Augmented Video Viewing: Transforming Video Consumption into an Active Experience

Maarten Wijnants* Jeroen Leën† Peter Quax* Wim Lamotte*
Hasselt University – tUL – iMinds – Expertise Centre for Digital Media
Wetenschapspark 2, 3590 Diepenbeek, Belgium
*firstname.lastname@uhasselt.be, †jeroen.leen@student.uhasselt.be

ABSTRACT

Traditional video productions fail to cater to the interactivity standards that the current generation of digitally native customers have become accustomed to. This paper therefore advertises the "activation" of the video consumption process. In particular, it proposes to enhance HTML5 video playback with interactive features in order to transform video viewing into a dynamic pastime. The objective is to enable the authoring of more captivating and rewarding video experiences for end-users. The proposed paradigm extends video consumption, much like Augmented Reality (AR) does with the physical world. Given this conceptual analogy, we have adopted the term *Augmented Video Viewing* (AVV) to denominate our approach. The current AVV implementation embraces two independent yet cooperative video augmentation concepts: enriching video playback with interactive overlay elements, and erecting real-time interaction bridges between video and digital games. A total of four AVV test cases are presented to provide a hint of how our vision can be realized and of the attainable creative results.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; H.5 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia

General Terms

Design, Experimentation

Keywords

Augmented Video Viewing, interactive video, augmented video, gaming, hypervideo, web technology, web standards

1. INTRODUCTION

Upon its inception in the 1900s, the video medium was intended to yield entirely static, linear experiences. Despite its clear restrictions with respect to interactivity, this "monolithic" video production-consumption chain has been embraced by the broad public for over a century. In recent years however, consumers have become accustomed to ever improving video production values. As a result, it is becoming continually harder for video producers to attract and retain the viewer's attention. As an example, users increasingly tend to combine the watching of TV shows with other activities like, for instance, web browsing. Certain video broadcasters have recently acknowledged this multitasking trend by deploying second screen initiatives [2]. Although second screen technology has already shown some promise in captivating and more deeply engaging the viewer, it mandates the introduction of dedicated partner hardware in the video viewing setup. As such, it fails to tackle the passive nature of the video medium in an integrated fashion.

In this paper, we introduce *Augmented Video Viewing* (AVV), a paradigm that is centered around the incorporation of end-user interaction and activity in the very fabric of HTML5 video consumption. The objective here is to captivate viewers and to maximize viewer retention by delivering profound, (inter)active and highly rewarding video experiences [3]. The AVV research does not represent a single technological contribution or a tightly integrated framework; instead, it promotes a novel conceptual mindset (and accompanying web-based codebase) with regard to video viewing behavior. We describe a total of four highly diverse prototypes that practically demonstrate how video can in a meaningful manner be transformed from an exclusively passive to a much more (inter)active communication medium. The AVV mindset can be imported in a plethora of application areas and usage contexts, including consumer entertainment and recreation, video-assisted remote tutoring, and online video-based advertising [3]. Business-wise, the AVV methodology might widen a video producer's customer base and as such allow it to capitalize on new consumer profiles.

2. AUGMENTED VIDEO VIEWING

This section will discuss the two disparate yet interoperable pillars that are discernible in the AVV codebase. Both pillars have their own right of existence and are independently capable of producing compelling interactive video solutions, yet holistic advantages are likely to ensue from an intelligent blend of both components. To maximize the portability of our implementation, the AVV codebase exclusively leverages standardized web technologies.

2.1 Interactive Video Overlays

The JavaScript Augmented Video Viewing API (JAVV API) facilitates the rendering of interactive *video overlay elements* or *hotspots* on top of a HTML5 video player. Compared to the computational overhead of plain video decoding, the additional demand that the library poses on client-side hardware and software platforms is marginal.

MMSys '14, Mar 19-21 2014, Singapore, Singapore
ACM 978-1-4503-2705-3/14/03.
http://dx.doi.org/10.1145/2557642.2579368

The JAVV API has been designed with generality in mind. Overlay elements can take on arbitrary graphical appearances (via CSS), can embed custom (HTML) content, and can execute arbitrary programmatic (JavaScript) logic. In addition, an event-driven scripting model allows dedicated interaction handlers to be associated with different types of viewer actions. Finally, both spatial and temporal constraints can be attached to video overlay elements in order to control their positioning and timing, respectively.

