLinInterp*' object. De manager zal custom events afvuren wanneer een overlay-element wordt toegevoegd of verwijderd.

De manager heeft een '*onAnimationStep*' functie die om de tien millieseconden wordt afgevuurd als de video aan het spelen is. Indien de video op pauze staat, wordt deze functie niet herhaaldelijk aangeroepen. Deze functie gaat voor elk element de '*enforceOverlayEltVisibility'* functie aanroepen. Deze gaat kijken of het element zichtbaar is (dus of de huidige videotijd tussen de starttijd en stoptijd van het overlay-element ligt). Indien het zichtbaar hoort te zijn gaat de manager het overlay-element positioneren met behulp van de '*positionOverlayElt'* functie en vervolgens zichtbaar maken.

voe-def bestanden

Dit zijn javascript bestanden waarin de overlay-elementen voor een bepaalde video staan gedefinieerd. Een voe-def bestand ziet er als volgt uit:

```
// Interaction handler data structure for the UserControlledNarrative use case
var UCN_InteractionHandler = function (videoFileName) {
	this.m_videoFileName = videoFileName;
	var that = this;
	this.click = function (e, vidOverlayElt) {
	UserControlledNarrative_ihClick(e, vidOverlayElt, that.m_videoFileName);
	};
	this.mouseover = function (e, vidOverlayElt) {
	UserControlledNarrative_ihMouseOver(e, vidOverlayElt, that.m_videoFileName);
	};
	this.mouseout = function (e, vidOverlayElt) {
	UserControlledNarrative_ihMouseOut(e, vidOverlayElt, that.m_videoFileName);
	};
};

var g_voeDefs = [];

//Define the video overlay-elements and their properties
//g_voeDefs.push( [ id,   timeStart,   timeStop,    posStartX,   posStartY,
	posStopX,    posStopY,    htmlContent, interactionHandler,
                        htmlStyle ]);
g_voeDefs.push(["Id1",2,5,100,100,100,100,"",new UCN_InteractionHandler("lego"),""
]);
g_voeDefs.push(["Id2",3,13,400,400,400,400,"",new UCN_InteractionHandler("dizzy"),
"background-color: rgb(10,10,10); color: white;"]);
g_voeDefs.push(["Id3",10,21,200,30,200,30,"",new
UCN_InteractionHandler("happyfeet2"),"width: 230px; height: 400px;"]);
```

Het bestand bestaat uit een array waarin we overlay-elementen gaan definiëren. Eén van de parameters van een overlay-element is de interactionHandler die ook gedefinieerd staat in het bestand. De interactionhandler roept functies aan uit de index.php pagina van de use case waarin dit voe-def bestand wordt gebruikt.

wnggav_html5video

In dit bestand worden functies gegroepeerd die functionaliteit aanbieden over HTML5 video. De '*switchVideoClipPlusOverlayElts'* functie wordt gebruikt om de video van bestand te laten wisselen en laat de manager de oude overlay-elementen verwijderen en voegt de nieuwe overlay-elementen toe. Dit gebeurt met behulp van het voe-def bestanden. Eerst worden alle overlay-elementen uit de manager verwijderd. Vervolgens wordt het voe-def bestand ingeladen en krijgt de functie als parameter de naam van het voe-def bestand mee. Dit bestand wordt dynamish ingeladen met behulp van een jQuery.get(voeDefFileName) call. Deze functie gaat het bestand inladen en de code in het bestand beschikbaar maken voor gebruik. Wanneer het bestand ingeladen is krijgen we toegang tot de g_voeDefs array via window.g_voeDefs. De functie roept vervolgens de '*installOverlayEltsFromArray*' functie aan die de factory gaat gebruiken om de overlay-elementen aan te maken en in de manager te steken. Als laatste gaat de '*switchVideoClipPlusOverlayElts*' functie het videobestand van het video-element veranderen en afspelen.

4.3 Implementatie

Data inladen

Het aanmaken van overlay-elementen met motiontrackdata gebeurt analoog aan het aanmaken van normale overlay-elementen. In de voe-def bestanden maken we een nieuwe array aan *'g_voeDefsMotion'* . In deze array definiëren we overlay-elementen volgens het volgende formaat:

g_voeDefsMotion.push([id, motionTrackingDataFile, htmlContent, interactionHandler, htmlStyle, animationstyle]);

De motiontrackdata gaan we inlezen via een txt-bestand. De link naar het txt-bestand wordt meegegeven in de definitie van het overlay-element (motionTrackingDataFile). De '*dataprovider'* klasse gaat deze links omzetten naar een string met de inhoud van het bestand. Dat doet de klasse door een ajax call uit te voeren. Normaal gebeurt deze call asynchroon, namelijk dat men de get aanroept en de code verdergaat. Door een extra parameter mee te geven, gaat de call synchroon gebeuren waardoor we het verdere proces van het inlezen van overlay-elementen niet asynchroon moeten laten lopen. Door de call synchroon te houden kunnen we de code vereenvoudigen aangezien we niet moeten werken met events die worden afgevuurd eens de data is ingelezen.

De gebruiker kan ook een parameter meegeven om aan te duiden welke animatietechniek hij wil gebruiken (animationstyle). De verschillende soorten animatietechnieken zijn:

1. **Predefined boundingbox**: Voorgedefinieerd overlay-element met vaste grootte (ingesteld door de gebruiker) laten bewegen volgens de motiontrackdata. Doorheen de animatie blijft de grootte constant en enkel de positie van het element verandert. De positie wordt bepaald door de trackingdata.
2. **maximum animation boundingbox**: Voorgedefinieerd overlay-element met vaste grootte, die bepaald wordt door de maximale boundingbox over de hele animatie, laten bewegen

volgens de motiontrackdata. Doorheen de animatie blijft de grootte constant (de maximum grootte die het getrackte object zou hebben doorheen de hele animatie), enkel de positie van het element verandert. De positie en maximum grootte worden bepaald door de trackingdata.

3. **Dynamic bounding box**: Voorgedefinieerd overlay-element laten bewegen en scalen volgens de motiontrackdata. Doorheen de animatie verandert het element van grootte en van positie. Deze grootte en positie worden bepaald door de trackingdata.
4. **Outlined trackingpoints**: Overlay-element met daarin een Canvas laten bewegen en hertekenen doorheen de animatie. In dit canvaselement worden de trackingpoints getekend en met elkaar verbonden. Doorheen de animatie verandert de grootte, positie en inhoud (de getekende punten in het canvas) van het element. Deze worden bepaald door de trackingdata

In een later deel van deze thesis worden de animatietechnieken meer in het detail toegelicht.

Data parsen

Datastructuur

Voordat we overlay-elementen kunnen animeren volgens motiontrackdata, moeten we eerst de motiontrackdata parsen naar een algemeen formaat. Dat formaat moet zo algemeen mogelijk zijn zodat trackingdata van verschillende trackingprogramma's kan worden omgezet naar dit formaat.

Bijna alle trackingprogramma's werken met het tracken van punten over verschillende frames/tijdstippen. Het trackingdataformaat volgt deze denkwijze. Het formaat ziet er als volgt uit:

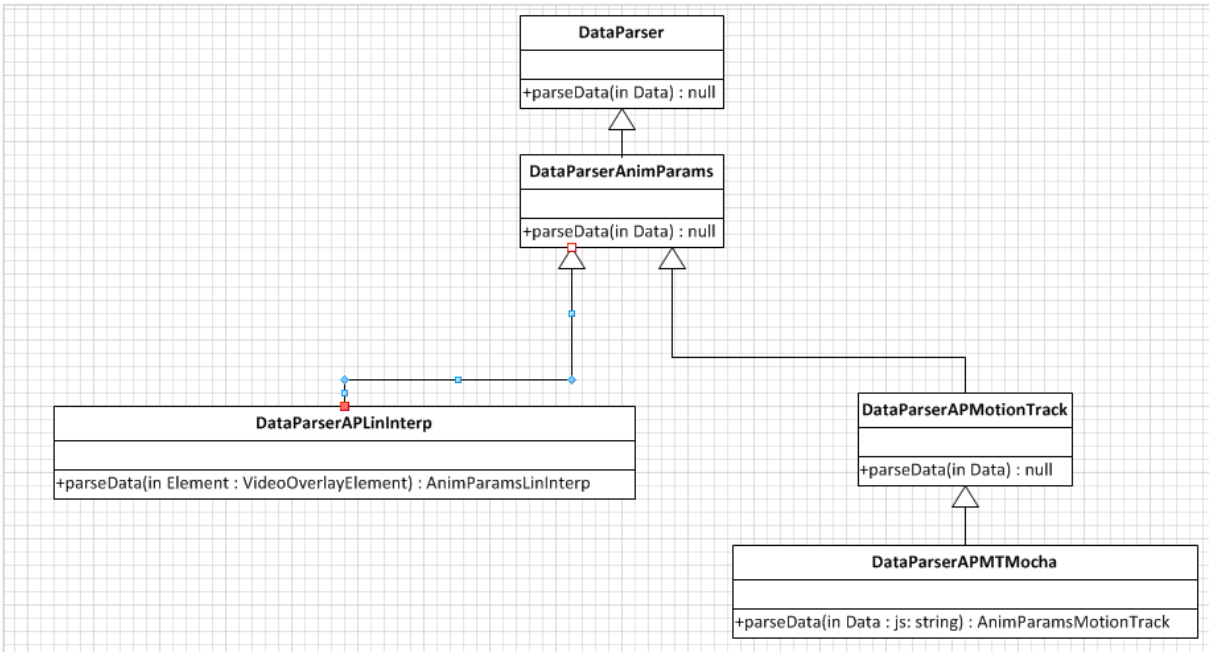
TrackingPunt1		TrackingPunt2		TrackingPunt3	
Tijdstip	Positie	Tijdstip	Positie	Tijdstip	Positie
Tijdstip	Positie	Tijdstip	Positie	Tijdstip	Positie
Tijdstip	Positie	Tijdstip	Positie	Tijdstip	Positie
Tijdstip	Positie	Tijdstip	Positie	Tijdstip	Positie

Ik heb hier gekozen voor het gebruik van tijdstippen in plaats van frames omdat er doorheen het programma constant wordt gewerkt met tijdstippen om de overlay-elementen te plaatsten. Dit is omdat de HTML5-video geen '*onNewFrame'* event ondersteunt dat elk frame wordt afgevuurd.

De positie is van het type *'wnggav_types.Position2D'*, een zelf gedefinieerd type dat een 2D-punt voorstelt.

Parsing structuur

De '*DataParser'* structuur ziet er als volgt uit:



Figuur 4.3: DataParser structuur

In de structuur hebben we een abstracte klasse *'DataParser'*. Van deze klasse gaan we al de verdere DataParsers afleiden. De klasse heeft één functie: *'parseData'*. Deze functie krijgt data in tekstuele vorm binnen als parameter en geeft het gewenste type terug.

Van de klasse '*DataParser'* leiden we de klasse *'DataParserAnimParams'* af. Deze klasse is een abstracte klasse waarvan andere klassen kunnen worden afgeleid die data gaan omvormen naar animatieparameters.

Voor lineaire interpolatie-animatie hebben we de klasse *'DataParserAPLinInterp'*. Deze klasse gaat een *'AnimParamsLinInterp'* object genereren uit een *'VideoOverlayElement'* object. Deze animatieparameters kunnen dan gebruikt worden door de lineaire interpolatie-animatie-engine.

Voor motiontrackdata heb ik nog een extra abstracte klasse gedefinieerd: *'DataParserAPMotionTrack'.* Van deze abstracte klasse kunnen we andere parsers afleiden die trackingdata van een specifieke bron, bv. Mocha, Adobe After Effects, gaat omvormen naar een '*AnimParamsMotionTrack' object*. De geparste data kan dan gebruikt worden door de motiontrack-animatie-engine.

Door deze structuur te hanteren, is het eenvoudig nieuwe DataParsers aan te maken voor verschillende bronnen van motiontrackdata.

Aanmaken overlay-element

Het aanmaken van de overlay-elementen gebeurt in de '*VideoOverlayEltFactory'* klasse. We voegen aan de factory een functie toe die overlay-elementen van het type motiontrack aanmaken. De begin- en eindtijd en begin- en eindpositie van het element wordt afgeleid aan de hand van de motiontrackdata. Posities worden als volgt berekent: van al de getrackte punten op een bepaald -tijdstip vragen we de positie op. We berekenen hier de minimale x- en y-waarden uit. Deze x- en y-waarde is dan de positie die we gaan gebruiken voor een overlay-element op een bepaald tijdstip.

Figuur 4.4: Voorbeeld trackingpoints

In deze afbeelding zijn de vier blauwe punten de getrackte punten op tijdstip t. Het rode punt is de berekende positie van het overlay-element gedefinieerd door de trackingindata op het tijdstip t.

Hieronder staat een beschrijving wat er gebeurt per animatietechniek wanneer het element wordt aangemaakt:

Bij de *'predefined boundingbox'* techniek gebruiken we de grootte die de gebruiker definieert in het '*htmlStyle'* veld als de grootte van het element doorheen de hele animatie van dit element. Indien de gebruiker hier geen grootte specificeert dan gebruiken we de standaard gedefinieerde grootte uit een css-file.

Bij de *'maximum animation boundingbox'* techniek gaan we op voorhand de grootste (axis aligned) bounding box berekenen die de trackingpunten hebben doorheen de hele trackperiode. Dit doen we door voor elk tijdstip in de trackingdata een bounding box te berekenen van de getrackte punten. We zoeken van deze bounding boxen de maximum breedte en hoogte en stellen deze in als breedte en hoogte van het overlay-element.

Bij de *'dynamic bounding box'* techniek gaan we voor het eerste tijdstip de breedte en hoogte berekenen van de bounding box van de trackingpunten. Deze stellen we in als breedte en hoogte van het element. Indien gebruikers in het '*htmlStyle'* veld een hoogte en breedte specifieren dan worden deze overschreven.

Bij de *'outlined trackingpoints'* techniek gaan we hetzelfde doen als bij *'dynamic bounding box'* maar we gaan hier het '*htmlContent'* overschrijven met een canvas. Op deze canvas gaan de trackingpunten getekend en met lijnen verbonden worden. Hierdoor krijgen we een veelhoek getekend, bij de vorige animatietechnieken was dit altijd een axis aligned bounding box.

InteractionHandler filter

Bij de *'outlined trackingpoints'* techniek gaan we een veelhoek tekenen van de getrackte punten. We willen dat het click event ook rekening gaat houden met de getekende veelhoek. Dat wil zeggen dat wanneer een gebruiker buiten de veelhoek (maar wel op het canvas/overlay-element) klikt dat het event niet geregistreerd mag worden. Om dit te bereiken heb ik '*InteractionHandler filters'*

geïmplementeerd. Deze filters zijn functies die vasthangen aan de klasse *'VideoOverlayelement'* . Standaard zijn dit functies die true teruggeven:

```
VideoOverlayelement.prototype.clickFilter = function(e){
	return true;
};
```

Wanneer er geklikt wordt op een video overlay-element wordt eerst de filterfunctie aangeroepen van het element. Enkel wanneer deze true teruggeeft, gaat het click event door naar de door de gebruiker ingestelde interactionhandler.

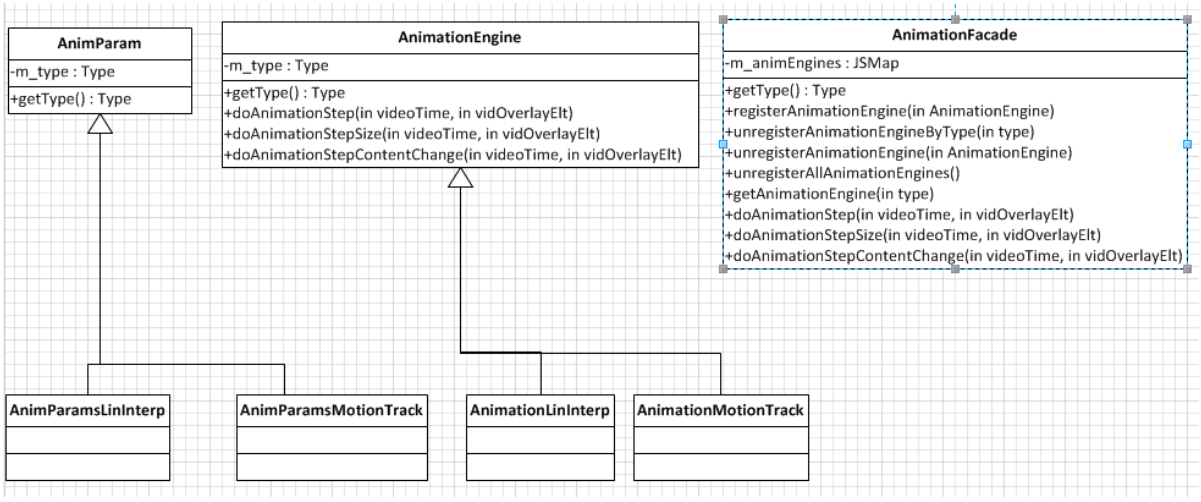
Bij de 'outlined trackingpoints' techniek gaan we een customfunctie instellen als clickfilter van het video overlay-element. Deze functie gaat van de aangeklikte pixel op het canvas de pixelwaarde opvragen. Wanneer we de punten en lijnen op het canvas tekenen tijdens het animeren, dan vullen we de veelhoek op met een witte kleur met zeer lage alphawaarden waardoor het niet zichtbaar is voor het blote oog. Pixels buiten de veelhoek hebben een alphawaarde van 0. De filterfunctie gaat dus in dit geval kijken naar de alphawaarden en gaat het clickevent doorlaten als de alphawaarde van de pixel niet nul is. Anders wordt het event genegeerd.

Verandering tegenover originele code

In de implementatie van dr. Maarten Wijnants werden de animatieparameters voor lineaire interpolatie overlay-elementen pas aangemaakt wanneer een overlay werd toegevoegd aan de manager. In de nieuwe implementatie is dit niet meer zo. De animatie parameters (zowel lineaire interpolatie als motiontrack animatie parameters) worden nu aangemaakt in de factory en worden meegegeven aan het '*VideoOverlayElement*' object als parameter. Hierdoor moeten de animatie engines ook geen lijst meer bijhouden van welke animatie parameters bij welke overlay-elementen horen.

Animeren overlay-elementen

Het animatiesysteem werd aangepast naar de volgende structuur ontworpen door Maarten Wijnants:



Figuur 4.5: Animatie structuur

Een overlay-element houdt nu een instantie van een afgeleide klasse van 'AnimParams' bij. Hierin staan de animatie parameters gedefinieerd voor dit overlay object. In de '*VideoOverlayManager*' is de functie '*positionOverlayElt*' aangepast zodat deze de huidige structuur ondersteunt. Hier gaat er nu gekeken worden naar welk type animatieparameters en type animatie het video overlay-element heeft. Naargelang het type worden er functies aangeroepen op de '*AnimationFacade*' klasse. De '*AnimationFacade*' klasse gaat kijken welk type '*AnimParams*' het binnenkrijgt en gaat dan de gelijknamige functies uitvoeren op de correcte animation engine. Deze engine gaat de feitelijke animatieberekeningen doen, maar gaat de animatie zelf niet uitvoeren. De resultaten van de berekeningen (de positie van het overlay-element in de video) worden teruggegeven aan de '*VideoOverlayManager*' klasse. Deze gaat dan de resultaten optellen bij de positie van het video-element op de pagina en dit instellen als de positie van het overlay-element. Dit gebeurt nog steeds in de '*positionOverlayElt*' functie.

Er zijn tot nu toe drie animatiefuncties toegankelijk. Elke animatie engine vult deze functies zelf in. De functies zijn:

1. DoAnimationStep: berekent de nieuwe locatie van een overlay-element
2. DoAnimationStepSize: berekent de grootte van een overlay-element
3. DoAnimationStepContentChange: berekent de inhoud van een overlay-element

Lineaire interpolatie

De lineaire interpolatie engine werd gemaakt door Maarten Wijnants. In de '*AnimParamsLinInterp*' worden de volgende parameters bijgehouden: deltaX, deltaY, deltaTime, deltaXperT en deltaYperT. De deltaXperT en deltaYperT stellen voor hoeveel het element beweegt per tijdseenheid. De lineaire interpolatie engine gaat in de '*doAnimationStep*' functie kijken naar de starttijd van de animatie evenals de huidige tijd en gaat aan de hand van deze waarden de huidig positie berekenen. Dit gebeurt door eerst te berekenen hoeveel het element moet bewegen per seconde (XperT,YperT). Erna kunnen we de positie berekenen aan hand van de volgende formule:

Huidige positie X = startpositie X + (huidige videotijd – start tijd overlay-element) * XperT

Huidige positie Y = startpositie Y + (huidige videotijd – start tijd overlay-element) * YperT

Motiontrack animatie

De 'AnimParamsMotionTrack' bevat de geparste motiontrackdata en het type animatie. De engine heeft de volgende implementatie voor de 3 doAnimation functies:

doAnimationStep

In deze functie wordt de positie berekend van een overlay-element. De positie wordt berekend aan de hand van de boundingbox van de motiontrackpunten op tijdstip T (zie "Figuur 4.4: Voorbeeld trackingpoints"). De positie van het overlay-element is de positie van de boundingbox in de video.

doAnimationStepSize

In deze functie wordt de grootte van het overlay-element berekend. De grootte wordt berekend aan de hand van de boundingbox van de motiontrackpunten op tijdstip T. De grootte van het overlay-element is de hoogte en breedte van de boundingbox in de video.

doAnimationStepContentChange

In deze functie wordt de inhoud van het overlay-element berekend. De inhoud voor een motiontrackelement zijn de lokale trackpunten op tijdstip T. Met lokale punten wordt bedoeld de relatieve positie van de trackpunten tegenover de positie van het overlay-element. Voor elk trackpunt worden de lokale coördinaten terug gegeven.

Naargelang welk type animatie er is, gaat de '*VideoOverlayManager'* de correcte functies aanroepen.

Predefined boundingbox

Bij deze animatietechniek wordt het voorgedefinieerde overlay-element verplaatst over het scherm volgens de motiontrackdata. Er worden dus geen veranderingen aangebracht op de grootte en inhoud van het overlay-element. Enkel de '*doAnimationStep'* functie wordt uitgevoerd voor dit soort animaties.

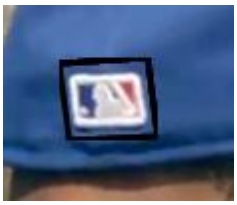
Maximum animation boundingbox

Bij deze animatietechniek wordt het voorberekende overlay-element verplaatst over het scherm volgens de motiontrackdata. Er worden dus geen veranderingen aangebracht op de grootte en inhoud van het overlay-element. Enkel de '*doAnimationStep'* functie wordt uitgevoerd voor dit soort animaties. Deze techniek heeft dus dezelfde animatie berekening als de predefined boundingbox techniek, maar verschilt in het feit dat de grootte van het element bij het aanmaken wordt bepaald door de motiontrack data.

Dynamic bounding box

Bij deze techniek gaan we een overlay-element verplaatsen en de grootte ervan aanpassen volgens de motiontrackdata. De positie wordt nog steeds bepaald door de '*doAnimationStep'* functie. De grootte van het overlay-element wordt berekend door de '*doAnimationStepSize'* functie.

Outlined trackingpoints



Figuur 4.6: Resultaat outlined trackingpoints techniek

Bij deze techniek gaan we een overlay-element verplaatsen volgens de motiontrackdata. De inhoud wordt aangepast zodat ze de positie van de trackpunten weergeeft op tijdstip T. De punten worden verbonden zodat de getrackte figuur wordt omringd door een veelhoek. Bij deze animatietechniek gaan de 3 animatiefuncties gebruikt worden. Zoals bij de Dynamic bounding box techniek worden de '*doAnimationStep'* en '*doAnimationStepSize'* functies gebruikt om de positie en grootte van het overlay-element te bepalen. De grootte van het canvas in het overlay-element wordt ook aangepast. In het canvaselement worden dan de punten getekend en verbonden volgens de resultaten van de '*doAnimationStepContentChange'* functie. Vervolgens wordt de veelhoek opgevuld met een witte achtergrondkleur met een zeer lage alphawaarde zodat de klikfilter van dit element correct kan werken. De alphawaarde is zo laag dat de gebruiker het niet zal opmerken met het blote oog.

4.4 Vergelijking resultaten van animatietechnieken

Hieronder geven we een vergelijking van de animatietechnieken. In dit voorbeeld maken we gebruik van een film over een groeiende plant. Er zijn 5 tijdstippen in het filmpje uitgekozen. Op elk van deze vijf tijdstippen is er een screenshot van de 4 technieken zichtbaar en een screenshot van de trackingdata zelf. Voor de predefined boundingboxtechniek hebben we een overlay-element gedefinieerd van 75 op 75 pixels groot.

Frame 479

Trackingdata

Predefined boundingbox

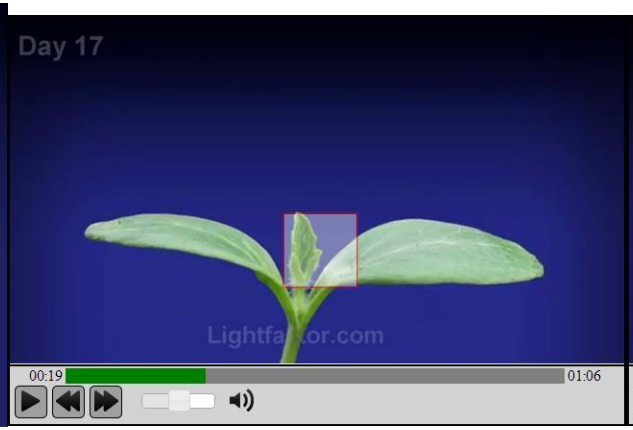
Maximum animation boundingbox

Dynamic bounding box

Outlined trackingpoints

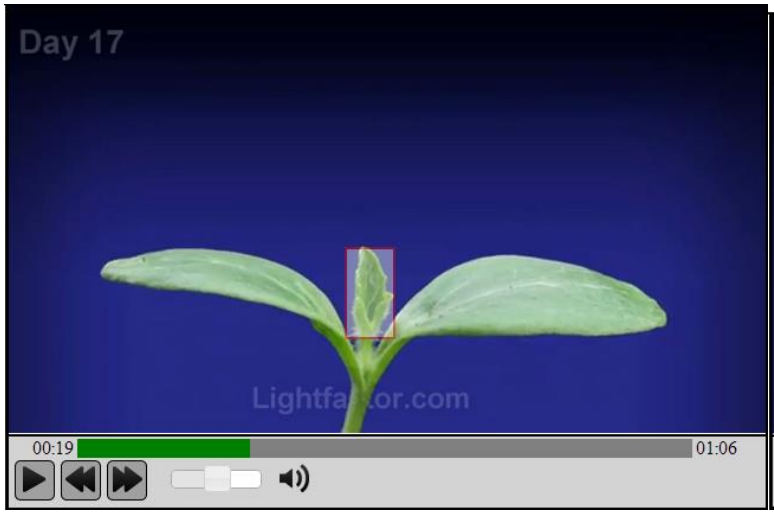
Frame 568

Trackingdata



Predefined boundingbox

Maximum animation boundingbox

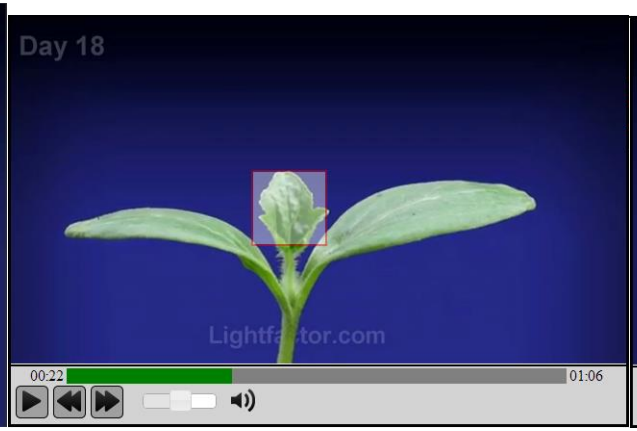


Dynamic bounding box

Outlined trackingpoints

Frame 657

Trackingdata



Predefined boundingbox

Maximum animation boundingbox

Dynamic bounding box

Outlined trackingpoints

Frame 746

Trackingdata

Predefined boundingbox

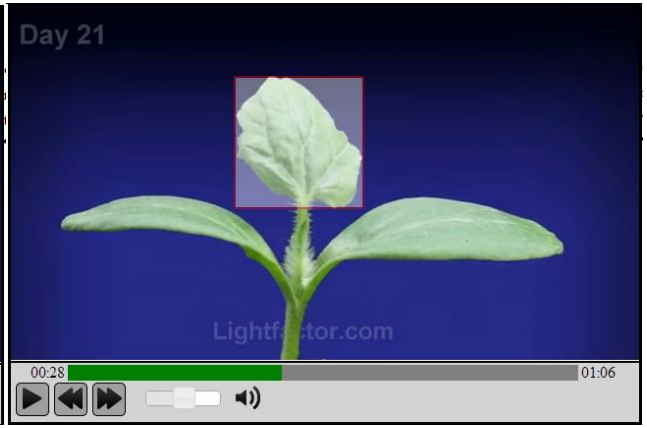
Maximum animation boundingbox

Dynamic bounding box

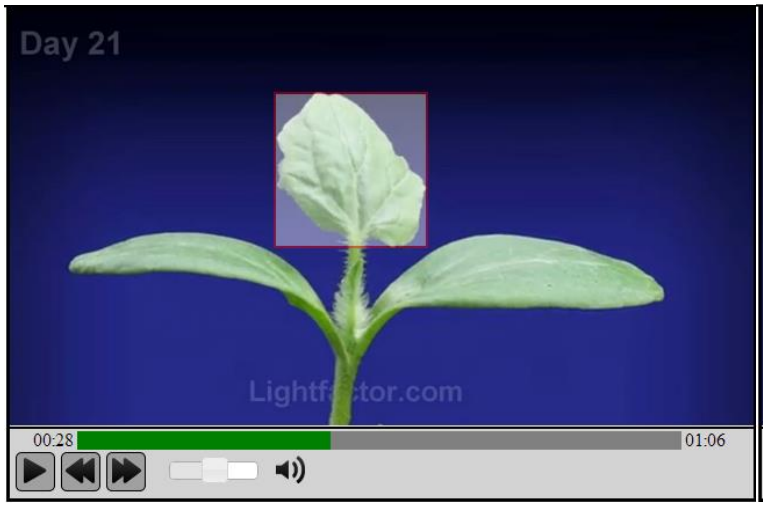
Outlined trackingpoints

Trackingdata

Predefined boundingbox



Maximum animation boundingbox



Dynamic bounding box

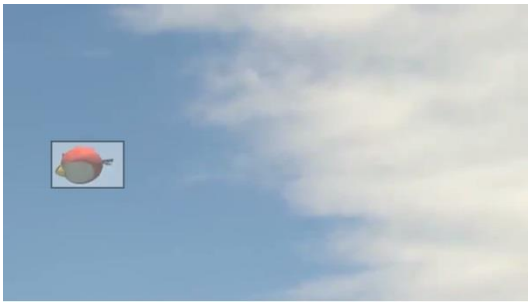
Outlined trackingpoints

Bij frame 479 biedt zich meteen een probleem aan dat kan optreden bij de eerste twee animatietechnieken. Wanneer de *'Predefined boundingbox'* techniek gebruikt wordt voor overlay-elementen van objecten die doorheen de film van grootte veranderen, dan gaat het overlay-element zelf niet van grootte mee veranderen. Hierdoor is het overlay-element alleen rond frame 657 relatief correct geplaatst. Bij de *'Maximum animation boundingbox'* techniek treed een gelijkaardig probleem op. Doorheen de video verandert het overlay-element niet van grootte. Omdat de grootte van het overlay-element ingesteld wordt als de maximale grootte staat het overlay-element alleen correct wanneer het getrackte object ongeveer dezelfde grootte heeft als zijn maximum grootte (zie frame 746 en verder).

Men kan best de *'Predefined boundingbox'* techniek gebruiken wanneer men overlay-elementen wil plaatsen op objecten die niet van grootte of rotatie veranderen. De *'Maximum animation boundingbox'* techniek kan men best gebruiken bij overlay-elementen die maar een beetje van grootte veranderen doorheen een film. Beide technieken zijn af te raden voor objecten waarvan de grootte veel verandert doorheen een film. Beide technieken gebruiken niet veel rekenkracht aangezien er alleen maar per frame voor een overlay-element een nieuwe positie moet berekend worden tijdens het afspelen van de film.

In het huidig voorbeeld geeft de *'Dynamic bounding box'* het beste resultaat. Dat komt omdat in elk frame het overlay-element wordt aangepast en zo telkens de optimale grootte en positie heeft. De trackingdata bestaat hier uit vier punten (de vierhoek die te zien is in de trackingdataframes). Dat zorgt ervoor dat de *'Outlined trackingpoints'* techniek een minder resultaat geeft vanwege het feit dat de figuur die we tracken een onregelmatige vorm heeft. Hierdoor wordt een gedeelte van het blad niet bedekt door het overlay-element. Een voorbeeld waarbij de *'Outlined trackingpoints'* techniek wel beter werkt is het volgende:

Figuur 4.7: Outlined Trackingpoints techniek



Figuur 4.8: Dynamic bounding box techniek

In dit voorbeeld vliegt een 'angry bird' over het scherm heen. We zien dat het overlay-element bij de *'Dynamic bounding box'* techniek groter is en meer lucht omvat dan die bij de *'Outlined trackingpoints'* techniek. Hoe goed de *'Outlined trackingpoints'* techniek werkt is dus afhankelijk van de kwaliteit van de trackingdata. Het framework ondersteunt trackingdata van meer dan vier trackpunten en zou dus betere resultaten kunnen geven, maar mocha AE geeft slechts vier trackingpunten weer in zijn data.

Men kan best de *'Dynamic bounding box'* techniek gebruiken wanneer men een object trackt dat doorheen de film van grootte verandert. De animatietechniek zal geen rekening houden met rotatie van het getrackte object. Hiermee wordt bedoeld dat wanneer een object in een video draait dan zal het overlay-element niet meedraaien (zoals te zien is in "Figuur 4.8", waarin de angrybird lichtjes

gedraaid is maar het overlay-element is nog steeds een axis aligned bounding box). Bij de *'Outlined trackingpoints'* techniek krijgen we wel het effect dat het overlay-element meedraait (zoals in "Figuur 4.7"). De animatietechniek gebruikt wel meer rekenkracht dan de '*Predefined'* en *'Maximum animation boundingbox'* techniek aangezien in elk frame ook de grootte wordt herberekend. De *'Outlined trackingpoints'* techniek verbruikt nog meer rekenkracht, maar geeft het beste resultaat. Het resultaat is bij deze techniek wel erg afhankelijk van de kwaliteit van de trackdata zelf.

4.5 Restricties en mogelijke uitbreidingen

In dit deel staan enkele beperkingen en mogelijke uitbreidingen op het systeem vermeld:

- Voorlopig ondersteunt het systeem enkel de resolutie van de film zelf. Dit wil zeggen dat het systeem verwacht dat de hoogte en breedte van het video-element moeten overeenkomen met de resolutie van de video. Het systeem doet voorlopig nog geen omzetting van coördinaten in de originele resolutie (dus de resolutie waarin objecten werden getrackt) naar de resolutie waarin de video getoond wordt.
- Telkens wanneer een gebruiker nu een video kijkt, moet het systeem de motiontrack data inladen en parsen naar animatieparameters. Het zou beter zijn als het syteem werd aangepast zodat dit maar eenmalig moet gebeuren. Dus dat de motiontrackdata éénmalig geparsed wordt en ergens wordt opgeslagen, en vervolgens van die plaats ingelezen telkens de data nodig is. Dit zou als nadeel hebben dat er meer opslagruimte moet voorzien worden voor deze data.
- We zouden het animatiesysteem kunnen optimalizeren door in een preprocessstap al de animatieberekeningen te doen. Hierdoor moet enkel at runtime de positie ingelezen en ingesteld worden. Dit zou het systeem lichter maken at runtime. Hierdoor is er wel weer meer storage nodig en hebben we een delay bij het opstarten van een video.
- Overlay-elementen die geheel of gedeeltelijk buiten de randen van het video-element treden worden niet geclipt. Deze feature is echter wel ingbouwd in het systeem rond omnidirectionele video.
- Het systeem is te zwaar voor gebruik op tablets. De '*onAnimationStep'* functie wordt normaal om de 10 milliseconden aangeroepen. Op tablets gebeurt dit echter tussen de 50 en 100 milliseconden. Dit is veel te weinig om een vlotte animatie te bekomen. Het systeem moet dus geoptimaliseerd worden vooralleer het klaar is voor gebruik op tablets.

4.6 Use case

Rond dit systeem heb ik een use case ontworpen. Deze use case is een aangepaste versie van de "UserControlledNarrative" use case ontworpen door dr. Maarten Wijnants. De bedoeling van de use case is dat de gebruiker zelf bepaald in welke richting het verhaal gaat. De gebruiker kan keuzes maken die invloed hebben op het verhaal. Deze keuzes zijn gekoppeld aan overlay-elementen in de video. In de originele use case krijgt de gebruiker een video te zien waarop overlay-elementen waren geplaatst. Wanneer de gebruiker over deze overlay-elementen hovert met zijn muis dan verschijnt er links van de video informatie over het overlay-element en pauzeert de video. Als de gebruiker op het element klikt dan verandert de video en worden andere overlay-elementen getoond.

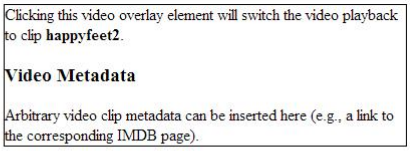
Ik heb de use case zo aangepast dat de inhoud van de video van belang is voor de overlay-elementen. In deze nieuwe use case krijgen gebruikers een video over angryBirds te zien. Gebruikers kunnen zelf bepalen welke richting het verhaal uit gaat door op bepaalde momenten op overlay-

elementen te klikken. Zo krijgt de gebruiker mogelijkheid in de eerste video om te kiezen wie er in de volgende video de angryBird gaat afvuren (zie “

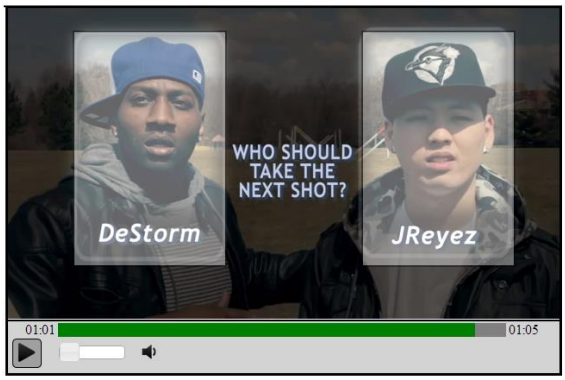
Figuur 4.10: Keuze volgende worp

”). De keuze bepaalt welke video als tweede wordt afgespeeld. In de tweede video moet de gebruiker op het correcte moment op een voorbij vliegende angryBird klikken. Lukt dit de gebruiker, dan gaat hij door naar de laatste video met het succesvolle einde. Lukt dit de gebruiker niet, dan blijft de gebruiker in de huidige video waarin het niet succesvolle einde zit. De gebruiker krijgt dan ook de kans om terug te gaan naar het begin of om naar de Youtube pagina te gaan van de maker van de video.

In de eerste video worden bepaalde overlay-elementen geplaatst volgens tracking data. Er wordt bijvoorbeeld op een bepaald moment een overlay-element getoond op het logo van de pet en op de oorring van een personage (zie “Figuur 4.11: Pet en oorbel overlay-element”). Wanneer de gebruiker op deze overlay-elementen klikt dan wordt er een site geopend in een iframe-element dat naast de video gepositioneerd is (zie rechts in “Figuur 4.11”). Bijvoorbeeld wanneer gebruikers op de oorring klikken, dan wordt de wikipediapagina van oorbel geopend.



Figuur 4.9: Use case dr. Maarten Wijnants

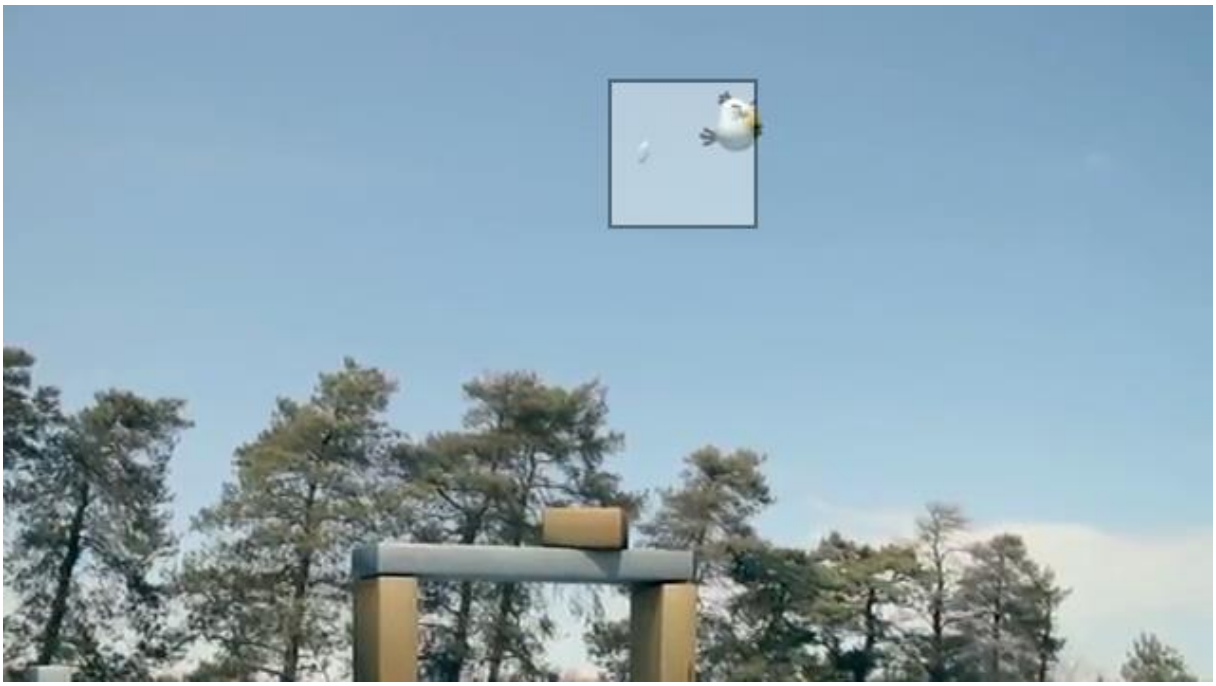


Figuur 4.10: Keuze volgende worp

Figuur 4.11: Pet en oorbel overlay-element

Conclusie use case

Met het nieuwe type overlay-element kunnen we nu overlay-elementen plaatsen in de video waar het voordien nog praktisch onhaalbaar was. Voordien konden er geen overlay-elementen geplaatst worden op objecten die bewegen in een video. Met de vier technieken die zijn toegevoegd is dit nu wel mogelijk. Voor sommige situaties is het echter toch aan te raden een statisch overlay-element te gebruiken. Dit is het geval in de tweede video die een gebruiker zou zien in de use-case. In de tweede video vliegt er een angryBird voorbij die een gebruiker op het juiste moment moet aanklikken. Indien we hier een overlay-element zouden laten bewegen volgens de trackingdata van de angryBirds dan zou dit overlay-element te snel bewegen. Hierdoor krijgt de gebruiker niet de kans om er op te klikken. Door een statisch element te gebruiken is het hier makkelijker voor de gebruiker om op het overlay-element te klikken (zie “Figuur 4.12”).



Figuur 4.12: Statisch overlay-element dat verwijst naar angryBirdsEnd.mp4

5 Web analytics tools

5.1 Literatuurstudie

Web analytics tools zijn programma's waarmee een eigenaar van een website informatie kan opvragen over zijn bezoekers. Zo kan de eigenaar zien hoeveel bezoekers de pagina heeft en waar deze bezoekers vandaan komen. Ook kan men zien welke zoektermen bezoekers gebruikten om op de site terecht te komen. Een andere interessante feature is dat de eigenaar kan zien waar de bezoekers het meest op klikken. Zo weet de eigenaar welke delen van de site succesvol zijn en welke minder, zodat hij hierop kan inspelen.

In dit deel van de masterproef wil ik een systeem ontwikkelen waarmee we kunnen bijhouden hoe de gebruikers omgaan met de overlay-elementen in een augmented video scenario. Welke elementen worden het meest aangeklikt? Hoe lang duurt het voordat een gebruiker een overlay-element aanklikt? Kijken gebruikers de video af of spoelen ze bepaalde delen door?

Voor deze masterproef zijn we op zoek gegaan naar een web analytics tool met de volgende eigenschappen:

- gratis
- click data bijhouden
- custom events toestaan (events die de maker van de site zelf kan definiëren)

Voor de literatuurstudie van dit onderdeel ben ik eerst op zoek gegaan naar sites die web analytic tools vergelijken. Uit deze vergelijkingen heb ik drie web analytics tools gekozen om uit te proberen. Deze drie kwamen het vaakst voor in verschillende opsommingen van web analytics tools en leken het best te passen in mijn gewenste implementatie. Hieronder staat een beschrijving van de drie gekozen web analytics tools.

Google Analytics

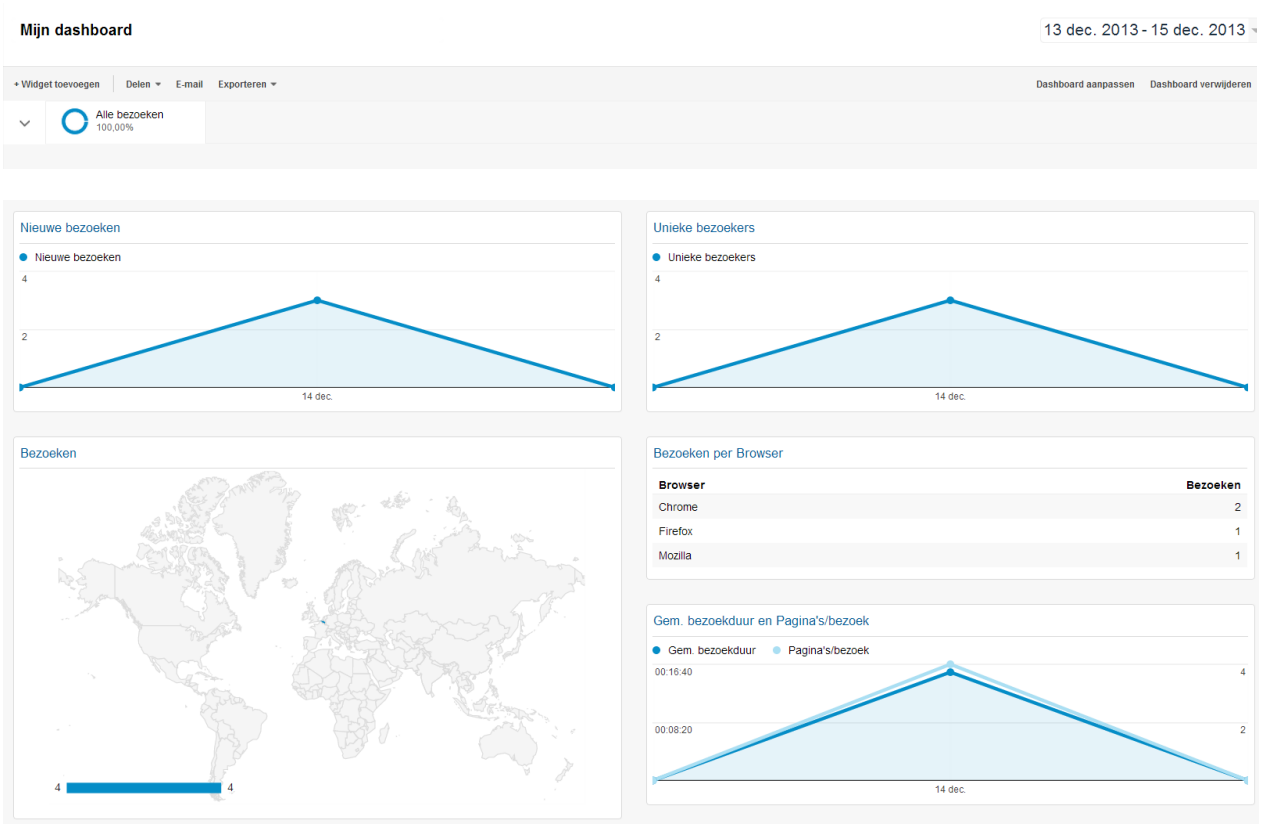
Google Analytics is één van de meest gebruikte web analytics tools. Met dit programma kan een eigenaar zien hoe gebruikers op de site terecht komen, via welke keywords de meeste gebruikers naar de site komen, wat gebruikers op de site doen, ... Google Analytics is gratis, eenvoudig te gebruiken en een robuuste tool. Het wordt ook regelmatig gebruikt als standaard om te vergelijken met andere web analytics tools[26][27]. Een nadeel van Google Analytics is dat het geen real time data weergeeft in het dashboard[28]. Als we kijken naar sites die analytics tools vergelijken dan zien we vaak dat Google Analytics bovenaan het lijstje staat. Google Analytics ondersteunt custom events[29] maar ondersteunt geen statistieken voor individuele gebruikers. De events worden altijd geaggregeerd voor alle gebruikers

Google Analytics is eenvoudig te installeren op een website. De ontwerper moet enkel het volgende script invoegen in zijn site:

```
<script>
      (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
        (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.par
entNode.insertBefore(a,m)})(window,document,'script','//www.google-
analytics.com/analytics.js','ga');
        ga('create', 'UA-XXXXXXX-X', name);
        ga('send', 'pageview');
</script>
```

De eigenaar zal zelf het ID van de site 'UA-XXXXXXX-X' - dit wordt automatisch gegenereerd door Google Analytics - en een naam voor het trackerobject moeten ingeven. Het script moet men onder de <Head>-tag zetten. Nadat dit script is ingevoegd, begint Google Analytics met het bijhouden van bezoekersdata.