2.1.1 Implementation

The central component of the JAVV API is a management entity that allows for the installation and destruction of video overlay elements per individual video clip. When registering an overlay with the manager, the latter instantiates a dedicated HTML DOM node and assigns it the video overlay element's content, interaction handling code and style customization. The DOM node is subsequently allocated an elevated value for its `z-index` CSS property so that the node is always visibly rendered in front of the video player. Upon unregistration, the video overlay element's associated DOM item is destructed.

The manager continuously monitors the playback time of the video clip and warrants that the spatiotemporal constraints of the registered video overlay elements are respected. Put simply, the manager guarantees that an overlay element's DOM representation is superimposed on the video at the correct location and at the appropriate time during video playback. To this end, it implements a keyframing-like animation system that performs linear interpolation between the overlay element's begin and end location during the element's visible state.

Responding to end-user interactions with video overlay elements also falls under the responsibility of the manager. The manager incorporates a dispatching model that allows dedicated scripting logic to be attached to HTML5 events that are emitted by the DOM representations of video overlay elements. No restrictions are enforced with regard to the event types that can be listened for. In a desktop application, for example, it will in many cases make sense to respond to `click`, `mouseover` and `mouseout` interactions. On (mobile) touchscreen platforms on the other hand, one will likely install callback methods for touch events, such as `touchstart` and `touchmove`. In any case, the event handling function receives a reference to the fired event and to the video overlay element that was interacted with, and can have an arbitrary implementation.

The timing, spatial positioning, style, contents and interaction handling code of video overlay elements are likely to be tailored to a specific video clip. This implies that the need arises to switch between multiple sets of overlay element definitions in case the video player supports playlists. Doing so in a JAVV API setup is straightforward, flexible and efficient. It suffices to express the video overlay element definitions that are associated with a particular video fragment in a proprietary JSON-encoded specification format, and to host the resulting file alongside the fragment itself on a webserver. Now whenever a novel source is loaded in the video player, the client just needs to destruct the currently deployed video overlay elements, fetch the new video overlay element definitions from the server, and install them. The fetching and installation steps can be implemented by means of technologies like AJAX or JSONP.

2.2 Coupling Classic Video with Game Engines

The second pillar of the AVV codebase is concerned with interweaving HTML5-driven classical video playback with (3D) virtual environments and games. The rationale here is to combine the positive assets of both media types in order to facilitate the realization of convincing and immersive experiences with advanced interactivity properties.

The most agile solution to create a hybrid video-plus-game installation that in addition prevents game engine vendor lock-in, is to adopt a loosely coupled architecture where both constituting technologies remain mutually detached, yet a bi-directional programmatic bridge is erected between them. In such a scheme, the composing technologies can evolve independently from each other and are separately exchangeable. It also allows to allocate each individual task to the technology that is most suited for handling it, which might yield benefits, both performance-wise and in terms of functional expressiveness. As an example, a video player might support hardware-accelerated video decoding, while a game engine might not. On the downside, some (minor) communication overhead and delay is clearly introduced compared to a tightly integrated setup.

2.2.1 Implementation with Unity3D

The Unity3D game engine (`http://unity3d.com/`) has impressive multi-platform publishing options. One of the supported platforms is the web: the Unity3D Web Player plug-in allows games to be played directly in web browsers. The game engine in addition provides an interface that affords full-duplex interaction between the Unity3D Web Player and the webpage that acts as its host. In particular, the encapsulating HTML page is able to transmit triggers to and invoke functions inside the embedded Unity3D Web Player, and the same holds true for the opposite direction.

2.2.2 General Purpose Implementation

Not all game engines are as generous as Unity3D in terms of offering external interfaces. To generalize our research, we have designed a generic solution that imposes no prerequisites other than HTTP support from the game engine. The proposed methodology revolves around the exchange of instructions between the HTML5 video player and the game by relaying them through an intermediate webserver. The exact modalities of the implementation can vary (e.g., server-side caching combined with client-side polling, server-side push, etcetera). Also note that the webserver not necessarily needs to be a disjoint entity; it might just as well be integrated in the game binary. As an example, a Unity3D executable is perfectly capable of hosting a WebSocket server. An integrated server solution holds the advantage that no network transmission overhead is incurred between the video player and the game, which will in turn reduce communication delay and hence increase responsiveness.