Het belangrijkste component uit Google Analytics is het Dashboard. Dit krijgt de eigenaar meteen te zien wanneer hij zich inlogt in zijn Google Analytics account.



Figuur 5.1: Google Analytics Dashboard

Het dashboard bestaat uit verschillende widgets die bezoekersdata weergeven. De gebruiker heeft hier de mogelijkheid om eigen widgets aan te maken of de bestaande widgets aan te passen. De gebruiker kan instellen voor welke datums hij data wil laten zien door dit rechtsbovenaan aan te duiden. De widgets geven algemene data weer zoals hoeveel bezoekers er op de site zijn geweest, uit welk land de bezoekers komen, welke browser ze gebruiken en hoe lang ze op de site verbleven. Deze data wordt niet real time geüpdatet: Google Analytics verzamelt al de data na een dag en geeft deze dan pas weer.

Google Analytics biedt gebruikers wel de mogelijkheid om real time te zien wie er op de site is. Zo kan de eigenaar, zoals op het dashboard, informatie opvragen over waar huidige bezoekers vandaan komen, op welke pagina ze zich bevinden, ...

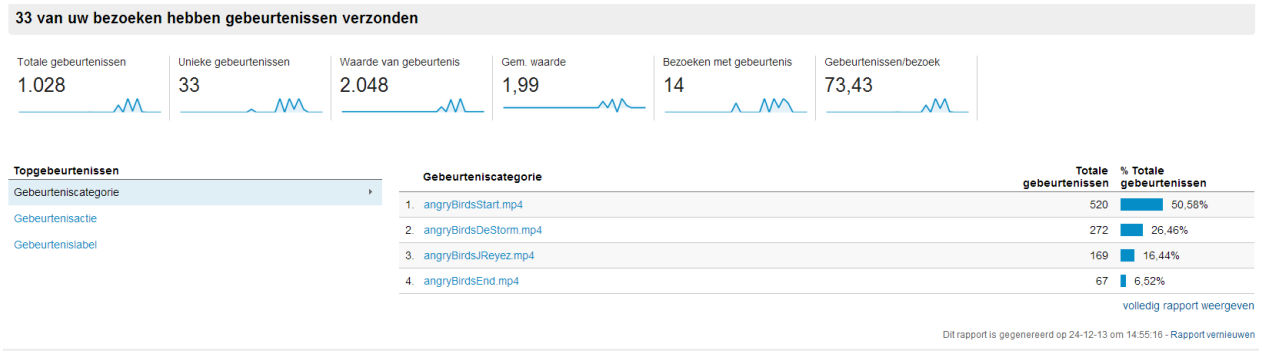
Google Analytics biedt de gebruiker ook de mogelijkheid zelf custom events te definiëren. Met custom events kan een eigenaar van een site enkele acties definiëren die hij wil tracken. Events worden geaggregeerd. Dit wil zeggen dat de eigenaar van een site niet kan zien welke events een individuele gebruiker heeft afgevuurd. Er kan enkel gekeken worden naar hoe vaak een event in totaal werd afgevuurd voor alle gebruikers. Een event wordt afgevuurd door in JavaScript de volgende functie aan te roepen:

_trackEvent(category, action, opt_label, opt_value, opt_noninteraction)

De category, action en opt_label zijn tekst velden, de opt_value is een integer en opt_noninteraction is een boolean. De laatste parameter laat Google Analytics weten of de events moeten worden gebruikt bij de bounce-rate berekeningen. De bounce-rate is een waarde die aangeeft hoeveel gebruikers naar een site komen en weggaan zonder andere pagina's te bezoeken op de site. Met de eerste drie parameters kan de gebruiker het event definiëren. Zo kan iemand die video events wil tracken de volgende structuur gebruiken:

category = "video" action=actie uitgevoerd op video opt_label= film naam

Met bovenstaande structuur kan een eigenaar bijvoorbeeld zien hoeveel keer gebruikers bepaalde acties uitvoeren per film. Voor de toekomstige implementatie gaan we echter een andere structuur hanteren. Een gebruiker kan real time zien welke events er binnenkomen, welke categorie, actie en label deze events hebben maar niet welke waarden. Na een dag wordt de eventdata geaggregeerd en weergegeven aan de gebruiker.



Figuur 5.2: Google Analytics Event overzicht

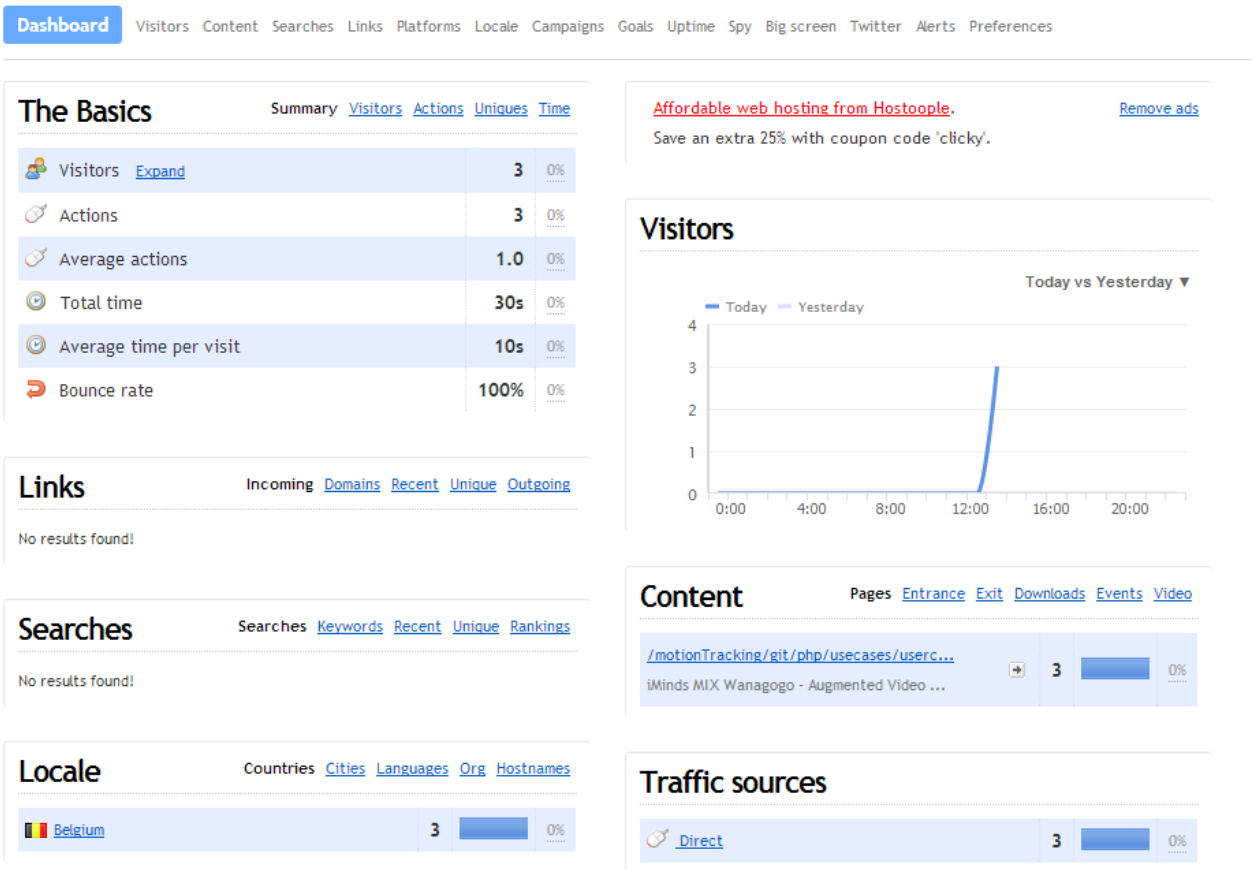
Deze events hebben wel enkele beperkingen. Zo staat er een maximum op het aantal events dat er kan verstuurd worden. Wanneer een bezoeker op de site komt, krijgt hij een 'event counter' toegewezen. Deze counter start met de waarde 10 en wordt met 1 verlaagd telkens als de bezoeker een event triggert (en er dus een event van de bezoeker naar de Google Analytics server wordt verzonden). Wanneer de counter op 0 staat, worden getriggerde events niet langer verstuurd. De counter wordt wel elke seconde verhoogd met 1 tot zijn maximum waarde (10).

Volgens de gebruikersovereenkomst van Google Analytics mogen ook geen persoonlijke gegevens of gegevens die een bezoeker kunnen identificeren, meegegeven worden aan deze events. Daarom zijn

deze events ook niet geschikt om individuele bezoekers op een site te tracken. Deze events zijn eerder bedoeld om een algemene indruk te krijgen van wat een gebruiker gemiddeld doet op de site.

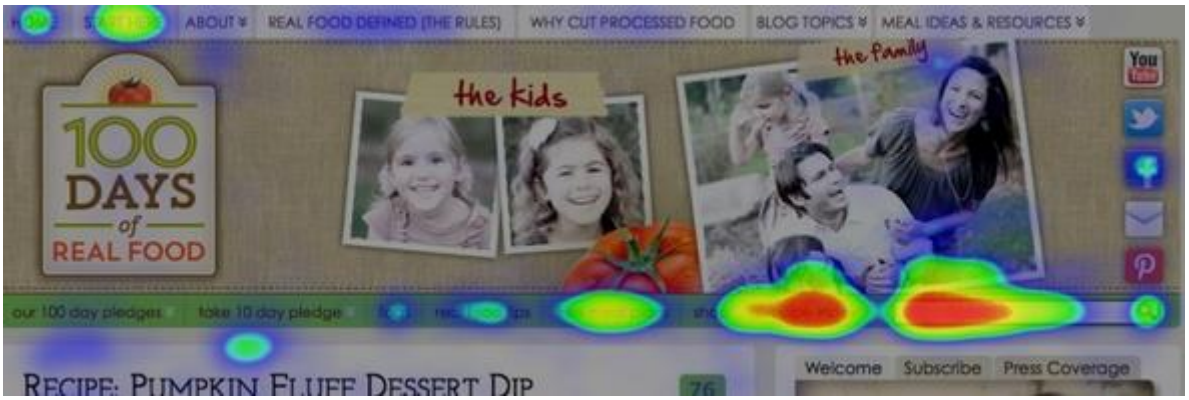
Clicky

Clicky is een gratis web analytics tool die de gebruiker toestaat om real time gegevens op te vragen. Ook heeft het een feature genaamd “Spy View” waarmee de gebruiker kan zien waarmee huidige bezoekers bezig zijn op de site. Deze spy view is alleen beschikbaar voor premium users. Clicky is gratis voor sites die minder 3000 page views per dag hebben. Clicky is zeer aanpasbaar en kan ook zoals Google Analytics JavaScript event tracken. Een interessante feature die Clicky ook ondersteunt is het genereren van ‘hitte mappen’ van waar gebruikers klikken op de site[30]. Met deze hitte mapen kunnen eigenaars zien op welke elementen op hun site er veel geklikt wordt.



Figuur 5.3: Clicky Dashboard

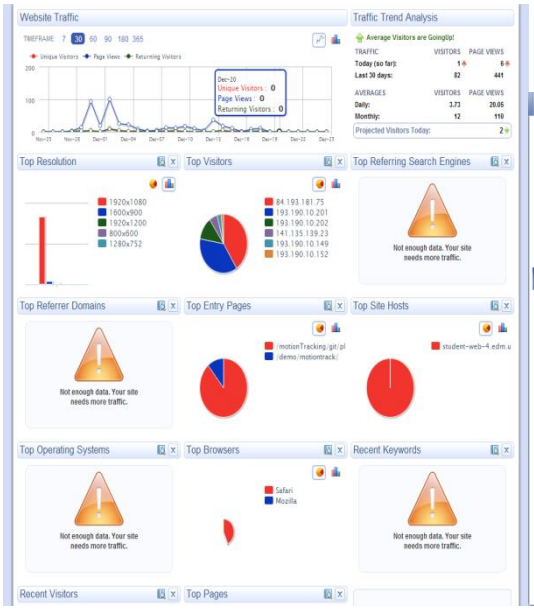
Zoals Google Analytics moet ook deze web analytics tool geïnstalleerd worden door tracking code in de webpagina in te voegen. Clicky biedt ook een dashboard aan om data weer te geven. Dit dashboard is echter eenvoudiger in vergelijking met dat van andere web analytics programma’s. Clicky biedt gebruikers de mogelijkheid om doelen en acties te definiëren maar deze functionaliteit is enkel beschikbaar voor premium gebruikers.



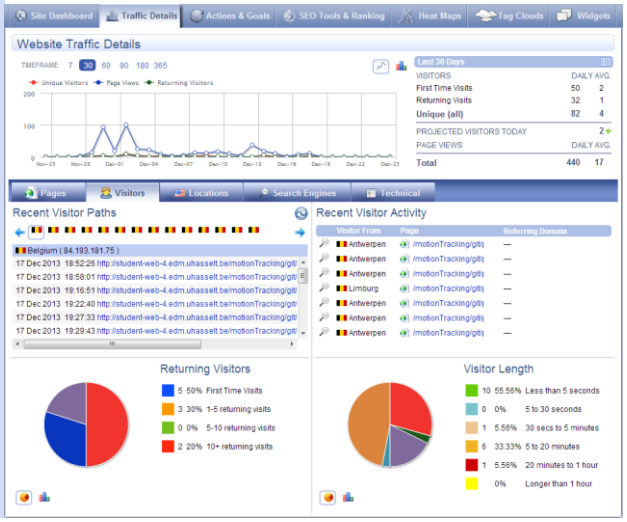
Figuur 5.4: Clicky heatmap

GoingUp

Deze analytics tool is volledig gratis en geeft een ruim aanbod van functionaliteiten [31]. Zoals Clicky ondersteunt ook dit programma hitte mappen van klicks. Er worden ook SEO of Search Engine Optimalization tools aangeboden. Dat zijn tools waarmee eigenaars van een site ervoor kunnen zorgen dat hun site hoger staat in de resultatenlijst wanneer mensen iets opzoeken in een zoekmachine.



Figuur 5.5: GoingUp Dashboard



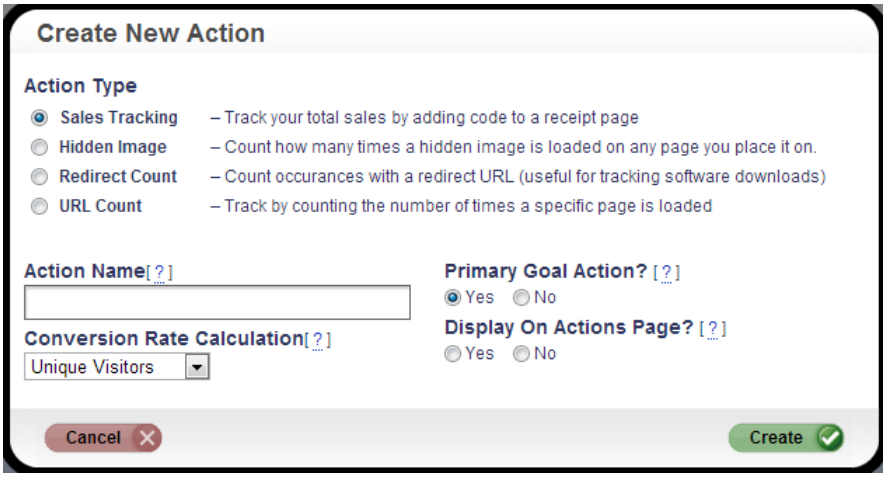
Figuur 5.6: GoingUp WebTrafic Details: Visitors

Zoals Google Analytics biedt ook GoingUp een dashboard aan waar gebruikers informatie kunnen opvragen over hun website. Dit dashboard wordt wel real time geüpdatet, dus we kunnen de informatie van de dag zelf ook zien. Het dashboard is gelijkaardig aan dat van Google Analytics, het bestaat uit verschillende widgets die de gebruiker kan aanpassen.

In tegenstelling tot Google Analytics biedt GoingUp wel de mogelijkheid om verschillende bezoekers van een site te identificeren en te tracken. Dit kan echter een privacy isue zijn volgens de wetgeving van sommige landen. Zo geeft GoingUp gebruikers de mogelijkheid om te zien welke pagina's een specifieke bezoeker bezocht heeft en in welke volgorde hij bepaalde pagina's heeft bekeken. Ook geeft GoingUp een Google Maps map weer van de locatie van de bezoekers van de site. Verder kan

men ook informatie zien over welke browser de bezoeker gebruikte en op welke resolutie hij de site bekeken heeft.

GoingUp biedt de gebruiker ook de mogelijkheid acties te definiëren. Deze zijn echter beperkt tot vier soorten (zie “Figuur 5.7: Create New Action bij GoingUp”). Ook kan men met de gratis versie van GoingUp slechts drie acties definiëren. Dus ofwel drie sales tracking events of drie hidden Image events of een combinatie van de drie van de vier soorten events.



Figuur 5.7: Create New Action bij GoingUp

Keuze:

Ik heb er voor gekozen om Google Analytics te gebruiken in combinatie met een zelf geschreven trackingsysteem omwille van de volgende redenen:

- Google Analytics is eenvoudig in gebruik en is zeergoed gedocumenteerd. Er zijn ook hulpfora beschikbaar waar vragen gesteld kunnen worden. Bovendien zijn er veel tutorials beschikbaar. Op de fora staan veel vragen die gebruikers al hebben opgelost. Ook zijn er veel mensen die blogs schrijven over hoe we bepaalde problemen kunnen aanpakken met Google Analytics;
- Het feit dat Google Analytics niet real time zijn data weergeeft in het dashboard is een nadeel. Dit is echter niet storend eens we een correct geïmplementeerd systeem hebben. Het is wel storend tijdens het implementeren van een trackingsysteem met Google Analytics aangezien we telkens een dag moeten wachten om te kijken of al de data correct is aangekomen zoals bedoeld was. Aangezien het doel niet is om de event data in real time ter beschikking moeten hebben is dit een aanvaardbaar nadeel.
- Google Analytics biedt niet de mogelijkheid om individuele personen te tracken. Sommige andere analytics programma’s bieden dit wel aan, maar ook slechts in beperkte mate of tegen betaling. Aangezien we het willen mogelijk maken dat het individuele personen op de site toch getrackt worden (bijvoorbeeld voor test doeleinden), is het beter het zelf te implementeren. Aangezien ik het zelf implementeer, weet ik exact welke data ik opsla en is het makkelijk hier ook aanpassingen in te maken.
- Het custom eventsysteem van Google Analytics is eenvoudig in gebruik en goed gedocumenteerd. Het is eenvoudig om zelf events te definiëren. Het maximum aantal events

dat kan gestuurd worden per tijdsinterval is hoog genoeg en wordt snel genoeg bijgevuld, waardoor een beperkt aantal events redelijk nauwkeurig hetrackt kan worden.

Ik ga Google Analytics gebruiken om algemene data te verzamelen over de bezoekers van de pagina. Zo zal ik dankzij Google Analytics te weten komen hoe lang een bezoeker naar bepaalde films kijkt, hoe vaak er geklikt wordt op bepaalde elementen, en hoe interactief gebruikers met bepaalde films omgaan. Het zelfgemaakte recorder systeem is voornamelijk bedoeld om gebruikerstesten af te nemen. Met deze gebruikerstesten kunnen makers van interactieve films in detail zien welke acties een specifieke gebruiker heeft uitgevoerd tijdens het consumeren van hun film. We kunnen dit ook gebruiken om gebruikerstesten rond het interactieve videosysteem zelf af te nemen.

5.2 Implementatie

Recorder systeem

Voor het recorden van events heb ik een nieuwe klasse gedefinieerd, namelijk de '*InteractionRecorder*'-klasse. Van deze klasse gaan we ander klassen afleiden die data gaan opslaan op bepaalde plaatsen. Voor het Google Analytics gedeelte gaat de afgeleide klasse '*GoogleAnalyticsRecorder*' events versturen naar een Google Analytics server. Voor het zelfgemaakte deel gaat een afgeleide klasse '*DatabaseRecorder*' een PHP-pagina aanroepen die data gaat invoeren in een MySQL-database. Het recorden van eventdata gebeurt door functies aan te roepen op deze recorderobjecten.

Bepaalde klassen zoals de '*VideoOverlayManager*' en de *'VideoPlayer'* worden aangepast zodat ze met een recorder overweg kunnen. Deze klassen krijgen een nieuwe member variabele die verwijst naar het recorderobject. Als deze variabele niet NULL is, dan weten deze klassen dat ze functies moeten aanroepen op dit object om events te laten registreren. Een voorbeeld hiervan is de terugspoelknop van de videoplayer. Wanneer een gebruiker klikt op de knop, wordt de volgende code aangeroepen:

```
if(recorder!=null){
            recorder.recordClickOnVideoPlayerButton( myVideo.currentTime,
"FastRewind");
}
```

In de huidige implementatie is er nog een derde afgeleide klasse gemaakt genaamd '*GAAndDBRecorder'.* Deze klasse dient als combinatie van de vorige twee recorderklassen en heeft dus een instantie van de *'DatabaseRecorder'* en de 'Google*AnalyticsRecorder'* als membervariabelen. Als de functies van de '*GAAndDBRecorder*' klasse worden aangeroepen, dan gaan deze de gelijknamige functies op de membervariabelen aanroepen. We willen bijvoorbeeld zowel in de database als Google Analytics opslaan wanneer een gebruiker klikt op een overlay-element, dus de code van de '*GAAndDBRecorder*'-klasse ziet er als volgt uit:

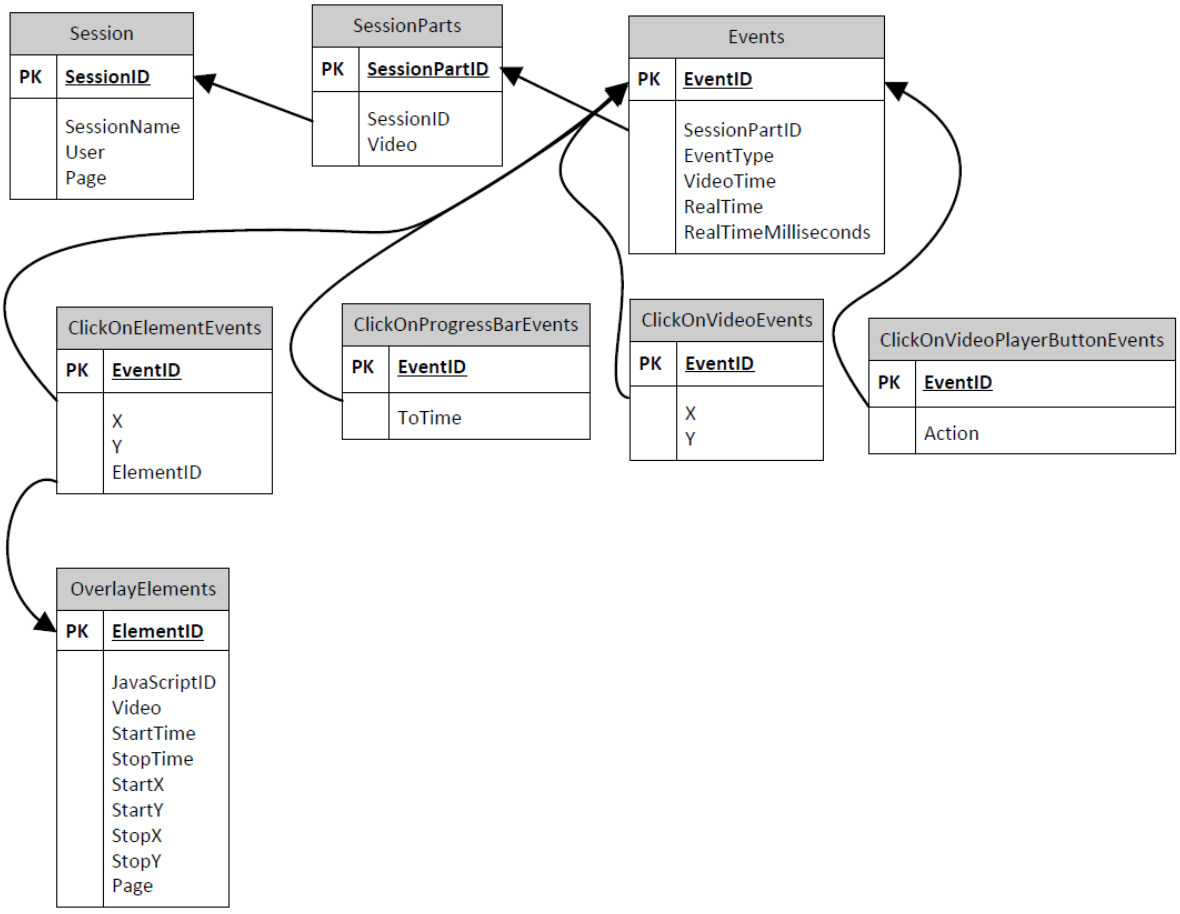
```
GAAndDBRecorder.prototype.recordClickOnElement = function
(overlayEltID,video,videoTime,x,y,page) {
  this.m_DBRecorder.recordClickOnElement(overlayEltID,video,videoTime,x,y,page);
  this.m_GARecorder.recordClickOnElement(overlayEltID,videoTime,x,y);
};
```

Database recorder

Met de databaserecorder gaan we bijhouden wat gebruikers allemaal doen tijdens een testsessie. Testsessies worden opgedeeld in testsessiedelen (‘*SessionParts*’). Telkens wanneer de gebruiker een nieuw filmpje kijkt, wordt er een nieuw testsessiedeel aangemaakt. Om data op te slaan in de database hebben we de afgeleide klasse ‘*DatabaseRecorder*’ gemaakt. We gaan de volgende dingen bijhouden in de MySQL-database:

- Welke filmpjes keken de gebruikers tijdens de testsessie?
- Op welke overlay-elementen klikte de gebruiker tijdens een testsessie?
- Waar in de film klikte de gebruiker tijdens het kijken van een film?
- Welke interacties deed de gebruiker met de video (play/pause, versnellen, doorspoelen,...)?

De database waar we deze data in opslaan heeft de volgende structuur:



Figuur 5.8: Database schema

We definiëren een event als iets wat gebeurt tijdens een SessionPart op een bepaald tijdstip in de video. We houden hier ook de echte tijd bij van het event zodat we kunnen zien hoeveel tijd er tussen bepaalde events zit bijvoorbeeld wanneer de video op pauze staat. Per soort event definiëren we een nieuwe tabel. In deze tabellen verwijst elk record naar een record in de ‘*Events*’-tabel. Per eventsoort wordt er bijkomende informatie opgeslaan in de tabel voor dat soort event. Bijvoorbeeld bij een ‘*ClickOnVideoPlayerButtonEvent*’ gaan we bijhouden welke actie de gebruiker heeft

uitgevoerd op de videoplayer (play/pause/versnellen/vertragen/mute). In de tabel '*Events*' zien we een kolom genaamd '*Real TimeMilliseconds*' . Deze kolom zorgt ervoor dat het real time tijdstip tot op de milliseconde nauwkeurig is aangezien MySQL versie 5.5.31 geen milliseconden ondersteunt bij tijdstippen. De '*VideoTime*' kolom heeft als type DOUBLE aangezien de currenttime van het HTML5-video-element ook altijd een double terug geeft. Sessies gebeuren op een bepaalde webpagina, overlay-elementen zijn uniek gedefinieerd door de pagina, JavaScriptID en video.

Er bestaat geen manier om direct vanuit JavaScript gegevens in een MySQL-database te plaatsen. Daarom gaan we dit doen via een PHP-pagina. Deze pagina gaan we aanroepen vanuit JavaScript door middel van een AJAX call. Aan deze AJAX call geven we een string mee waarin staat welke query de PHP-pagina moet uitvoeren en wat de parameters zijn van deze query. De querry wordt als een GET parameter toegevoegd aan het HTTPrequest

JavaScript AJAX call:

```
xmlhttp.open("GET","http://student-web-
4.edm.uhasselt.be/motionTracking/git/PHP/MySqlInterface/InsertQuery.PHP?q="+query,
false);
xmlhttp.send();
```

PHP opvragen van de parameters:

```
$request = $_GET['q'];
```

De waarde die wordt doorgestuurd als GET parameter heeft volgende syntax:

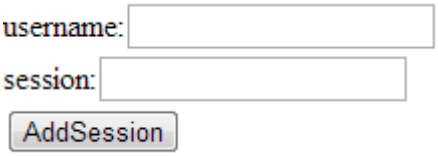
Var waarde = code voor query + ";" + waarde1+","+waarde2+","+....

Bijvoorbeeld:

"INSERT sessionParts;Sessie1,AngryBirdsStart.mp4"

De PHP-pagina gaat deze string ontleden en kijken naar het eerste deel om te zien welke query er moet uitgevoerd worden (in dit voorbeeld is dit een INSERT-query op de table SessionParts). Daarna gaat de PHP-pagina alle juiste parameters halen uit het tweede deel van de string. Vervolgens voert de PHP-pagina de query uit. Soms geeft de PHP-pagina een waarde terug, zoals wanneer een session of een sessionPart ingevoegd wordt. Hierbij geeft de PHP-pagina de primary keywaarde terug zodat de databaserecorder deze kan meegeven aan zijn toekomstige query's.

Om testsessies aan te maken heb ik een '*installGUI' functie* geïmplementeerd voor de recorder. Deze functie krijgt als parameter een div-element waarin de functie enkele HTML-elementen gaat zetten waardoor een testsessie kan worden aangemaakt op de pagina. Wanneer de gebruiker op de knop duwt gaat de recorderklasse de PHP-pagina aanroepen met een string waarin de gegevens voor de query staan. Deze string heeft gelijkaardige syntax als bovenaan besproken.



Figuur 5.9: Resultaat van de installGUI functie

Google Analyticsrecorder

Met de Google Analyticsrecorder gaan we bijhouden wat de gemiddelde bezoeker op de site doet. Op het einde van de dag worden alle acties verzameld door Google Analytics en de volgende dag zijn ze zichtbaar voor de eigenaar van de site. Het *'GoogleAnalyticsRecorder'*-object gaat in zijn functies telkens de _trackEvent functie aanroepen waardoor de events worden doorgestuurd naar de Google Analytics server. Een functie van het *'GoogleAnalyticsRecorder'* object ziet er dus als volgt uit:

```
GoogleAnalyticsRecorder.prototype.recordClickOnElement = function
(overlayEltID,videoTime) {
	_gaq.push(['_trackEvent', this.m_video, "Visitor clicked on element" , "ID:
"+overlayEltID, 1]);
};
```

Zoals eerder aangegeven worden events door Google Analytics geregistreeerd wanneer de _trackEvent functie wordt aangeroepen. In mijn implementatie registreren we de volgende events:

- een gebruiker start met het kijken van een film;
- een gebruiker klikt op een element;
- een gebruiker kijkt x aantal seconden van een film
- een gebruiker verlaat een film door op element met ID X te klikken;
- een gebruiker verlaat een film door de pagina af te sluiten;
- een gebruiker spoelt in de film door of terug.

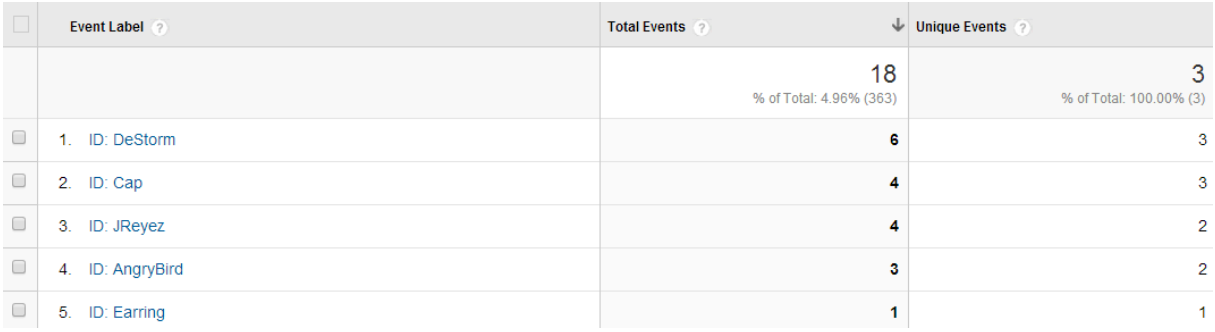
Een Google Analytics event heeft een category, action, label en opt_value. Als category gaan we altijd de film meegeven waar de actie in gebeurt. Als action geven we mee welke actie er gebeurd is. Voor het verlaten van een video via een element definiëren we een actie per element. Dit gebeurt omdat er als label het moment wordt meegeven waarop de gebruiker de video verlaat. Door de actions en category zo in te stellen krijgen we een splitsing van events per film (zoals te zien is in "Figuur 5.2: Google Analytics Event overzicht") waarbij we per film een overzicht kunnen genereren van de events voor die film (zoals in "Figuur 5.10:Google Analytics Events van 1 video ")

	Event Action	Total Events
		187 % of Total: 51.52% (363)
☐	1. Visitor watched x seconds of this video	142
☐	2. Visitor clicked on element	18
☐	3. Visitor started watching this video	10
☐	4. Visitor seeked Video	7
☐	5. Visitor left this video by clicking on element with ID: DeStorm	6
☐	6. Visitor left this video by clicking on element with ID: JReyez	4

Figuur 5.10:Google Analytics Events van 1 video

Per event gaan we soms ook nog het label en de value instellen zodat we per event nog een verdere onderverdeling kunnen zien.

- **Visitor clicked on element**: dit event wordt afgevuurd in de click eventhandler van een overlay-element in het '*VideoOverlayManager*'-object . Hierbij stellen we het label van het event in op het JavaScript ID van het overlay-element waarop geklikt is. De value wordt op 1 ingesteld. Hierdoor krijgen we het volgende overzicht waarop we kunnen zien hoeveel keer er in een film op een bepaald element is geklikt:



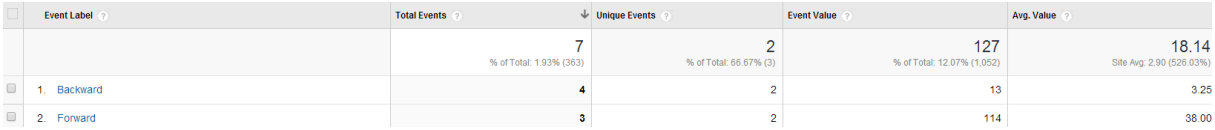
Event Label	Total Events ↓	Unique Events
	18 % of Total: 4.96% (363)	3 % of Total: 100.00% (3)
1. ID: DeStorm	6	3
2. ID: Cap	4	3
3. ID: JReyez	4	2
4. ID: AngryBird	3	2
5. ID: Earring	1	1

Figuur 5.11: Visitor clicked on element overzicht

In de bovenstaande figuur zien we hoeveel er geklikt is geweest op een bepaald element in tweede kolom. De derde kolom geeft weer door hoeveel unieke bezoekers dit event is uitgevoerd: bijvoorbeeld drie unieke bezoekers hebben in het totaal 6 keer op het overlay-element "DeStorm" geklikt. Een unieke bezoeker is een nieuwe bezoeker op de site. Indien de bezoeker 30 minuten niet meer actief is geweest op de site en erna terugkomt wordt hij herkend als een nieuwe unieke bezoeker, anders wordt hij herkend als dezelfde bezoeker.

- **Visitor started watching this video**: dit event wordt afgevuurd op het '*loadeddata*' van het HTML5 video-object. Dit event wordt telkens aangeroepen als een nieuw videobestand wordt ingeladen. Het event krijgt geen label en geen value.

- **Visitor seeked video**: dit event wordt afgevuurd in de 'clickOnBar' functie in het '*videoPlayer*'-object. Deze functie wordt aangeroepen wanneer een gebruiker het video playback tijdstip aanpast door middel van de videoplayer. Bij dit event zetten we het label op 'Backward' als de gebruiker terugspoelt en 'Forward' als de gebruiker doorspoelt. Als value geven we mee hoeveel seconden de gebruiker terug- of doorspoelt. Hierdoor krijgen we het volgende overzicht:



Event Label	Total Events ↓	Unique Events	Event Value	Avg. Value
	7 % of Total: 1.93% (363)	2 % of Total: 66.67% (3)	127 % of Total: 12.07% (1,052)	18.14 Site Avg: 2.90 (526.03%)
1. Backward	4	2	13	3.25
2. Forward	3	2	114	38.00

Figuur 5.12: Visitor seeked video overzicht

In deze video hebben gebruikers vier maal doorgespoeld en drie maal terug gespoeld. Er is in totaal 13 seconden terug- en 114 seconden doorgespoeld. In de laatste kolom is het gemiddeld aantal seconden te zien.

- **Visitor left this video by clicking on element with ID**: dit event wordt afgevuurd in de zelfgemaakte onclick interactionhandler van elementen die de video veranderen. Zoals dit bijvoorbeeld het geval is in de use case besproken in sectie 4.6. Voor dit type actie geven we de ID mee in de actie zodat er een actie wordt gemaakt per element. Als label geven we mee hoe ver in de film de gebruiker zat wanneer hij hierop klikte. Deze waarde wordt in % uitgedrukt. Hierdoor krijgen we een overzicht waarmee we kunnen zien wanneer gebruikers op elementen klikken:

Event Label	Total Events
	6 % of Total: 1.65% (363)
1. Left at 77% in video	3
2. Left at 76% in video	2
3. Left at 80% in video	1

Figuur 5.13: Visitor left this video by clicking on element with ID:id1 overzicht

We zien in bovenstaande figuur dat zes gebruikers via dit event de video hebben verlaten. Drie van de gebruikers deden dit 77% ver in de video. Twee deden dit 76% ver in de video en één deed dit 80% ver in de video.

- **Visitor left by leaving page**: dit event wordt afgevuurd in het *'window.onbeforeunload'* event. Dat event is echter onbetrouwbaar en de functie die we er aan koppelen wordt dus soms niet helemaal uitgevoerd of bepaalde request worden niet compleet afgehandeld. Het gaat hier bijvoorbeeld over het versturen van data naar de Google Analytics server.HTML biedt geen betrouwbaar event aan dat wordt afgevuurd wanneer een site wordt afgesloten. Dit heeft dus als nadeel dat de waarde van hoeveel mensen de video verlaten hebben door de site af te sluiten die we in het Google Analytics overzicht te zien krijgen, mogelijk niet compleet is. We kunnen echter wel de exacte waarde berekenen door te kijken naar hoeveel mensen gestart zijn de video te bekijken min het aantal mensen die de video verlaten hebben door op een element te klikken. Als label geven we aan dit event mee hoever de persoon was in de video voor hij de pagina afsloot. Deze waarde wordt ook uitgedrukt als een percentage van de totale speelduur van de video. Dit geeft ons een gelijkaardig overzicht als in "Figuur 5.13: Visitor left this video by clicking on element with ID:id1 overzicht".

- **Visitor watched x seconds of video**: dit event werd eerst afgevuurd in het *'window.onbeforeunload'* event en in de zelfgemaakte onclick interactionhandler van elementen die de video veranderen. Zoals dit bijvoorbeeld het geval is in de use case besproken in sectie 4.6. Als value werd het aantal seconden meegegeven dat de gebruiker naar de video had gekeken. Omdat het *'window.onbeforeunload'* event onbetrouwbaar is, viel er vaak een groot aantal seconden gekeken video weg uit het totaal. Dit was onaanvaardbaar. Daarom ben ik overgeschakeld naar een systeem van periodische events. Nu wordt het event verstuurd in de *'onAnimationStep'* functie van het *'VideoOverlayManager'* object . Hier wordt telkens gekeken of de gebruiker drie seconden video heeft gekeken sinds de vorige keer dat het event werd gevuurd, of sinds de start van

de video. Indien er drie seconden gekeken zijn, wordt het event opnieuw afgevuurd. Het event heeft geen label en krijgt als value 3. Het event wordt ook afgevuurd in het '*window.onbeforeunload'* event. Hier wordt berekend hoeveel seconden de gebruiker gekeken heeft sinds het vorige event werd afgevuurd, en dit aantal seconden wordt verstuurd als value bij het event. Door deze methode te gebruiken, hebben we een systeem dat in grote mate betrouwbaar kan weergeven hoelang een gebruiker naar een video gekeken heeft. De drie seconden is zo gekozen dat de 'event counter' niet te snel leeg zou geraken en tegelijk niet te veel gekeken seconden verloren kunnen gaan wanneer een gebruiker de site afsluit. De eigenaar kan nu in het event overzicht zien hoeveel seconden gebruikers in het totaal gekeken hebben naar een video.

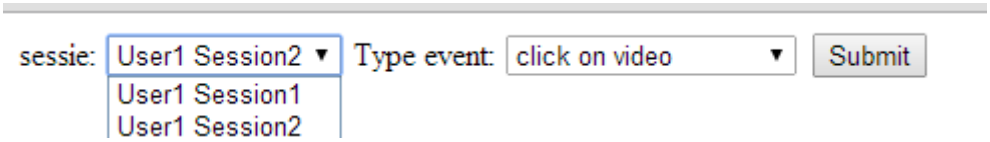
Event Action	Total Events ↓	Unique Events	Event Value
	187 % of Total: 51.52% (363)	3 % of Total: 100.00% (3)	558 % of Total: 53.04% (1,052)
1. Visitor watched x seconds of this video	**142**	3	413

Figuur 5.14: Visitor watched x seconds of this video

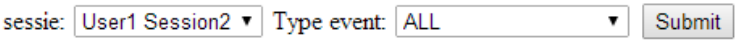
Op bovenstaande afbeelding zien we in de derde kolom hoeveel seconden er in totaal naar deze video bekeken zijn. Door deze waarde te delen door het aantal personen dat deze video gekeken heeft bekomen we het gemiddelde.

5.3 Resultaat

Ik heb de implementatie getest door verschillende keren de webpagina te bezoeken en acties uit te voeren op de use case besproken in sectie 4.6. De use case is zo aangepast dat alle overlay-elementen die niet verwijzen naar een andere video een pop-up tonen. De acties werden gerecorded door Google Analytics en sommige ervan eveneens door de databaserecorder. Voor het weergeven van de data in de MySQL-database heb ik een eenvoudige PHP-pagina aangemaakt waar gebruikers een testsessie kunnen kiezen en het type event. De pagina zal dan de gepaste data uit de database weergeven. Eigenaars van een site kunnen deze pagina dus gebruiken om de acties te zien die een specifieke gebruiker heeft uitgevoerd. Men kan ook zelf een complexere pagina maken waarmee men de data uit de database kan halen.



Figuur 5.15: Sessie selecteren met de testGUI



Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

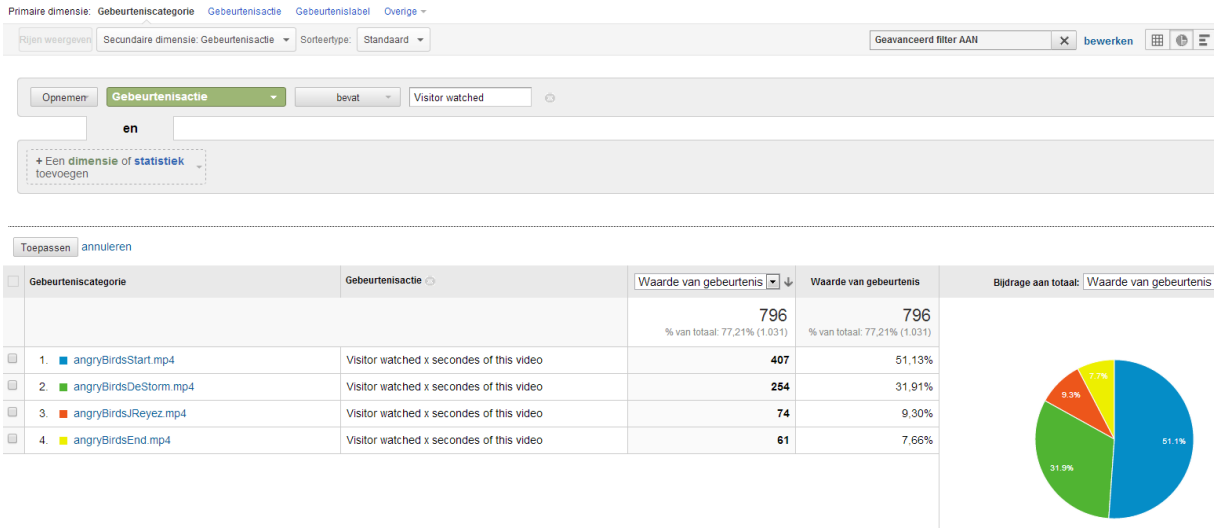
EventID	SessionPart	Video	EventType	VideoTime	RealTime
505	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	0.124584	2014-01-17 21:13:32.602
506	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	23.9614	2014-01-17 21:13:56.532
507	3	angryBirdsStart.mp4	clickOnVideo	23.9614	2014-01-17 21:13:58.109
508	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	23.9614	2014-01-17 21:13:59.957
509	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	49.096	2014-01-17 21:14:25.82
510	3	angryBirdsStart.mp4	clickOnElement	49.096	2014-01-17 21:14:32.716
511	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:14:58.15
512	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:15:00.689
513	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:15:02.434
514	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8947	2014-01-17 21:15:05.701
515	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:26.561
516	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:30.182
517	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:32.745
518	4	angryBirdsDeStorm.mp4	clickOnProgressBar	56.1564	2014-01-17 21:15:36.167
519	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	39.1271	2014-01-17 21:15:52.333
520	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	50.8203	2014-01-17 21:16:04.107
521	4	angryBirdsDeStorm.mp4	clickOnElement	50.8203	2014-01-17 21:16:06.227
522	5	angryBirdsEnd.mp4	clickOnVideoPlayerButton	10.2056	2014-01-17 21:16:16.664

Figuur 5.16: alle events van een session2 voor user1

De uitgevoerde acties, de verwachte resultaten van Google Analytics, de resultaten in Google Analytics en de resultaten in de database zijn te vinden in bijlage 1. We zien dat de verwachte resultaten overeenkomen met de resultaten die we zien in Google Analytics. Alleen bij de video “angryBirds Start” is er een afwijking van één seconde bij Visitor seeked video: Backward. Deze

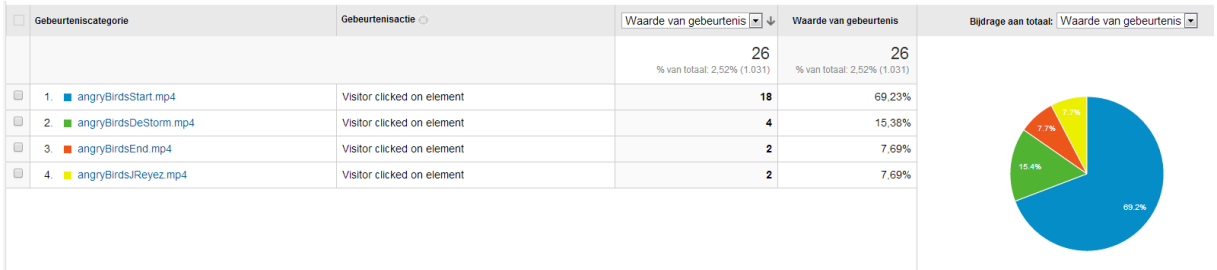
afwijking is te wijten aan een afronding die gebeurt in de code. De tijdstippen van de video die de gebruiker ziet op de videoplayer zijn afgerond, hoeveel seconden een gebruiker door- of terugspoelt wordt berekend uit de niet-afgeronde tijdstippen en wordt daarna afgerond voor het wordt verstuurd naar de GA server. We zien ook een afwijking bij het aantal gekeken seconden van “angryBirds End” en “angryBirds DeStorm”. Dit komt omdat de laatst bekeken seconden voor de gebruiker de pagina afsloot niet betrouwbaar konden verzonden worden.

Door verschillende filters toe te passen op de Google Analytics data kunnen we interessante grafieken genereren. Dit doen we door in het menu naar Gedrag->Gebeurtenissen->pagina’s te gaan. We selecteren hier de pagina die we willen waarvan we data willen zien, in dit geval de UserControlledNarrative pagina. Nu zien we de vier categorieën waar de events zijn in onderverdeeld, dat betekent dus de vier films waaruit deze pagina bestaat. We kunnen hier een secundaire dimensie instellen om gegevens weer te geven. We kiezen hier als secundaire dimensie de gebeurtenisacties. Vervolgens kunnen we filters plaatsen op de getoonde resultaten. Wanneer de filter (op actie veld) ingesteld wordt op “Visitor watched” krijgen we een overzicht van hoeveel seconden gebruikers naar iedere film gekeken hebben. Het resultaat is te zien in “Figuur 5.17”. Hier zien we in de derde kolom hoeveel seconden er gekeken is naar iedere film. We zien ook een taartdiagram dat de verhoudingen tussen de films weer geeft.



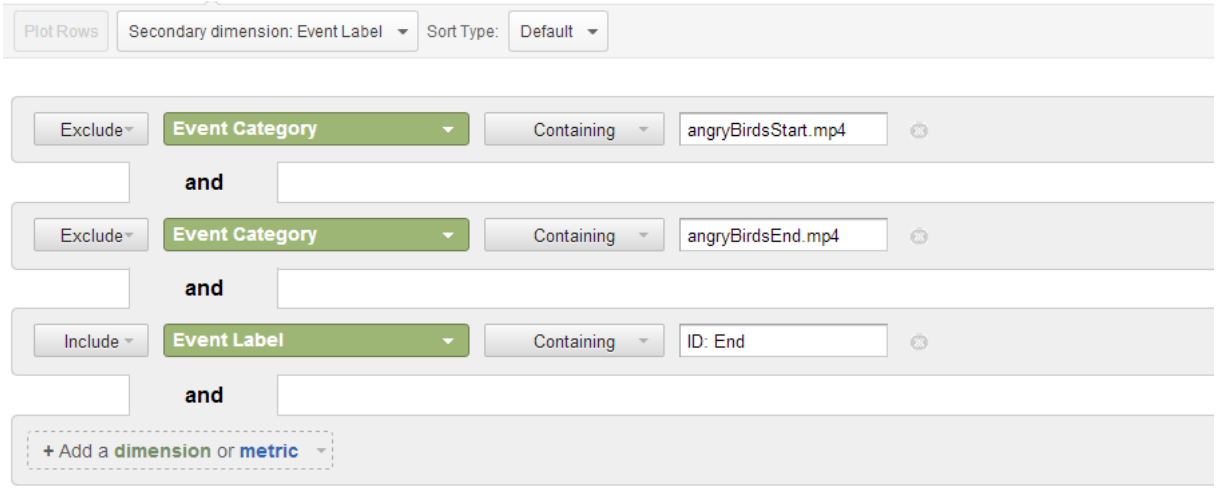
Figuur5.17: Overzicht seconden gekeken per film

Door de filter in te stellen als “Visitor clicked on element” krijgen we een overzicht van hoeveel keer er op een element is geklikt per film:



Figuur 5.18: Overzicht aantal keer op een element geklikt in een film

Deze grafiek kan gebruikt worden om te zien hoe interactief gebruikers zijn omgegaan met iedere film. In dit voorbeeld is dus te zien dat er het meeste geklikt is geweest op overlay-elementen in angryBirdsStart.mp4. Door iets complexere filters te maken kunnen we ook meer gerichte informatie krijgen. Een voorbeeld is het plaatsen van de volgende filter:



Figuur 5.19: angry bird filter per film

Met bovenstaande filter gaan we weergeven hoeveel keer gebruikers succesvol op het overlay-element met ID “End”hebben kunnen klikken in de films. In de filter hebben we de secundaire dimensie (de tweede kolom) ingesteld op het gebeurtenis label (event label), de eerste kolom geeft de category weer (film namen). Hierdoor krijgen we een overzicht van alle combinaties van label met category. We filteren de events van de films waar het overlay-element niet in voorkomt weg (dit overlay-element bevindt zich alleen in angryBirdsJReyez en angryBirdsDeStorm). Vervolgens stellen we het ID van het overlay-element (dit is voor beide films hetzelfde) in als de filter. Hierdoor krijgen we het volgende overzicht:

Figuur 5.20: Overzicht kliks op angry bird

We kunnen deze filter functionaliteit op verscheidene manieren gebruiken. We zouden als secundaire dimensie bijvoorbeeld het land waar de gebruiker zich in bevindt kunnen meegeven; hierdoor zouden we met de correcte filters kunnen zien of bepaalde overlay-elementen vaker worden aangeklikt in bepaalde landen. We kunnen bijvoorbeeld als secundaire parameter het uur van de dag meegeven, waardoor we een overzicht kunnen genereren welke events zijn gebeurd per uur.

Met Google Analytics kan een eigenaar van een site dus een duidelijk overzicht krijgen van wat gebruikers doen op de site, welke video's populairder zijn en welke elementen het meest aangeklikt worden. Dit zal makers van interactieve video's helpen met het creëren van nieuwe interactieve video's aangezien ze duidelijk kunnen zien uit de vorige video's wat populair is en wat niet. Indien een eigenaar specifiekere info wil krijgen over bepaalde zaken in zijn films, zal hij testsessies moeten organizeren. Hierbij kan hij dan bijvoorbeeld zien of gebruikers vaak naast bepaalde overlay-elementen klikken of dat gebruikers vaak op pauze moeten klikken voor ze een bepaald element kunnen aanklikken. Deze testen zullen echter een grotere kost hebben aangezien men mensen moet uitnodigen, testers inschakelen en desnoods nog extra test- of analysesoftware moet aanschaffen (zoals Morae of een statistiekverwerkingsprogramma).

6 Interactieve omnidirectionele video

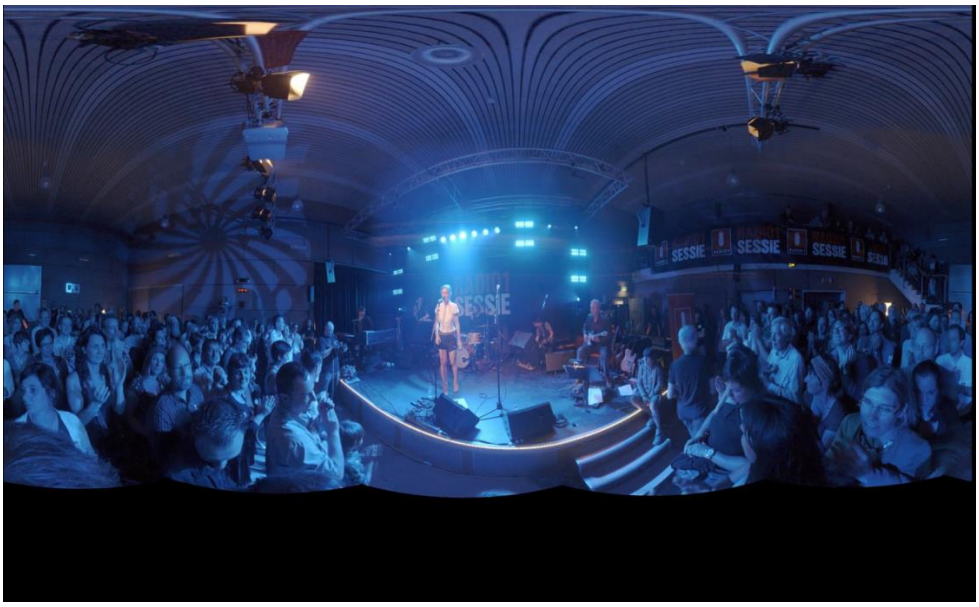
In dit deel van de masterthesis gaan we overlay-elementen combineren met een onmindirectionele video. Een onmindirectionele video (ODV) is een video opgenomen met een camera die alles in een cirkel van 360 graden rondom zich kan filmen. Wanneer gebruikers naar een ODV kijken dan krijgen ze vaak maar een klein deel van de video te zien. Ze kunnen zelf bepalen naar waar ze kijken. Hierdoor krijgen gebruikers de indruk dat ze op de plaats van de camera staan en alles rondom hen zien gebeuren. Gebruikers kunnen ook in- en uitzoomen op de video. In dit deel van de masterthesis ga ik overlay-elementen toevoegen aan een ODV. De kijkpositie en schaal van de video gaan hier invloed hebben op de overlay-elementen. Zo moeten overlay-elementen ook meeschalen met de video en overlay-elementen die buiten het zichtveld van de gebruiker liggen mogen niet zichtbaar zijn.

6.1 Implementatie

ODV videoplayer

Voor dit onderdeel van de masterthesis werken we met de ODV player gemaakt door dr. Jori Liesenborgs.

De ODV player werkt met videobestanden zoals getoond in “Figuur 6.1”. Dit is een 360 graden opname die vervolgens platgedrukt werd in een 2D-vlak. De ODV player gaat de gebruiker een viewwindow geven waardoor de gebruiker naar de video kan kijken. Dit viewwindow gaat de gebruiker echter maar een gedeelte van de video laten zien, een voorbeeld hiervan is “Figuur 6.2”. De gebruiker kan andere delen van de video bekijken door met de muis op de viewwindow te klikken en met de muis te slepen naar de tegenovergestelde richting van waar hij naar toe wil. De gebruiker kan ook in- en uitzoomen waardoor hij respectievelijk minder en meer van de video krijgt te zien. De ODV player gaat het linker en het rechter uiteinde van de video aan elkaar plakken waardoor de gebruiker de indruk krijgt dat de video cilindrisch is. De boven- en de onderkant van de video worden niet aan elkaar geplakt. Wanneer een gebruiker naar boven (of onder) beweegt in de video en hij het hoogste (of laagste) punt bereikt, dan blijft de viewwindow op dat punt staan. In “Figuur 6.1” zien we links en rechts een gedeelte van eenzelfde deur. In “Figuur 6.3” zien we wat er gebeurt wanneer een gebruiker zijn viewwindow positioneert op de rand van de ODV. De twee eindes van de video zijn zo aan elkaar geplakt dat de gebruiker niet merkt dat het videobestand daar normaal zou eindigen.



Figuur 6.6: ODV in normale videoplayer

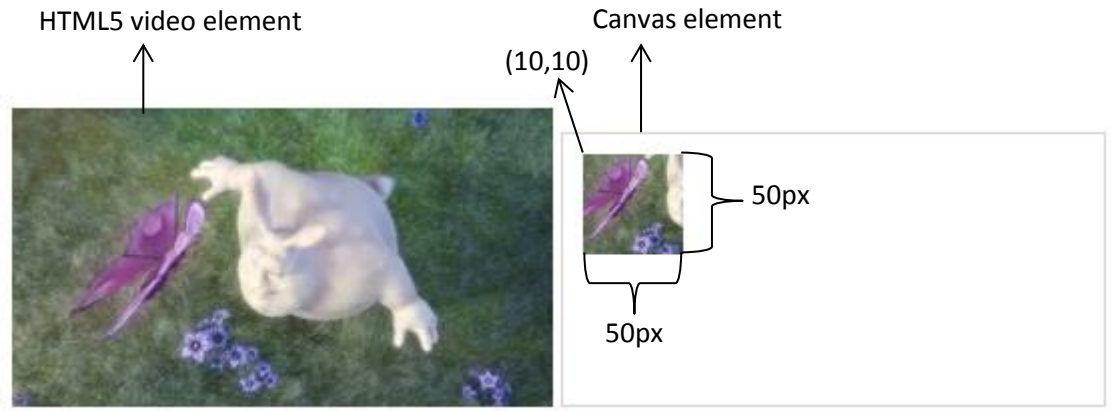
Figuur 6.7: Viewwindow van ODV player

Figuur 6.8: Viewwindow op rand van video

De ODV player bestaat uit een onzichtbaar HTML5 video element en een zichtbaar canvas element. Get canvas element is de viewwindow waardoor we naar de video kijken. We kunnen op een canvas (een gedeelte van) de huidige frame van een HTML5 video tekenen. Door dit herhaaldelijk aan een snel tempo te doen, krijgen we een video op het canvas element. De ODV player doet dit met een updatefunctie. Het tekenen van een videoframe gebeurt door middel van de volgende code:

```
var v=document.getElementById("video1");
var c=document.getElementById("myCanvas");
ctx=c.getContext('2d');
ctx.drawImage(v, 50,50,100,100,10,10,50,50);
```

Afbeelding “Figuur 6.4” geeft het resultaat weer van bovenstaande code. Het 100 op 100 vierkant dat we gaan kopiëren uit de video wordt gerescaled naar een 50 op 50 vierkant op de canvas.



Figuur 6.9: Video op canvas tekenen

De drawImage[32] parameters zijn als volgt:

1. Image, canvas of video element waarvan de imagedata afkomstig is (in dit geval een video)
2. X-coördinaat in de video waar clipping start (dus x-coördinaat in video)
3. Y-coördinaat in de video waar clipping start (dus y-coördinaat in video)
4. Breedte van geclipte afbeelding in de video
5. Hoogte van geclipte afbeelding in de video
6. X-coördinaat waar de afbeelding getekend moet worden op het canvas
7. Y-coördinaat waar de afbeelding getekend moet worden op het canvas
8. Breedte van de afbeelding op het canvas
9. Hoogte van de afbeelding op het canvas

De ODV videoplayer gaat om de tien milliseconden kijken waar de viewwindow staat in de video (dit gebeurt in de updatefunctie). Naargelang de positie en het zoomlevel van de viewwindow gaat de ODV videoplayer het gepaste deel uit de video tekenen op het canvas. Wanneer de viewwindow op de linker of rechter rand staat van een video, gaat de ODV player twee plaatsen in de video moeten tekenen op het canvas. Dat is het geval in “Figuur 6.3”. Hier moet de ODV player het linker deel van het canvas opvullen met het gepaste rechter deel van de video, en het rechter deel van het canvas met het gepaste linkerdeel van de video. De onzichtbare lijn tussen de twee getekende delen van de video op het canvas noemen we de breakline.

De ODV player voert de volgende stappen uit om aan het gewenste resultaat te komen:

1. Bereken het videocoördinaat van het linker bovenhoekpunt van de viewwindow (x1,y1).
2. Bereken de breedte en hoogte van het videodeel dat getekend moet worden op het canvas Deze waardes wordt uitgedrukt in aantal videopixels, dus niet de breedte en hoogte van het canvas (breedte1,hoogte1).
3. Kijk of x1+breedte1 groter is dan de breedte van de video.

NEE:

4. Gebruik de drawImage functie om op het canvas te tekenen als volgt:

ctx.drawImage(videoElement, x1,y1, breedte1, hoogte1,0,0,canvasBreedte, canvasHoogte)

JA:

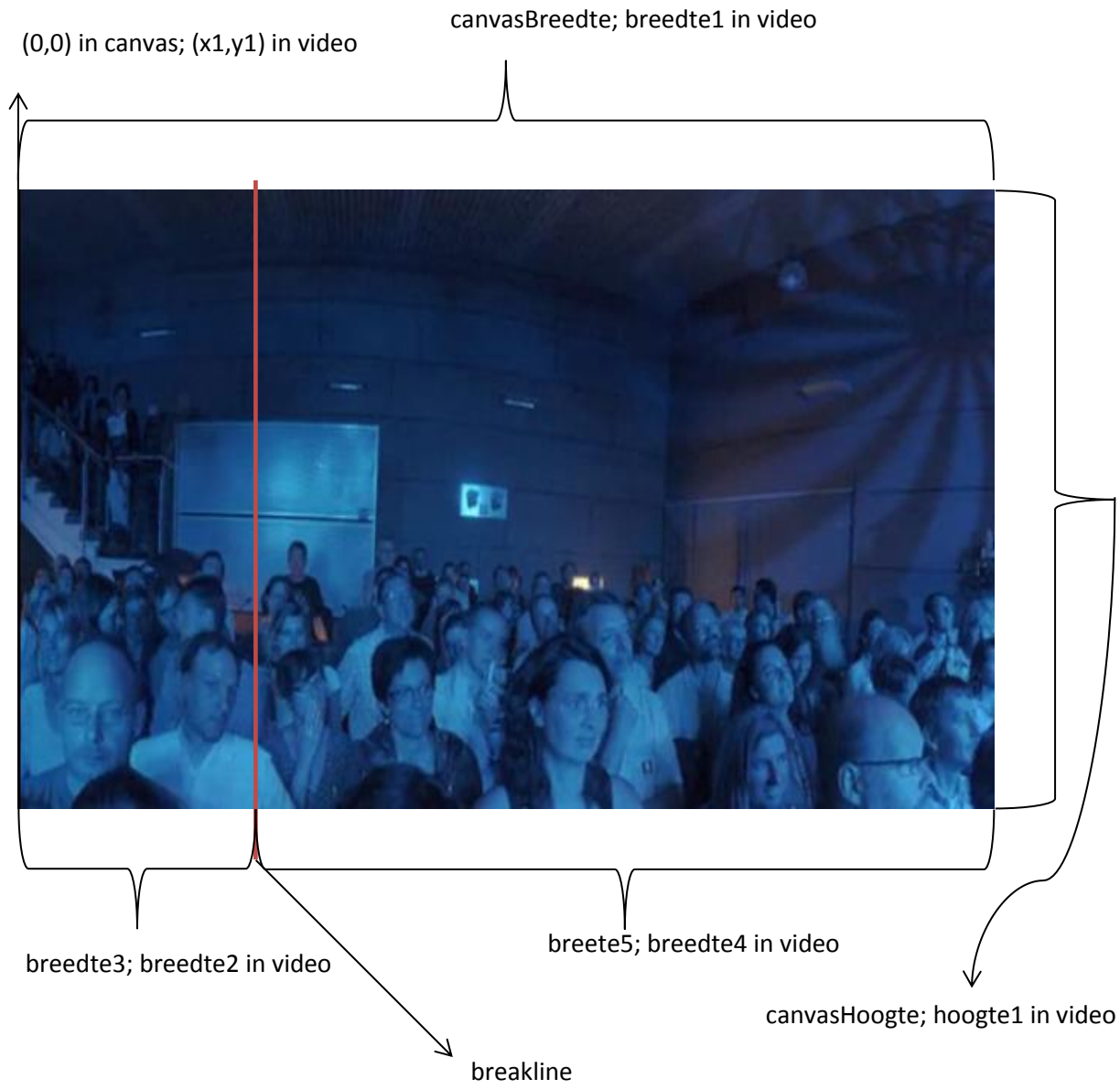
4. Bereken hoeveel videopixels er zijn tussen x1 en het einde van de video (breedte2).
5. Bereken hoeveel pixels dit zijn op het canvas (breedte3).
6. Gebruik de drawImagefunctie om dit gedeelte te tekenen op het canvas als volgt:

ctx.drawImage(videoElement, x1,y1, breedte2, hoogte1,0,0, breedte3, canvasHoogte);

7. Bereken hoeveel videopixels in de breedte nog moeten getekend worden (breedte4).
8. Bereken hoeveel pixels dit zijn op het canvas (breedte5).
9. Gebruik de drawImagefunctie om dit gedeelte te tekenen op het canvas als volgt:

ctx.drawImage(videoElement,0,y1, breedte4, hoogte1, breedte3,0, breedte5, canvasHoogte);

Op onderstaande figuur staan al de breedtes en punten voor scenario's waarin een breakline voorkomt aangeduid:



Figuur 6.10: Punten en breedtes van ODV player

Het zoomen van het viewwindow gebeurt door grotere of kleinere delen uit de video te halen en deze te tekenen op het canvas. Deze blijft zijn grootte behouden, enkel de grootte van het deel dat uit de video getekend wordt, verandert. De drawImage functie gaat het getekend deel dan rescalen zoals ook gebeurt in "Figuur 6.4".

Overlay-elementen correct weergeven in een ODV

Voor het correct weergeven van overlay-elementen in een ODV moeten we met vier dingen rekening houden. Ten eerste moet de '*VideoOverlayManager*' zich bewust zijn van de positionering van het viewwindow tegenover de video. Overlay-elementen die volledig buiten de viewwindow liggen mogen niet zichtbaar zijn. Vervolgens moet er voor gezorgd worden dat overlay-elementen die gedeeltelijk zichtbaar zijn geclipt worden zodat alleen het gedeelte in de viewwindow zichtbaar is.

Ook kan de ODV player in- en uitzoomen. Hierdoor moeten ook de overlay-elementen mee geschaald worden. Als laatste moet er rekening gehouden worden met het feit dat de overlay-elementen niet de basisfunctionaliteit van de ODV player mogen verstoren.

Viewwindow in de VideoOverlayManager

Als eerste heb ik een nieuwe overlayManager klasse gedefinieerd '*ODVVideoOverlayManager'* die overerft van de '*VideoOverlayManager'* klasse. De '*ODVVideoOverlayManager'* moet zich ervan bewust zijn dat de video niet getoond wordt in het video element. Daarom krijgt de manager een extra membervariabele die verwijst naar het element dat instaat voor de visualisatie van de video, in dit geval het canvas element. Zo kan de manager de overlay-elementen correct positioneren ten opzichte van het canvas.

Zoals eerder vermeld, werkt de ODV player met een updatefunctie die om de tien milliseconden wordt aangeroepen. De '*VideoOverlayManager'* klasse vuurt normaal gezien zijn '*onAnimationStep'* functie ook om de tien milliseconden af, maar dit is nu niet meer zo. Nadat de ODV player in de updatefunctie alles heeft getekend, wordt de '*onAnimationStep*' functie aangeroepen.

De '*ODVVideoOverlayManager'* moet zich bewust zijn van de viewwindow. Daarom voegen we extra membervariabelen toe aan de manager die informatie bijhouden over de viewwindow. Zo gaat de manager de volgende dingen bijhouden:

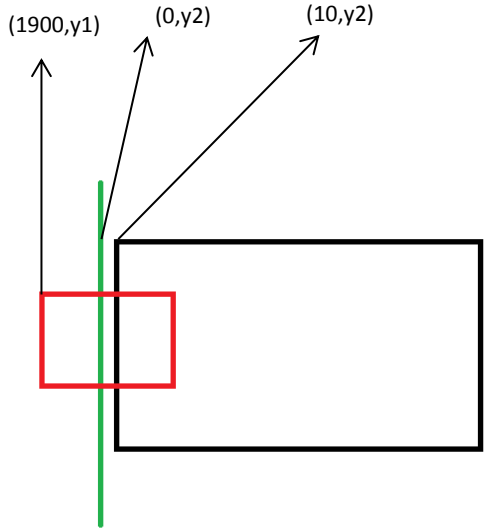
1. De videocoördinaten van de linkerbovenhoek van de viewwindow
2. De videobreedte en hoogte van de viewwindow
3. De echte breedte en hoogte van de viewwindow (dus de breedte en hoogte van het canvas)
4. Waar in de viewwindow de breakline staat, indien deze aanwezig is

Deze 'ODVVideoOverlayManager' krijgt de functies '*setViewWindowPart'* en '*ResetViewWindow*'. Met deze functies wordt de manager zich bewust van de viewwindow. De ODV player roept in zijn updatefunctie eerst de '*ResetViewWindow'*-functie aan. Hierdoor wist de manager de vorige windowgegevens. Meteen na elke drawImage functie aanroep wordt in de ODV player de '*setViewWindowPart'*-functie aangeroepen. Aan deze functie worden de volgende parameters meegegeven:

1. Videocoördinaten van de linkerbovenhoek van het getekend gedeelte
2. Videohoogte en breedte van het getekend gedeelte
3. Canvashoogte en -breedte van het getekend gedeelte

Indien de breakline niet aanwezig is, zal de '*setViewWindowPart'*-functie dus maar één keer worden aangeroepen. De waarde van de breakline member variabele is dan -1. Indien de breakline wel aanwezig is,zal de '*setViewWindowPart'* functie tweemaal worden aangeroepen: de eerste keer na het tekenen van het linkerdeel en de tweede keer na het tekenen van het rechterdeel. De tweede keer dat de functie wordt aangeroepen, wordt de videobreedte van de eerste aanroep ingesteld als de breakline member variabele. De video- en canvasbreedte worden opgeteld bij die van de eerste aanroep om zo de totale breedte te bekomen. Het eerst ingestelde coördinaat van de linkerbovenhoek blijft ongewijzigd.

Hierna zijn dus al de membervariabele van de '*ODVVideoOverlayManager'* ingevuld. Vervolgens gaat er in de updatefunctie van de ODV player de 'onAnimationStep' functie van de manager worden aangeroepen. De manager gaat, zoals bij normale video, weer kijken of een element zichtbaar moet zijn (volgens de start- en stoptijd van het element). Vervolgens wordt de '*positionOverlayElt'* functie aangeroepen om het overlay-element te positioneren. Indien de '*positionOverlayElt'* functie merkt dat het overlay-element helemaal buiten de viewwindow ligt, dan geeft deze 'false' terug, anders 'true'. Als de functie 'true' heeft teruggegeven, wordt het overlay-element zichtbaar gemaakt, anders niet.



Figuur 6.6: Situatie 2, geen breakline

In de '*positionOverlayElt'* functie wordt zoals bij de normale '*VideoOverlayManager'* eerst de animatie engine aangeroepen om de positie van het overlay-element te bepalen in de video. Vervolgens gaat de '*ODVVideoOverlayManager'* kijken of het overlay-element in de viewwindow ligt. Hierbij gaat de manager moeten rekening houden met het al dan niet aanwezig zijn van de breakline.

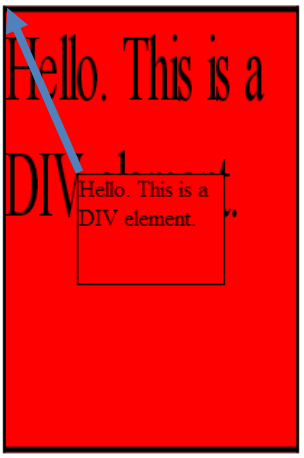
Indien de breakline niet zichtbaar is, dan gaat de manager kijken of het overlay-element in de viewwindow ligt. Dit gaat de manager doen met behulp van de '*squaresOverlap'* functie die kijkt of twee vierhoeken in elkaar liggen of elkaar gedeeltelijk overlappen. Hierbij moet er wel rekening gehouden worden met een speciaal geval, namelijk dat van "Figuur 6.6". In deze figuur is de zwarte vierhoek de viewwindow. De rode vierhoek is een overlay-element en de groene lijn is de breakline. De video heeft een breedte van 1920 pixels, de viewwindow een breedte van 300 pixels. Het overlay-element dat 100 pixels breed is zou dus moeten getekend worden maar aangezien we hier met een 2D-coördinatenstelsel werken (dat van de video) gaat de '*squaresOverlap'* functie zeggen dat deze twee vierhoeken elkaar niet overlappen. Dit is zo omdat een vierhoek op positie (1900,y1) met breedte 100 niet overlapt met een vierhoek op positie (10,y2) met breedte 300. Om dit op te lossen, wordt de '*squaresOverlap'* functie nog een tweede keer uitgevoerd maar dan wordt het x-coördinaat van het element aangepast naar (1900 - videobreedte). Indien de functie de tweede keer ook zegt dat het element en het window elkaar niet overlappen, gaat de '*positionOverlayElt'* functie 'false' teruggeven, waardoor het element niet zal worden weergegeven.

Indien de breakline wel zichtbaar is, gaat de manager kijken of het overlay-element zichtbaar is in het linker- of rechterdeel van de viewwindow (links of rechts van de breakline). Eerst gaat er gekeken

worden of het overlay-element in het linkerdeel ligt met behulp van de '*squaresOverlap'* functie. Indien het overlay-element niet in het linkerdeel ligt, dan gaan we kijken of het in het rechterdeel ligt. Als het overlay-element niet in één van de twee delen voorkomt, is het niet zichtbaar en gaat de '*positionOverlayElt'* functie 'false' terug geven.

Vervolgens gaat er gekeken worden naar het zoomlevel van de viewwindow. Overlay-elementen moeten ongeacht het zoomlevel op dezelfde positie in de video staan. Ook moet de grootte van het overlay-element evenredig met het zoomlevel worden aangepast. De schaal wordt zowel in horizontale als verticale richting berekend aan de hand van de hoogte en breedte van het canvas en de hoogte en breedte van het video gedeelte getoond in het canvas. Het schalen van de overlay-elementen gebeurt met de volgende css-functies:

```
transform:scale(2,4);
-ms-transform:scale(2,4); /* IE 9 */
-webkit-transform:scale(2,4); /* Safari and Chrome */
```



Figuur 6.7: Voorbeeld van het effect van de transform: scale css property

Het resultaat van deze functies is te zien in "Figuur 6.7", de functies zijn niet hetzelfde voor alle browsers, elke browser heeft zijn eigen syntax voor deze functie. De eerste parameter is het horizontale zoomlevel, de tweede parameter het verticale zoomlevel. Beide div elementen hebben dezelfde absolute positie. De schaal-functie gaat er dus voor zorgen dat het div element ook van plaats verandert (aangezien de linkerbovenhoek van beide div's niet meer hetzelfde is). Dit effect gaan we tegengaan door de verplaatste afstand te berekenen (aangeduid met de blauwe pijl). Deze afstand moeten zowel voor de verticale als horizontale as berekend worden. We noemen deze waardes (de horizontale en verticale afstand) de scalePositionCompensation waardes. Ze worden als volgt berekend:

```
horizontalScalePositionCompensation = -(parseInt(domVidOverlayElt.style.width)/2-
horizontalScale*parseInt(domVidOverlayElt.style.width)/2);
verticalScalePositionCompensation = -(parseInt(domVidOverlayElt.style.height)/2-
verticalScale*parseInt(domVidOverlayElt.style.height)/2);
```

Figuur 6.8: Overlay-element zonder scalePositionCompensation

In bovenstaande figuur zien we wat er gebeurt met de positionering wanneer de scale compensation niet aanwezig is. Het rechtse overlay-element is een ingezoomde versie van het linkse. Het element heeft een vaste positie en hoort niet te bewegen. Toch zien we dat door het inzoomen de positie van het overlay-element tegenover de video veranderd is.

Nu de scalePositionCompensation waardes berekend zijn kunnen we het overlay-element positioneren. Het verticale coördinaat van een overlay-element zal altijd hetzelfde berekend worden. Het horizontale coördinaat wordt als volgt berekend:

Legende code:

domVidOverlayElt: het element dat we gaan positioneren
newPos: de berekende positie van het overlay-element in de video
TopVideoCorOfWindow: videocoördinaat in de linkerbovenhoek van de viewwindow
horizontalScale, verticalScale: het zoomlevel van het viewwindow respectievelijk horizontale en verticale richting
horizontalScalePositionCompensation, verticaltalScalePositionCompensation: de ScalePositionCompensation waarden

Geen breakline in viewwindow :

Normaal element:

```
domVidOverlayElt.style.left = parseInt(posVidScreen.left)   +
(Math.round(newPos.x)- this.m_TopVideoCorOfWindow.x)* horizontalScale +
horizontalScalePositionCompensation +"px";
```

Element in de situatie van “Figuur 6. 6”

```
domVidOverlayElt.style.left = parseInt(posVidScreen.left)   + (Math.round(newPos.x
- this.m_videoWidth)- this.m_TopVideoCorOfWindow.x) * horizontalScale +
horizontalScalePositionCompensation +"px";
```

Breakline in viewwindow :

Element links van breakline:

```
domVidOverlayElt.style.left = parseInt(posVidScreen.left)   +
(Math.round(newPos.x)- this.m_TopVideoCorOfWindow.x)* horizontalScale +
horizontalScalePositionCompensation +"px";
```

Element rechts van breakline:

```
domVidOverlayElt.style.left = parseInt(posVidScreen.left) + (this.m_breakLine +
Math.round(newPos.x))* horizontalScale + horizontalScalePositionCompensation+"px";
```

Verticale positie :

```
domVidOverlayElt.style.top = parseInt(posVidScreen.top)  + (Math.round(newPos.y)-
this.m_TopVideoCorOfWindow.y) * verticalScale + verticalScalePositionCompensation
+"px";
```

Nu het overlay-element correct gepositioneerd en geschaald is, moet het enkel nog geclipt worden. Het clippen van HTML elementen gebeurt met de volgende functie[33]:

```
Element.style.clip="rect(top right bottom left)";
```

De vier parameters die aan de functie moeten worden meegegeven zijn:

1. Top: het aantal pixels dat boven van het element moet worden weggeclipt
2. Right: alle pixels rechts van deze waarde worden weggeclipt
3. Bottom: alle pixels onder deze waarde worden weggeclipt
4. Left: het aantal pixels dat links van het element moeten worden weggeclipt

In “Figuur 6.9” zien we het resultaat van onderstaande code op een afbeelding met 140pixel hoogte en 100 pixels breedte.

clip:rect(10px,60px,140px,0px);

Figuur 6.9: Resultaat clip functie

Om te berekenen hoeveel van het element moet weggeclipt worden, moet eerst de afstand worden berekend tussen het overlay-element en de randen van de viewwindow. Deze afstanden worden berekend in het coördinatenstelsel van het scherm. We gebruiken de jQuery *‘offset’*[34] (deze geeft de positie terug van het overlay-element op het scherm*)* en *‘getBoundingClientRect’*[35] (deze geeft de grootte van het overlay-element op het scherm terug, inclusief borders)functies om exacte positie

en de grootte van het overlay-element op te vragen. Zo wordt bijvoorbeeld de afstand tussen het element en de bovenkant en onderkant van de viewwindow berekend door[1]:

```
var topDistance = posVidScreen.top - parseFloat($(domVidOverlayElt).offset().top);

var botDistance = (parseFloat($(domVidOverlayElt).offset().top)+
parseFloat(domVidOverlayElt.getBoundingClientRect().height))  -
(parseFloat(posVidScreen.top)+ this.m_actualWindowHeight);
```

Bij het clippen moet er rekening gehouden worden met een mogelijk ingestelde border op het overlay-element. Bovenstaande code houdt rekening met de borders van een overlay-element. De afstand die berekend wordt is dus inclusief de borderbreedte. Wanneer we van een element de style.height opvragen is dit exclusief de borderbreedte.

Vervolgens kan de manager nu berekenen hoeveel er van de figuur moet afgeclipt worden. Dit gebeurt met de volgende code:

```
if(topDistance > 0){
      topClip =topDistance*1/verticalScale;
}

botClip=parseFloat(domVidOverlayElt.style.height)+borderwidth*2;
if(botDistance > 0){
            botClip = (parseFloat(domVidOverlayElt.style.height) +2*borderwidth -
            botDistance*1/verticalScale) ;
}
```

We moeten hier de afstand, die tegelijkertijd ook het aantal weg te clippen pixels is, vermenigvuldigen met 1/verticalScale omdat de clipfunctie de scale transformatie negeert. De clipfunctie moet dus waarden krijgen die overeen komen met de originele grootte van het element. De clip functie houdt wel rekening met de border van het element. Daarom moeten we hier dus twee keer de border breedte optellen bij de hoogte van het element. Het clippen van het overlay-element in de horizontale richting gebeurt analoog aan de getoonde code.

Alles wat besproken werd in bovenstaande tekst, resulteert in een correcte weergave van overlay-elementen bovenop een ODV player. Een probleem dat nu nog optreedt is dat de overlay-elementen storen bij het bewegen van de viewwindow in de ODV. Het bewegen van de viewwindow gebeurt door te klikken op de ODV en dan te slepen met de muis (mousemove event). Wanneer een gebruiker op een overlay-element klikt en vervolgens een sleepbeweging uitvoert met de muis, mag de viewwindow van de ODV video niet bewegen. Dit gebeurt automatisch wegens de plaatsing van de overlay-elementen op het canvas. Wanneer de muis op een overlay-element staat, dan krijgt het overlay-element de click en mouseover events binnen en niet het canvas zolang de muis binnen het overlay-element blijft. Indien de muis buiten het overlay-element beweegt dan vangt het canvas element de mouseover events op. Deze worden echter genegeerd aangezien er geen onclick was op het canvas. Wanneer een gebruiker echter klikt en sleept op het canvaselement en met zijn muis over een overlay-element komt, dan mag het overlay-element geen click of mouseover events binnenkrijgen, deze moeten blijven binnenkomen bij het canvaselement. Als dit niet gebeurt, stopt

[1] De $ in de code is jQuery syntax.

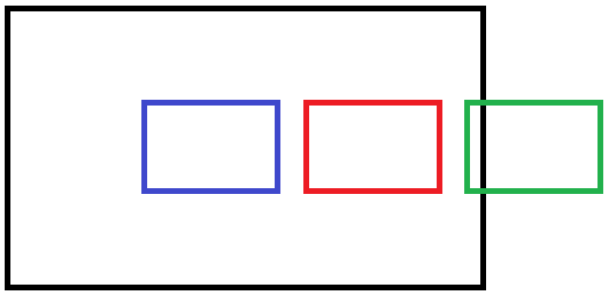
de beweging van de viewwindow abrupt. Om dit tegen te gaan gaat de ODV player de '*disableElementsMouseEvents'* functie aanroepen van de manager bij een mousedown event op het canvas. Deze functie gaat al de mouse events op al de overlay-elementen uitschakelen. Hiervoor wordt volgende code gebruikt:

```
vidOverlayElt.style.pointerEvents = "none";
```

Bij mouseleave of mouseup events op het canvas element gaat de '*enableElementsMouseEvents'* functie aangeroepen worden. Deze gaat de mouse events op de overlay-elementen terug inschakelen. Dankzij deze functies storen de overlay-elementen nu niet meer bij het bewegen van de viewwindow in de ODV player.

Reactie-overlay-elementen

Een extra feature die toegevoegd werd is de mogelijkheid om reactie-overlay-elementen te definiëren. Deze overlay-elementen hebben de eigenschap dat ze pas zichtbaar worden als er op een bepaald overlay-element wordt geklikt (zoals te zien is in “Figuur 6.11” en “Figuur 6.12”). Het element waarop een reactie-overlay-element een reactie is, wordt voortaan het bronelement genoemd. De positie van het reactie-overlay-element wordt bepaald door deze van het bronelement. Het reactie-overlay-element zal altijd een vaste positie hebben ten opzichten van het bronelement. Reactie-overlay-elementen schalen niet mee met hun bronelementen en worden niet geclipt. In plaats daarvan worden reactie-overlay-elementen geplaatst waar plaats is in de viewwindow. Dat wil zeggen dat als er plaats is rechts van het bronelement, het reactie-overlay-element dan rechts geplaatst word, is er rechts geen plaats, dan wordt het links geplaatst. Een voorbeeld hiervan is “Figuur 6.10” waar het rode element het bronelement is. Het groene element is het reactie-overlay-element als het rechts zou geplaatst worden. We zien hier echter dat het element zou moeten geclipt worden. Aangezien reactie-overlay-elementen niet geclipt worden, gaat het een andere positie krijgen. Het reactie-overlay-element wordt dus links geplaatst. Dus het krijgt de positie van het blauwe element.



Figuur 6.10: Positionering reactie element

Figuur 6.11: Niet aangeklikt overlay-element

Figuur 6.12: Aangeklikt overlay-element met reactie-overlay-element

De reactie-overlay-elementen hebben de eigenschap dat ze kunnen weergegeven worden als een call-out box, zie “Figuur 6.13”. Hierbij wijst het uiteinde van de call-out box altijd naar het bronelement.

Figuur 6.13: Reactie-overlay-element met call-out box stijl

Definiëren reactie-overlay-elementen

Reactie-overlay-elementen worden gedefinieerd zoals andere overlay-elementen in een voe-def bestand. In het voe-def bestand wordt een nieuwe array, ‘*g_voeDefsReaction’,* aangemaakt voor de

definitie van reactie-overlay-elementen. De definitie van een reactie-overlay-element ziet er als volgt uit:

```
g_voeDefsReaction.push( [ "urId1", "uID7", 10,"L LA LB",true,"Inhoud", "height:
50px; width:70px;",true ] );
```

De parameters die moeten meegegeven worden zijn:

1. Het ID van het reactie-overlay-element.
2. Het ID van het bronelement.
3. De afstand tussen het reactie-overlay-element en het bronelement uitgedrukt in pixels. Deze afstand is een afstand gemeten in schermcoördinaten en niet videocoördinaten. De afstand schaalt dus ook niet mee als de video wordt in- of uitgezoomd.
4. Een lijst van mogelijke posities tegenover het bronelement waar het reactie-overlay-element mag komen te staan. De mogelijkheden zijn: links (L), rechts (R), boven (A), onder (B), linksboven (LA), linksonder (LB), rechtsboven (RA) en rechtsonder (RB). De volgorde van posities die gespecificeerd wordt is van belang. De animatie engine zal deze lijst af gaan en kijken of het element kan geplaatst worden op de eerste positie, zo niet gaat het door naar de volgende positie, enzv. Het reactie-overlay-element komt niet te staan op plaatsen die niet in de lijst voor komen.
5. Een boolean waarde die bepaalt of het element zijn huidige plaats moet behouden wanneer er een plaats met hogere prioriteit vrij komt. Neem bij wijze van voorbeeld een reactie-overlay-element met als vierde parameter “L R”. Indien het reactie-overlay-element niet links kan geplaatst worden, wordt het rechts geplaatst. De gebruiker beweegt met de viewwindow en er komt genoeg plaats vrij links van het bronelement om het reactie-overlay-element daar te plaatsen (er is ook nog genoeg plaats rechts van het element om het daar te plaatsen). Als deze boolean parameter op ‘false’ staat, wordt het reactie-overlay-element links geplaatst van het moment dat er rechts weer plaats is. Als deze variabele ‘true’ is dan behoudt het reactie-overlay-element zijn plaats tot dat er rechts van het bronelement geen plaats meer is. We noemen deze waarde de ‘*keepPosition’* waarde.
6. De inhoud van het reactie-overlay-element.
7. De opmaak van het reactie-overlay-element. Hierin is de gebruiker verplicht een breedte en hoogte in te vullen.
8. Een boolean waarde om aan te geven dat dit reactie-overlay-element een call-out box reactie-overlay-element moet zijn.

Aanmaken reactie-overlay-elementen

Reactie-overlay-elementen worden altijd pas als laatste aangemaakt in de code. Dit komt omdat de elementen een reactie zijn op een bronelement en dit bronelement moet dus eerst aangemaakt worden. Voor het aanmaken van reactie-overlay-elementen krijgt de ‘*VideoOverlayEltFactory’* klasse een nieuwe functie. Het aanmaken van een niet call-out box reactie-overlay-element gebeurt analoog aan het aanmaken van een normaal overlay-element. Het verschil hier is dat aan deze functie ook de overlaymanager is meegegeven. De functie gaat via de manager het dom-element van het bronelement opvragen. De onclickfunctie van het overlayelement object (en dus ook het dom element) gaat vervangen worden door een functie die een parameter in de animatieparameters van het reactie-overlay-element aanpast (zie verder).

Bij call-out box reactie-overlay-ellementen wordt de content die de gebruiker meegeeft aangepast. De inhoud van het overlay-element wordt een canvas waar we de call-out box op gaan tekenen en een div waar de inhoud in komt te staan. De div waar de inhoud (inhoud-div) in komt te staan, krijgt de groote die gebruiker heeft opgegeven in de opmaak van het overlay-element. De grootte van het gehele overlay-element en het canvas is 15 pixels breder en hoger dan de opgegeven grootte. Dit is gedaan zodat de call-out box rondom de inhoud kan getekend worden. In "Figuur 6.14" is de rode vierhoek het canvaselement en de groene vierhoek de inhoud-div. De inhoud-div wordt tijdens het animeren correct gepositioneerd zodat het ver genoeg staat van al de randen van de call-out box. Aan de inhoud-div wordt ook nog de overflow css property op auto ingesteld. Deze zorgt er voor dat de inhoud-div scrollbars krijgt indien de inhoud groter is dan de grootte van de inhoud-div. (Zie "Figuur 6.12")

Figuur 6.14: Reactie call-out box inhoud

Animeren reactie-overlay-elementen

Voor het animeren van de reactie-overlay-elementen werd er een nieuwe animatie engine aangemaakt namelijk de '*AnimationReaction'* engine. Reactie-overlay-elementen krijgen als animatieparameters een '*AnimParamsReaction'* object. In dit '*AnimParamsReaction'* object worden de volgende dingen bijgehouden:

1. Is het bronelement aangeklikt geweest (true/false) ? Dit is de parameter die wordt aangepast in de onclick eventhandler van het bronelement.
2. Positie en afmetingen van het bronelement in pagina.
3. Positie en afmetingen van het videoscherm (canvaselement) in pagina.
4. Hoogte en breedte van het reactie-overlay-element.
5. Afstand tussen bron en reactie-overlay-element.
6. Lijst van mogelijke posities tov het bron overlay-element die het reactie-overlay-element kan aannemen.
7. Moet het reactie-overlay-element zijn positie t.o.v. het bron overlay-element behouden als een positie van hogere prioriteit vrij komt ? ('*keepPosition'* waarde)

Om reactie-overlay-elementen correct te positioneren, gebeurt in de '*positionOverlayElt'* functie de volgende stappen:

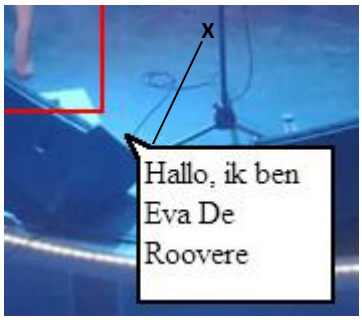
1. Vraag informatie op over het videoscherm en het bron overlay-element en geef deze informatie aan de animatie parameters van het reactie-overlay-element.
2. Laat de '*AnimationReaction'* engine de positie en inhoud berekenen van het reactie-overlay-element. In tegenstelling tot de twee andere animatie engines gaan de positie coördinaten scherm coördinaten zijn in plaats van video coördinaten.

De positie wordt berekend met de '*doAnimationStep'* functie en de inhoud met de '*doAnimationStepContentChange'* functie.

De '*AnimationReaction'* engine kijkt eerst of de '*keepPosition'* waarde op 'true' staat. Als deze 'true' is, wordt eerst gekeken of het reactie-overlay-element kan geplaatst worden op de positie die het in het vorige frame iteratie had t.o.v. het bron-overlay-element. Indien dit niet mogelijk is, wordt de lijst

van mogelijke posities t.o.v. het bron-overlay-element afgegaan. Als de '*keepPosition'* waarde 'false' is, wordt de positie berekend door de lijst van mogelijke posities t.o.v. het bron-overlay-element af te gaan. De engine gaat kijken of het element op de eerste positie t.o.v. het bron-overlay-element in de lijst kan staan door de afstanden te berekenen tussen het bron-overlay-element en de randen van het videoscherm (canvaselement). Indien de grootte van het reactie-overlay-element plus de afstand tussen bron en reactie-overlay-element groter is dan de berekende afstand, dan gaat de animatie engine kijken naar de volgende positie t.o.v. het bron-overlay-element in de lijst. Als het een positie heeft gevonden waar het reactie-overlay-element past, dan geeft de engine de positie-coördinaten terug. Indien er geen mogelijke positie t.o.v. het bron-overlay-element wordt gevonden waar het reactie-overlay-element getekend kan worden, dan geeft de engine null terug. Als de engine null terug geeft, dan weet de manager dat dit element niet getekend moet worden en gaat de '*positionOverlayElt'* functie 'false' terug geven.

De inhoud wordt berekend door te kijken op welke positie t.o.v. het bron-overlay-element het reactie-overlay-element staat. Afhankelijk van deze relatieve positie wordt een lijst van punten terug gegeven. Indien het reactie-overlay-element geen call-out box opmaak heeft, geeft de animatie engine null terug. De manager gaat deze punten tekenen op het canvas van het reactie-overlay-element. De getekende figuur wordt ook opgevuld met wit. De inhoud-div moet nu correct gepositioneerd worden in het reactie-overlay-element. Dit doet de manager door te kijken naar de lijst van teruggekregen punten. Het eerste punt in de lijst is meestal het meest links boven getekend punt. Bij dit punt gaat de manager een marge optellen. Dit is dan de positie van de inhoud-div. Enkel wanneer het element rechts vanonder staat is dit anders. Hierbij is het eerste punt in de lijst het punt X op "Figuur 6.15". Hierbij moet in de horizontale richting een marge worden afgetrokken in in de verticale richting een marge worden bijgeteld.

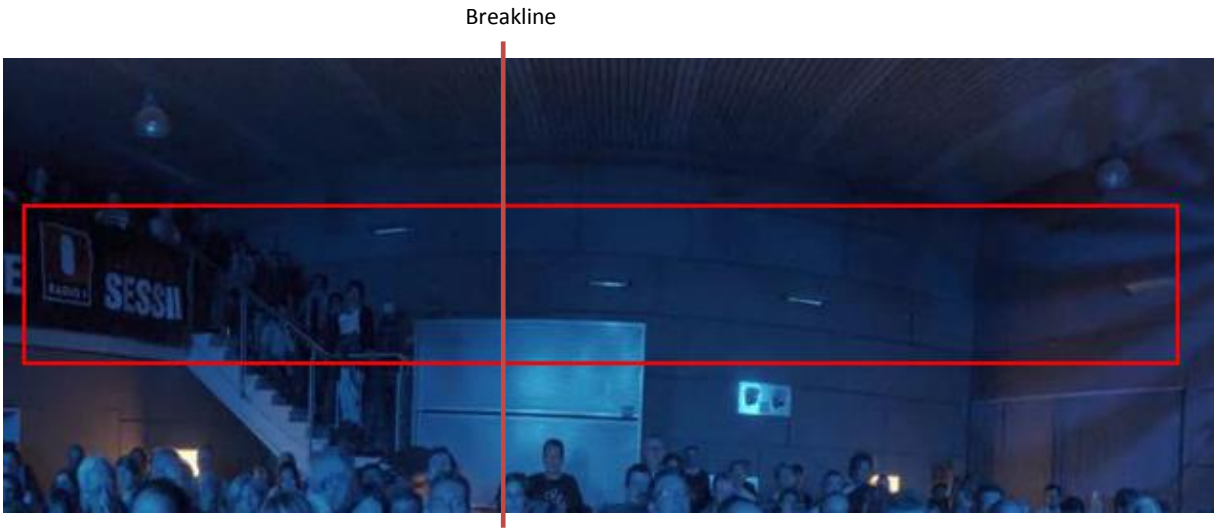


Figuur 6.15: Reactie call-out box rechtsonder.

6.2 Resultaat en beperkingen

Het systeem werkt zowel voor Chrome, Firefox als Internet Explorer.

Overlay-elementen kunnen overal in de video geplaatst worden en worden correct weergegeven en geschaleerd. Het systeem kan ook omgaan met overlay-elementen die met trackingdata geanimeerd worden.



Figuur 6.16: Breed overlay-element over breakline

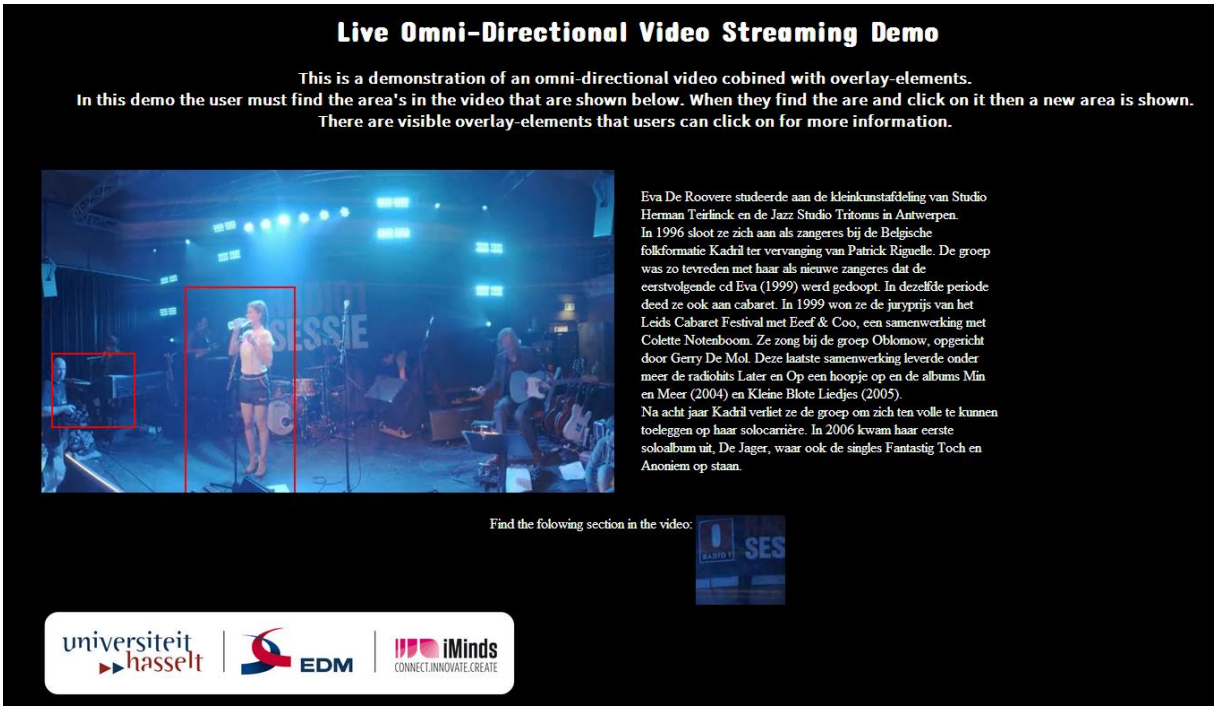
Figuur 6.17: Breed overlay-element over breakline met breakline aanwezig in window

In “Figuur 6.17” zien we een voorbeeld overlay-element in de situatie van “Figuur 6.6”. Het overlay-element is zeer breed en steekt de breakline over. Toch blijft het de correcte positie aanhouden ook al is de breakline niet meer in zicht is, zoals het geval is in “Figuur 6.17”.

Figuur 6.18: Extreem breed overlay-element

Het systeem heeft echter een beperking, namelijk dat het niet om kan gaan met extreem lange overlay-elementen. In “Figuur 6.18” lijkt alles op het eerste gezicht in orde te zijn maar dat is niet zo. Het overlay-element dat we zien heeft een breedte van 1900. Het videobestand heeft een breedte van 1920. Dit wil zeggen dat het andere uiteinde van het overlay-element normaal gezien links terug zichtbaar moet zijn. Om dit probleem op te lossen zou het systeem moeten uitgebreid worden zodat het overweg kan met deel-overlay-elementen. Hiermee wordt bedoeld dat een overlay-element kan opgesplitst worden in verschillende delen die op meerdere plaatsen in de video zichtbaar zijn.

6.3 Use case



Figuur 6.19: ODV use case

Als laatste heb ik een use case ontworpen rond het huidige systeem. In deze use case krijgen de gebruikers een ODV te zien van een concert van Eva de Roovere voor Radio1, zie “Figuur 6.19”. De gebruiker kan in de video op bepaalde overlay-elementen klikken. Als ze klikken op het overlay-element van Eva, komt er een reactie-overlay-element tevoorschijn zoals te zien is in “Figuur 6.13”. Als de gebruiker met zijn muis over het overlay-element van Eva beweegt, verschijnt er rechts een korte biografie van haar. Andere zichtbare overlay-elementen hebben ook reactie-overlay-elementen er aan gekoppeld.

De gebruiker moet in de use case op zoek gaan naar bepaalde plaatsen in de video. Onder de video wordt een klein deel van de video uitgeknipt en op een canvas getekend (gelijkaardig aan het tekenen van de video op het canvas van de ODV player). De gebruiker moet deze plaats vinden en er op klikken. Als de gebruiker er op klikt dan verandert de plaats die hij moet zoeken naar een volgende plaats. Als de gebruiker al de plaatsen heeft gevonden, dan komt er een bericht tevoorschijn.

Besluit

Ik ben deze masterthesis begonnen met het maken van een videoplayer voor de drie browsers met de meeste gebruikers: Internet Explorer, Firefox en Chrome. Deze player heeft als eigenschap dat hij voor deze drie browsers dezelfde functionaliteit aanbiedt en er hetzelfde uitziet.

Vervolgens heb ik enkele testen gemaakt rond interactieve video in HTML5 in combinatie met een virtuele 3D-wereld geïmplementeerd in Unity3D. Hieruit blijkt dat het mogelijk is om beide technieken te combineren tot een werkend geheel. Zo kunnen interacties in een video, zoals het klikken op een overlay-element, een effect hebben in een virtuele 3D-wereld. Hetzelfde geldt ook in omgekeerde richting. Aan de hand van het systeem kan ik de tweede onderzoeksvraag “Welke soorten interacties zijn er allemaal mogelijk tussen een video en een virtuele 3D-wereld?” beantwoorden. In het tweede hoofdstuk heb ik enkele voorbeelden gegeven van interacties zoals het donkerder maken van de video, het aanpassen van de grootte van de video, het veranderen van tijdsstip in de video en het aanpassen van het geluidsvolume van de video naargelang acties in de virtuele 3D-wereld. Omgekeerd is het ook mogelijk om veranderingen in de virtuele 3D-wereld aan te brengen naar gelang een gebruiker een actie voltooit op een video, zoals een lamp die aangaat als de gebruiker op een video klikt. Het systeem voorgesteld in dit hoofdstuk heeft wel als nadeel dat het niet werkt op tabletpc’s aangezien hier de Unity webplayer nog niet op werkt. Andere systemen waarmee men virtuele 3D-werelden kan weergeven in websites kunnen ook voor dit systeem gebruikt worden zolang er communicatie mogelijk is tussen de 3D-wereld en de website waar het zich in bevindt.

Uit test 1 van hoofdstuk drie blijkt dat een canvas overlay niet optimaal is om interactieve video te implementeren. Dit komt omdat wanneer we met één grote canvas overlay werken we moeilijk kunnen detecteren wanneer een gebruiker op een tekst overlay-element heeft geklikt. Ook kan het systeem moeilijk overweg met veel overlay-elementen die tegelijk getoond worden. Daarom heb ik in hoofdstuk vier verdergewerkt op het interactieve videosysteem van dr. Wijnants. Dit systeem gebruikt div-elementen om interactieve video overlays te visualiseren bovenop een HTML5 videoplayer. Het voordeel aan deze methode is dat de gebruiker veel vrijheid heeft over de inhoud van een overlay-element. De gebruiker heeft ook de mogelijkheid zelf de afhandeling van click, mouseover en mouseout events van de overlay-elementen in te stellen. Hierdoor kan men zowel bovenop als naast de video veranderingen aanbrengen na het aanklikken van een overlay-element. In het originele interactieve videosysteem bewegen deze elementen enkel via lineaire interpolatie. Dit systeem heb ik uitgebreid met motiontracking. In de nieuwe versie van het systeem kunnen gebruikers motiontrack data meegeven aan de definitie van een overlay-element. In deze thesis heb ik gewerkt met Mocha AE om objecten in een video te tracken. In Mocha AE kan een gebruiker vier trackingpunten aanduiden. Het programma zal dan de beweging van deze punten doorheen de videoscène volgen. Vervolgens kan de motiontrack data geëxporteerd worden naar een bestand dat vervolgens ingelezen kan worden in het interactieve videosysteem. Het overlay-element zal dan bewegen volgens deze motiontrack data. Het voorgestelde systeem bevat vier animatietechnieken om een overlay-element aan de hand van motiontrack data te visualiseren:

1. *Predefined boundingbox:* dit is een overlay-element met zelfingestelde vaste grootte De inhoud van het overlay-element is op voorhand gedefinieerd.
2. *Maximum animation boundingbox:* hierbij wordt de maximale grootte van het overlay-element berekend aan de hand van de motiontrack data. Deze grootte behoudt het overlay-

element doorheen de hele animatie. De inhoud van het overlay-element is op voorhand gedefinieerd.

3. *Dynamic bounding box:* dit is een het overlay-element waarbij de grootte verandert op elk tijdstip aan de hand van de motiontrack data. De inhoud van het overlay-element is op voorhand gedefinieerd.
4. *Outlined trackingpoints:* hierbij wordt de inhoud van het overlay-element vervangen door een canvas. Op deze canvas worden de trackingpunten getekend en verbonden zodat ze een vierhoek vormen. Hierdoor worden de trackingpunten gevisualiseerd.

Een duidelijk overzicht van hoe deze vier animatietechnieken er in de praktijk uit zien is te vinden in sectie 4.4. De '*Predefined boundingbox* 'en de '*Maximum animation boundingbox*' techniek zijn computationeel minder zwaar maar kunnen niet goed omgaan met objecten die van grootte veranderen doorheen een video. De '*Dynamic bounding box'* en de *'Outlined trackingpoints'* techniek zijn computationeel zwaarder maar geven een beter visueel resultaat. Deze vier technieken zijn echter zeer afhankelijk van de kwaliteit van de motiontrack data. Deze nieuwe overlay-elementen geven makers van interactieve video's opties en creatieve mogelijkheden. Het gemaakte systeem kan gebruikt worden voor het beantwoorden van de eerste onderzoeksvraag "Welke soorten interactie kunnen we toevoegen aan een video om deze video interessanter te maken?". Wanneer men bijvoorbeeld een auto aanklikbaar wil maken in een autoreclame, dan moet deze niet meer stilstaan om er een overlay-element op te kunnen plaatsen.

Met de integratie van Google Analytics kunnen eigenaars van interactieve video's duidelijke statistieken opvragen over hoe gebruikers omgaan met de video. Zo kunnen ze zien hoeveel en hoelang gebruikers naar de video gekeken hebben, op welke overlay-elementen er geklikt werd en hoe ze de video hebben verlaten. Het nadeel aan Google Analytics is dat de resultaten pas na een dag beschikbaar zijn en dat data wordt geaggregeerd over alle gebruikers. Het is dus niet mogelijk om te zien welke acties een individuele gebruiker heeft uitgevoerd op een site. Voor het geval dat de eigenaar nog meer informatie wil vergaren, heb ik een databaserecorder geïmplementeerd waarmee gedetailleerdere informatie wordt opgeslagen zoals waar op het overlay-element de gebruiker klikt of welke knoppen van de videoplayer de gebruiker gebruikt. Zo kunnen makers zien of gebruikers vaak de video pauzeren moeten om op een overlay-element te kunnen klikken. Deze tools zijn ook geschikt om usertesten uit te voeren, bijvoorbeeld over toekomstige toevoegingen aan het systeem.

In het laatste deel van de masterthesis heb ik het interactieve video systeem uitgebreid zodat het ook om kan gaan met omnidirectionele video (ODV). Dit is een video die opgenomen is door een camera die 360 graden rondom zich kan filmen. De ODV player voorziet een viewwindow via hetwelke gebruikers naar een deel van de video kunnen kijken. Dit window kunnen ze verplaatsen en er kan mee worden in- en uitgezoomed zodat men andere delen van de video kan zien. De ODV-player is gemaakt door dr. Jori Liesenborgs. Aangezien in de odv player de volledige video niet langer zichtbaar is, moeten de overlay-elementen in dit geval correct gepositioneerd worden ten opzichte van het viewwindow. Zo mogen overlay-elementen die buiten het viewwindow liggen niet zichtbaar zijn. Overlay-elementen die gedeeltelijk zichtbaar zijn, moeten geclipt worden. Ook moeten de overlay-elementen worden aangepast naargelang het zoomlevel van het viewwindow. In dit hoofdstuk heb ik ook een nieuw soort overlay-element gedefinieerd, namelijk het reactie-overlay-element. Dit element verschijnt pas wanneer er op een ander overlay-element wordt geklikt. Het

reactie-overlay-element wordt gepositioneerd naargelang de positie van het element waarop het een reactie is. Met deze reactie-overlay-elementen kan er extra informatie verschijnen over de inhoud van een ander overlay-element, zoals het geval was in de use case in sectie 6.3. Aan de hand van dit uitgebreid systeem kan ik de derde onderzoeksvraag beantwoorden “Hoe kunnen we overlay-elementen op een zinvolle imersieve manier toevoegen aan een omnidirectionele video?”. Gebruikers kunnen aan de hand van de reactie-overlay-elementen personen in een video laten spreken. Dat kan bijvoorbeeld gebeuren door tekst in de call-out box te plaaten of door er een ander audio-element in te steken. Een mogelijke toepassing is ook de lyrics van het lied dat een zanger zingt in de call-out box te laten verschijnen.

Doorheen de thesis heb ik verscheidene vormen van augmented video uitgewerkt. Hieruit heb ik afgeleid dat de combinatie tussen video en virtuele 3D-wereld een goede combinatie is om bepaalde doeleinden te bereiken. Met het interactieve video systeem hebben gebruikers veel vrijheid om video’s te maken. Dankzij deze interactieve video’s kunnen kijkers interactiever omgaan met de video waardoor de interesse in de video kan verhoogd worden.

Toekomstig werk

In deze sectie van de masterthesis bespreek ik mogelijke verbeteringen die aan het systeem kunnen worden gebracht.

Het interactieve videosysteem is niet performant genoeg voor tablets. De visualisatie van de overlay-elementen gebeurt niet aan een snel tempo waardoor de animaties niet vlot zijn. Aangezien tablet-pc's meer en meer gebruikt worden moet dit systeem hier ook voor beschikbaar worden gemaakt. De Unity Webplayer is nog niet beschikbaar voor tablets. Indien deze beschikbaar wordt, kan er nog eens gekeken worden naar hoe de combinatie werkt op een tablet-pc.

In deze thesis hebben we gewerkt met Mocha AE voor het tracken van objecten in een video. Het systeem zou echter moeten getest worden door ook andere tracking programma's te gebruiken. Het systeem is gebouwd zodat het kan omgaan met trackingprogramma's die meer dan vier trackingpunten gebruiken. Hier zijn echter geen testen rond uitgevoerd. Vooral de *'Outlined trackingpoints'* animatietechniek zou visueel veel betere resultaten kunnen geven indien er meer trackingpunten worden gebruikt. Deze animatietechniek heeft tot nu toe ook enkel maar één visualisatiemogelijkheid. Het gaat hier om zwarte rechte lijnen die de punten verbinden. Dit zou uitgebreid kunnen worden naar een systeem dat toelaat de opmaak in te stellen. Zo zouden gebruikers de kleur en lijndikte kunnen instellen.

Het systeem voorgesteld in deze thesis is niet getest geweest aan de hand van formele testen. Er zouden testen moeten uitgevoerd worden over hoe gebruikers omgaan met de overlay-elementen. Er kan getest worden of de overlay-elementen niet te veel van de aandacht van de gebruiker opnemen waardoor men minder op de inhoud van de video let. Ook moet er getest worden hoe goed gebruikers hun aandacht kunnen verdelen tussen de virtuele 3D-wereld en een interactieve video, indien deze technieken gecombineerd worden.

Overlay-elementen die extreem lang zijn moeten correct gevisualiseerd worden in een ODV. In de huidige implementatie is dit nog niet mogelijk. Ook is er veel ruimte voor uitbreidingen aan het reactie-overlay-element. Momenteel verspring het reactie-overlay-element van één locatie naar een andere als er geen plaats meer is. Dat zou kunnen worden verbeterd door het element geleidelijk van plaats te laten veranderen. Bij het positioneren van het reactie-overlay-element wordt er nu alleen maar rekening gehouden met het element waarop het een reactie is en het viewwindow. Hier zou het systeem ook kunnen worden uitgebreid naar een systeem waarbij het reactie-overlay-element rekening houdt met de positie van andere overlay-elementen zodat deze elkaar niet overlappen.

Dankwoord

Ten eerste zou ik graag mijn assessor dr. Maarten Wijnants willen bedanken. Zonder zijn hulp had ik deze thesis nooit tot een goed einde kunnen brengen. Hij was altijd bereid mij te helpen indien nodig en spoorde mij aan om niet meteen tevreden te zijn met het werk dat ik leverde. Hij heeft mij doorheen deze thesis veel bijgeleerd.

Vervolgens gaat mijn dank uit naar mijn promotor professor Wim Lamotte. Zijn advies over de inhoud van deze masterthesis werd zeer geapprecieerd.

Daarnaast wil ik ook graag mijn co-promotor professor Peter Quax bedanken. Hij stond altijd paraat met de nodige apparatuur om mijn masterthesis te verwezenlijken.

Mijn dankbaarheid gaat ook uit naar dr. Jori Liesenborgs. Dankzij zijn ODV player kon ik het laatste deel van mijn masterthesis maken.

Als laatste zou ik nog graag mijn vriendin Katrien Martens bedanken. Zij verbeterde steeds opnieuw mijn teksten.

Referenties

[1] V. M. Bove, Jr, J. Dakss, E. Chalom, en S. Agamanolis. Hyperlinked television research at the MIT Media Laboratory. IBM Systems Journal Volume 39 Issue 3-4. P. 470-478, New York, NY, USA Juli 2000

[2] Guang-Ho Cha. Object-Based Interactive Video Access for Consumer-Driven Advertising. Proceedings of the 8th International Conference, EC-Web 2007. P. 222–228, Regensburg, Germany, September 3-7, 2007

[3] M. Sadallah, O. Aubert, and Y. Prié. CHM: An Annotation- and Component-based Hypervideo Model for the Web. Multimedia Tools and Applications. P. 1-35, 2012.

[4] Popcorn.js. Website http://popcornjs.org/ (22/01/2014)

[5] YouTube Getting started: creating or editing annotations. Website http://www.google.com/support/youtube/bin/answer.py?hl=en&answer=92710 (22/01/2014)

[6] WireWax The world's first taggable video tool. Website https://www.wirewax.com/ (22/01/2014)

[7] P. Hoffmann, T. Kochems, en M. Herczeg. HyLive:Hypervideo-Authoring for Live Television. InChanging Television Environments, volume 5066 van Lecture Notes in Computer Science. P. 51-60. Springer Berlin Heidelberg, 2008

[8] Nitin “Nick” Sawhney, David Balcom, en Ian Smith. HyperCafe: Narrative and Aesthetic Properties of Hypervideo. Proceedings of the seventh ACM conference on Hypertext. p. 1-10, New York, NY, USA

[9] Frank Shipman Texas, Andreas Girgensohn, en Lynn Wilcox. An Overview of Hyper-Hitchcock. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) Volume 5 Issue 2, November 2008 Artikel No. 15.

[10] Britta Meixner, Beate Siegel, en Günter Hölbling. SIVA Suite Authoring System and Player for Interactive Non-linear Videos. Geplubliceerd in the international conference on Multimedia. p. 1563-1566. ACM New York, NY, USA. 2010

[11] Tae-Jin Park, Jae-Kyung Kim, en Yoon-Chul Choy. Creating a clickable TV program by sketching and tracking freeform triggers. Multimedia Tools and Applications Volume 59, Issue 3. p. 833-850, Augustus 2012

[12] Toshiharu Sugawara, Satoshi Kurihara, Shigemi Aoyagi, Koji Sato, en Toshihiro Takada. An Implementation for Capturing Clickable Moving Objects. Proceerdings of the 6th Asia Pacific Conference, APCHI 2004. p. 441-450, Rotorua, New Zealand, June 29 - Juli 2, 2004

[13] Unity3D. http://unity3d.com/ (22/01/2014)

[14] Three.js. http://threejs.org/ (22/01/2014)

[15] Scalable Vector Graphics. http://www.w3.org/Graphics/SVG/ (22/01/2014)

[16] WebGL. http://www.khronos.org/webgl/ (22/01/2014)

[17] Cross Browser HTML5 Progress Bars In Depth. Website http://www.useragentman.com/blog/2012/01/03/cross-browser-html5-progress-bars-in-depth/ (22/01/2014)

[18] Coherent blog. Adding a HTML5 minigame in Unity3D with Coherent UI (Tutorial). Website http://blog.coherent-labs.com/2013/02/html5-game-unity-3d.html (22/01/2014)

[19] Open source computer vision. Website http://opencv.org/ (22/01/2014)

[20] OpenCV Color Detection & Object Tracking. Website http://opencv-srf.blogspot.ro/2010/09/object-detection-using-color-seperation.html (22/01/2014)

[21] Tracking-Learning-Detection. Website http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html (22/01/2014)

[22] Youtube Predator: Camera That Learns. Website http://www.youtube.com/watch?v=1GhNXHCQGsM (22/01/2014)

[23] Zdenek Kalal. Tracking Learning Detection. Centre for Vision, Speech and Signal Processing Faculty of Engineering and Physical Sciences University of Surrey. April 2011

[24] After Effect CC. Website http://www.adobe.com/be_nl/products/aftereffects.html (22/01/2014)

[25] Mocha developed by imagineer systems. Website http://www.imagineersystems.com/products/mochaAE (22/01/2014)

[26] 11 Best Web Analytics Tools. Website http://www.inc.com/guides/12/2010/11-best-web-analytics-tools.html/1 (22/01/2014)

[27] Top 30 Web Analytics Tools | Google Analytics Alternative. Website http://www.adpushup.com/blog/web-analytics-tools-google-analytics-alternatives/ (22/01/2014)

[28] Afbeelding http://searchengineland.com/figz/wp-content/seloads/2013/02/web-analytics-software-comparison1.jpg (22/01/2014)

[29] Event Tracking - Web Tracking (ga.js). Website https://developers.google.com/analytics/devguides/collection/gajs/eventTrackerGuide?hl=nl (22/01/2014)

[30] Clicky -VS- The Other Guys. Website http://clicky.com/compare/ (22/01/2014)

[31] GoingUp. Website http://www.goingup.com/ (22/01/2014)

[32] HTML Canvas 2D Context W3C Candidate Recommendation 6 August 2013. Website http://www.w3.org/TR/2dcontext/ (22/01/2014)

[33] Visual effects W3C Recommendation. Website http://www.w3.org/TR/CSS21/visufx.html

[34] .offset(). Website http://api.jquery.com/offset/ (22/01/2014)

[35] CSSOM View Module W3C Working Draft 17 December 2013. Website http://www.w3.org/TR/2013/WD-cssom-view-20131217/ (22/01/2014)

Bijlage1

1. Test acties

Test1 Google chrome, geen test sessie

Angrybirds Start

0:50 klik op DeStorm

Angrybirds DeStorm

1:16 klik op Try again

Angrybirds Start

0:07 spoel door naar 0:49

0:49 klik op DeStorm

Angrybirds DeStorm

0:51 klik op End

Angrybirds End

0:17 sluit af

Test2 Google chrome, geen test sessie

Angrybirds Start

0:09 klik op logo van de pet (id: Cap)

0:13 klik op angryBird

0:49 klik op JReyez

Angrybirds JReyez

0:40 spoel terug naar 0:31

0:38 klik op End

Angrybirds End

0:16 klik YoutubeChannel

0:16 sluit af

Test3 FireFox, geen test sessie

Angrybirds Start

0:09 klik op logo van de pet

0:52 klik op DeStorm

Angrybirds DeStorm

0:14 sluit af

Test4 IE, geen test sessie

Angrybirds Start

0:13 Klik op angryBird

0:13 spoel terug naar 0:03

0:09 klik op logo van de pet

0:50 klik op DeStorm

Angrybirds DeStorm

0:46 spoel door naar 1:22

1:21 klik op try again

Angrybirds Start

0:09 spoel door naar 0:56

0:56 klik op JReyez

Angrybirds JReyez

0:15 sluit af

Test5 Chrome, met test sessie, User1 & Session1

Angrybirds Start

0:01 spoel terug naar 0:00

0:00 sessie naam: Session1 gebruiker: User1

0:00 play

0:09 pauze

0:09 klik op logo van de pet

0:09 klik op oorbel

0:09: play

0:52 pauze

0:52 klik op JReyez

Angrybirds JReyez

0:03 pauze

0:03 sluit af

Test6 Chrome, met test sessie

Angrybirds Start

0:02 spoel terug naar 0:00

0:01 sessie naam: Session2 gebruiker: User1

0:01 play

0:24 pauze

0:24 klik op video

0:24 play

0:49 pauze

0:49 klik op DeStorm

Angrybirds DeStorm

0:25 pauze

0:25 versneld kijken

0:25 mute

0:35 play

0:56 pauze

0:56 vertraagd kijken

0:56 unmute

0:56 spoel terug naar 0:39

0:39 play

0:51 pauze

0:51 klik op End

Angrybirds End

0:10 pauze

0:10 sluit af

Test7 Chrome, met test sessie

Angrybirds Start

0:01 spoel terug naar 0:00

0:00 sessie naam: Session1 gebruiker: User2

0:00 play

0:06 pauze

0:06 spoel door naar 0:30

0:30 klik 2 x op video

0:30 play

0:56 pauze

0:56 klik op JReyez

Angrybirds JReyez

0:04 pauze

0:04 spoel door naar 0:34

0:34 play

0:38 pauze

0:38 klik op End

Angrybirds End

0:18 pauze

0:18 klik YoutubeChannel

0:18 sluit af

Test8 Chrome, geen test sessie

Angrybirds Start

0:12 klik op angryBird

0:50 klik op DeStorm

Angrybirds DeStorm

0:08 sluit af

2. Verwachte resultaten in Google Analytics

AngryBirds Start:

Action	Label	Value
Visitor started watching this video		10
Visitor clicked on element	DeStorm	6
	JReyez	4
	Cap	4
	AngryBird	3
	Earring	1
Visitor left this video by clicking on element with ID: DeStorm	X%	6
Visitor left this video by clicking on element with ID: JReyez	X%	4
Visitor left this video by leaving page	X%	
Visitor watched x seconds of this video		50+7+49+52+60+9+53+51+32+50=413
Visitor seeked Video	Forward	42+47+24=113
	Backward	10+1+2+1=14

AngryBirds DeStorm

Action	Label	Value
Visitor started watching this video		6
Visitor clicked on element	Try again	2
	End	2
Visitor left this video by clicking on element with ID: TryAgain	X%	2
Visitor left this video by clicking on element with ID: End	X%	2
Visitor left this video by leaving page	X%	2
Visitor watched x seconds of this video		76+51+14+46+68+8=263
Visitor seeked Video	Forward	36
	Backward	17

AngryBirds JReyez

Action	Label	Value
Visitor started watching this video		4
Visitor clicked on element	Try again	
	End	2
Visitor left this video by clicking on element with ID: TryAgain	X%	
Visitor left this video by clicking on element with ID: End	X%	2
Visitor left this video by leaving page	X%	2
Visitor watched x seconds of this video		47+15+3+8=73
Visitor seeked Video	Backward	9
	Forward	30

AngryBirds End

Action	Label	Value
Visitor started watching this video		4
Visitor watched x seconds of this video		17+16+10+18=61
Visitor left this video by leaving page	X%	4
Visitor clicked on element	YoutubeChannel	2

3. Resultaten in Google Analytics

AngryBirds Start:

Algemeen overzicht:

Event Action	Total Events	Unique Events	Event Value	Avg. Value
	187 % of Total: 51.52% (363)	3 % of Total: 33.33% (9)	558 % of Total: 53.04% (1,052)	2.98 Site Avg: 2.90 (2.96%)
1. Visitor watched x seconds of this video	142	3	413	2.91
2. Visitor clicked on element	18	3	18	1.00
3. Visitor started watching this video	10	3	0	0.00
4. Visitor seeked Video	7	2	127	18.14
5. Visitor left this video by clicking on element with ID: DeStorm	6	3	0	0.00
6. Visitor left this video by clicking on element with ID: JReyez	4	2	0	0.00

Visitor clicked on element:

Event Label	Total Events	Unique Events
	18 % of Total: 4.96% (363)	3 % of Total: 33.33% (9)
1. ID: DeStorm	6	3
2. ID: Cap	4	3
3. ID: JReyez	4	2
4. ID: AngryBird	3	2
5. ID: Earring	1	1

Visitor left video by clicking element with ID: DeStorm

Event Label	Total Events
	6 % of Total: 1.65% (363)
1. Left at 77% in video	3
2. Left at 76% in video	2
3. Left at 80% in video	1

Visitor left video by clicking element with ID: JReyez

Event Label	Total Events
	4 % of Total: 1.10% (363)
1. Left at 86% in video	2
2. Left at 76% in video	1
3. Left at 80% in video	1

Visitor seeked video:

Event Label	Total Events	Unique Events	Event Value	Avg. Value
	7 % of Total: 1.93% (363)	2 % of Total: 22.22% (9)	127 % of Total: 12.07% (1,052)	18.14 Site Avg: 2.90 (526.03%)
1. Backward	4	2	13	3.25
2. Forward	3	2	114	38.00

AngryBirds DeStorm:

Algemeen overzicht

Event Action	Total Events	Unique Events	Event Value	Avg. Value
	106 % of Total: 29.20% (363)	3 % of Total: 33.33% (9)	319 % of Total: 30.32% (1,052)	3.01 Site Avg: 2.90 (3.84%)
1. Visitor watched x seconds of this video	89	3	262	2.94
2. Visitor started watching this video	6	3	0	0.00
3. Visitor clicked on element	4	2	4	1.00
4. Visitor left this video by clicking on element with ID: End	2	1	0	0.00
5. Visitor left this video by clicking on element with ID: TryAgain	2	2	0	0.00
6. Visitor seeked Video	2	2	53	26.50
7. Visitor left this video by leaving page	1	1	0	0.00

Visitor clicked on element:

Event Label	Total Events
	4 % of Total: 1.10% (363)
1. ID: End	2
2. ID: TryAgain	2

Visitor left video by clicking element with ID: End

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. Left at 55% in video	2

Visitor left video by clicking element with ID: TryAgain

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. Left at 83% in video	1
2. Left at 90% in video	1

Visitor seeked video

Event Label	Total Events	Unique Events	Event Value	Avg. Value
	2 % of Total: 0.55% (363)	2 % of Total: 22.22% (9)	53 % of Total: 5.04% (1,052)	26.50 Site Avg: 2.90 (814.40%)
1. Backward	1	1	17	17.00
2. Forward	1	1	36	36.00

Visitor left video by leaving page:

Event Label	Total Events
	1 % of Total: 0.28% (363)
1. Left at 9% in video	1

AngryBirds JReyez:

Algemeen overzicht

Event Action	Total Events	Unique Events	Event Value	Avg. Value
	38 % of Total: 10.47% (363)	2 % of Total: 22.22% (9)	114 % of Total: 10.84% (1,052)	3.00 Site Avg: 2.90 (3.52%)
1. Visitor watched x seconds of this video	26	2	73	2.81
2. Visitor started watching this video	4	2	0	0.00
3. Visitor clicked on element	2	1	2	1.00
4. Visitor left this video by clicking on element with ID: End	2	1	0	0.00
5. Visitor left this video by leaving page	2	2	0	0.00
6. Visitor seeked Video	2	1	39	19.50

Visitor clicked on element:

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. ID: End	2

Visitor left video by clicking element with ID: End

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. Left at 48% in video	2

Visitor seeked video

Event Label	Total Events	Unique Events	Event Value	Avg. Value
	2 % of Total: 0.55% (363)	1 % of Total: 11.11% (9)	39 % of Total: 3.71% (1,052)	19.50 Site Avg: 2.90 (572.86%)
1. Backward	1	1	9	9.00
2. Forward	1	1	30	30.00

Visitor left video by leaving page:

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. Left at 19% in video	1
2. Left at 4% in video	1

AngryBirds End:

Algemeen overzicht

Event Action	Total Events	Unique Events	Event Value	Avg. Value
	32 % of Total: 8.82% (363)	1 % of Total: 11.11% (9)	61 % of Total: 5.80% (1,052)	1.91 Site Avg: 2.90 (-34.22%)
1. Visitor watched x seconds of this video	22	1	59	2.68
2. Visitor left this video by leaving page	4	1	0	0.00
3. Visitor started watching this video	4	1	0	0.00
4. Visitor clicked on element	2	1	2	1.00

Visitor clicked on element:

Event Label	Total Events
	2 % of Total: 0.55% (363)
1. ID: YoutubeChannel	2

Visitor left video by leaving page:

Event Label	Total Events
	4 % of Total: 1.10% (363)
1. Left at 27% in video	1
2. Left at 42% in video	1
3. Left at 45% in video	1
4. Left at 47% in video	1

4. Resultaten in MySQL database

Session1 User1:

Alle events:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	EventType	VideoTime	RealTime
497	1	angryBirdsStart.mp4	clickOnVideoPlayerButton	0	2014-01-17 21:10:32.616
498	1	angryBirdsStart.mp4	clickOnVideoPlayerButton	9.12327	2014-01-17 21:10:41.822
499	1	angryBirdsStart.mp4	clickOnElement	9.12327	2014-01-17 21:10:47.557
500	1	angryBirdsStart.mp4	clickOnElement	9.12327	2014-01-17 21:10:51.146
501	1	angryBirdsStart.mp4	clickOnVideoPlayerButton	9.17188	2014-01-17 21:10:55.79
502	1	angryBirdsStart.mp4	clickOnVideoPlayerButton	52.047	2014-01-17 21:11:38.20
503	1	angryBirdsStart.mp4	clickOnElement	52.047	2014-01-17 21:11:40.331
504	2	angryBirdsJReyez.mp4	clickOnVideoPlayerButton	2.8651	2014-01-17 21:11:43.470

Click on element:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	VideoClickposition	EllementJavascriptID	EllementStartTime	EllementStopTime	EllementStartPos	EllementEndPos
499	1	angryBirdsStart.mp4	9.12327	2014-01-17 21:10:47.557	x:295,y:79	Cap	8.17484	10.3437	x:610,y:-4	x:300,y:-22
500	1	angryBirdsStart.mp4	9.12327	2014-01-17 21:10:51.146	x:415,y:197	Earring	8.29997	10.302	x:611,y:130	x:401,y:129
503	1	angryBirdsStart.mp4	52.047	2014-01-17 21:11:40.331	x:537,y:210	JReyez	48	65	x:410,y:26	x:410,y:26

Click on video player button:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	Action
497	1	angryBirdsStart.mp4	0	2014-01-17 21:10:32.616	Pause
498	1	angryBirdsStart.mp4	9.12327	2014-01-17 21:10:41.822	Play
501	1	angryBirdsStart.mp4	9.17188	2014-01-17 21:10:55.79	Pause
502	1	angryBirdsStart.mp4	52.047	2014-01-17 21:11:38.20	Play
504	2	angryBirdsJReyez.mp4	2.8651	2014-01-17 21:11:43.470	Play

Session2 User1:

All events:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	EventType	VideoTime	RealTime
505	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	0.124584	2014-01-17 21:13:32.602
506	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	23.9614	2014-01-17 21:13:56.532
507	3	angryBirdsStart.mp4	clickOnVideo	23.9614	2014-01-17 21:13:58.109
508	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	23.9614	2014-01-17 21:13:59.957
509	3	angryBirdsStart.mp4	clickOnVideoPlayerButton	49.096	2014-01-17 21:14:25.82
510	3	angryBirdsStart.mp4	clickOnElement	49.096	2014-01-17 21:14:32.716
511	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:14:58.15
512	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:15:00.689
513	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8932	2014-01-17 21:15:02.434
514	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	24.8947	2014-01-17 21:15:05.701
515	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:26.561
516	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:30.182
517	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	56.1564	2014-01-17 21:15:32.745
518	4	angryBirdsDeStorm.mp4	clickOnProgressBar	56.1564	2014-01-17 21:15:36.167
519	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	39.1271	2014-01-17 21:15:52.333
520	4	angryBirdsDeStorm.mp4	clickOnVideoPlayerButton	50.8203	2014-01-17 21:16:04.107
521	4	angryBirdsDeStorm.mp4	clickOnElement	50.8203	2014-01-17 21:16:06.227
522	5	angryBirdsEnd.mp4	clickOnVideoPlayerButton	10.2056	2014-01-17 21:16:16.664

Click on video:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	ClickPosition
507	3	angryBirdsStart.mp4	23.9614	2014-01-17 21:13:58.109	x:363,y:122

Click on element:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	VideoClickposition	EllementJavascriptID	EllementStartTime	EllementStopTime	EllementStartPos	EllementEndPos
510	3	angryBirdsStart.mp4	49.096	2014-01-17 21:14:32.716	x:125,y:234	DeStorm	48	65	x:75,y:26	x:75,y:26
521	4	angryBirdsDeStorm.mp4	50.8203	2014-01-17 21:16:06.227	x:355,y:66	End	50.5	51.5	x:320,y:40	x:320,y:40

Click on video player button:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	Action
505	3	angryBirdsStart.mp4	0.124584	2014-01-17 21:13:32.602	Pause
506	3	angryBirdsStart.mp4	23.9614	2014-01-17 21:13:56.532	Play
508	3	angryBirdsStart.mp4	23.9614	2014-01-17 21:13:59.957	Pause
509	3	angryBirdsStart.mp4	49.096	2014-01-17 21:14:25.82	Play
511	4	angryBirdsDeStorm.mp4	24.8932	2014-01-17 21:14:58.15	Play
512	4	angryBirdsDeStorm.mp4	24.8932	2014-01-17 21:15:00.689	FastForward
513	4	angryBirdsDeStorm.mp4	24.8932	2014-01-17 21:15:02.434	mute
514	4	angryBirdsDeStorm.mp4	24.8947	2014-01-17 21:15:05.701	Pause
515	4	angryBirdsDeStorm.mp4	56.1564	2014-01-17 21:15:26.561	Play
516	4	angryBirdsDeStorm.mp4	56.1564	2014-01-17 21:15:30.182	FastRewind
517	4	angryBirdsDeStorm.mp4	56.1564	2014-01-17 21:15:32.745	mute
519	4	angryBirdsDeStorm.mp4	39.1271	2014-01-17 21:15:52.333	Pause
520	4	angryBirdsDeStorm.mp4	50.8203	2014-01-17 21:16:04.107	Play
522	5	angryBirdsEnd.mp4	10.2056	2014-01-17 21:16:16.664	Play

Click on progress bar:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	ToTime
518	4	angryBirdsDeStorm.mp4	56.1564	2014-01-17 21:15:36.167	39.1271

Session1 User2:

All events:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	EventType	VideoTime	RealTime
523	6	angryBirdsStart.mp4	clickOnVideoPlayerButton	0.249168	2014-01-17 21:18:41.121
524	6	angryBirdsStart.mp4	clickOnVideoPlayerButton	6.01241	2014-01-17 21:18:46.985
525	6	angryBirdsStart.mp4	clickOnProgressBar	6.01241	2014-01-17 21:18:49.507
526	6	angryBirdsStart.mp4	clickOnVideo	30.0247	2014-01-17 21:18:59.393
527	6	angryBirdsStart.mp4	clickOnVideo	30.0247	2014-01-17 21:19:00.222
528	6	angryBirdsStart.mp4	clickOnVideoPlayerButton	30.0247	2014-01-17 21:19:02.97
529	6	angryBirdsStart.mp4	clickOnVideoPlayerButton	55.9701	2014-01-17 21:19:28.134
530	6	angryBirdsStart.mp4	clickOnElement	55.9701	2014-01-17 21:19:34.614
531	7	angryBirdsJReyez.mp4	clickOnVideoPlayerButton	4.20625	2014-01-17 21:19:39.158
532	7	angryBirdsJReyez.mp4	clickOnProgressBar	4.20625	2014-01-17 21:19:43.851
533	7	angryBirdsJReyez.mp4	clickOnVideoPlayerButton	34.2514	2014-01-17 21:19:52.192
534	7	angryBirdsJReyez.mp4	clickOnVideoPlayerButton	37.8121	2014-01-17 21:19:55.833
535	7	angryBirdsJReyez.mp4	clickOnElement	37.8121	2014-01-17 21:19:57.783
536	8	angryBirdsEnd.mp4	clickOnVideoPlayerButton	18.0347	2014-01-17 21:20:16.51
537	8	angryBirdsEnd.mp4	clickOnElement	18.0347	2014-01-17 21:20:17.785

Click on video:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	ClickPosition
526	6	angryBirdsStart.mp4	30.0247	2014-01-17 21:18:59.393	x:136,y:119
527	6	angryBirdsStart.mp4	30.0247	2014-01-17 21:19:00.222	x:583,y:293

Click on element:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	VideoClickposition	ElementJavascriptID	ElementStartTime	ElementStopTime	ElementStartPos	ElementEndPos
530	6	angryBirdsStart.mp4	55.9701	2014-01-17 21:19:34.614	x:480,y:192	JReyez	48	65	x:410,y:26	x:410,y:26
535	7	angryBirdsJReyez.mp4	37.8121	2014-01-17 21:19:57.783	x:368,y:77	End	37.5	38.5	x:320,y:40	x:320,y:40
537	8	angryBirdsEnd.mp4	18.0347	2014-01-17 21:20:17.785	x:333,y:130	YoutubeChannel	15	38	x:225,y:110	x:225,y:110

Click on video player button:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	Action
523	6	angryBirdsStart.mp4	0.249168	2014-01-17 21:18:41.121	Pause
524	6	angryBirdsStart.mp4	6.01241	2014-01-17 21:18:46.985	Play
528	6	angryBirdsStart.mp4	30.0247	2014-01-17 21:19:02.97	Pause
529	6	angryBirdsStart.mp4	55.9701	2014-01-17 21:19:28.134	Play
531	7	angryBirdsJReyez.mp4	4.20625	2014-01-17 21:19:39.158	Play
533	7	angryBirdsJReyez.mp4	34.2514	2014-01-17 21:19:52.192	Pause
534	7	angryBirdsJReyez.mp4	37.8121	2014-01-17 21:19:55.833	Play
536	8	angryBirdsEnd.mp4	18.0347	2014-01-17 21:20:16.51	Play

Click on progress bar:

Page: /motionTracking/git/php/usecases/usercontrollednarrative/index.php

EventID	SessionPart	Video	VideoTime	RealTime	ToTime
525	6	angryBirdsStart.mp4	6.01241	2014-01-17 21:18:49.507	30.0247
532	7	angryBirdsJReyez.mp4	4.20625	2014-01-17 21:19:43.851	34.2514

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Augmented video: 3D virtuele wereld-gekoppelde en interactieve video playback

Richting: **master in de informatica-multimedia**
Jaar: **2014**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Leën, Jeroen

Datum: **31/01/2014**