3. AVV TEST CASES

In this section, we will present a total of four AVV prototypes. The first two showcase the versatility of the JAVV API by exemplifying the heterogeneous experiences that can be authored with it. A similar role is fulfilled by the two final demonstrators, yet in these cases the potential and benefits of scaffolding interplay between respectively traditional video consumption and game environments are highlighted. All prototypes have been developed for desktop platforms

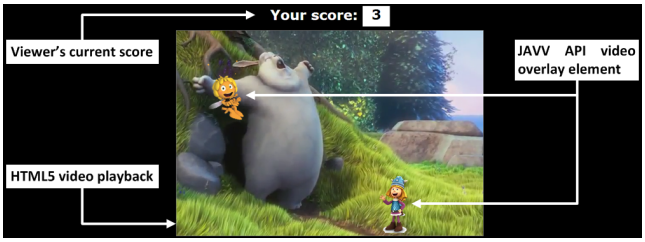

Figure 1: Click-based gameplay with HTML5 video

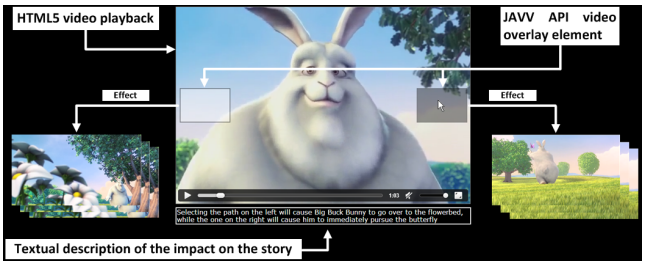

Figure 2: Non-linear, dynamic storytelling

but could easily be ported to other hardware and software environments (including portable devices with touchscreens). The prototypes are geared towards young children.

3.1 Timely Click Game

Figure 1 depicts a JAVV API-powered prototype that integrates "quick time event"-like gameplay in traditional video consumption. During playback, a number of overlay elements appear on top of the video content, at predefined locations and times. Viewers earn points by timely clicking on these interactive items. The viewer's score is visualized on the encapsulating webpage and is updated in real-time.

3.2 Interactive Video Stories

The second demonstrator exerts the JAVV API to implement an interactive storytelling use case in which the viewer is granted some degree of control over the unwinding of the plot (see Figure 2). At well-determined moments during video consumption, the playback pauses and the viewer is instructed to actively choose between a number of alternatives (represented by video overlay elements) that will define the future direction of the narrative. In the developed prototype, the video overlay elements are positioned on top of in-scene objects or regions using a semi-transparent style so that the underlying segment of the video remains visible. In situations where the contextual information that is conveyed by an overlay element itself (and the in-video object or region which it encapsulates) does not suffice to communicate its repercussions on the storyline, a short textual description is visualized when the viewer hovers his mouse over it in order to allow for informed decision making.

3.3 Video Racing

The next test case links video playback to a 3D racing game that is implemented in Unity3D and deployed inside a HTML page via the Unity3D Web Player plug-in for web browsers. The game requires players to navigate along a trajectory of checkpoints under predefined temporal deadlines. As long as players respect the imposed time limits and punctually follow the intended route, the video plays back normally. However, when the player fails to reach the next checkpoint in time, he respawns at the previous checkpoint

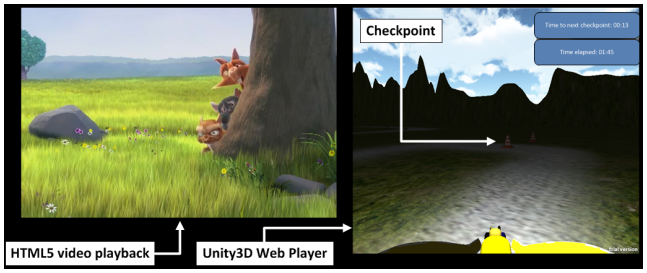

Figure 3: The Video Racing demonstrator

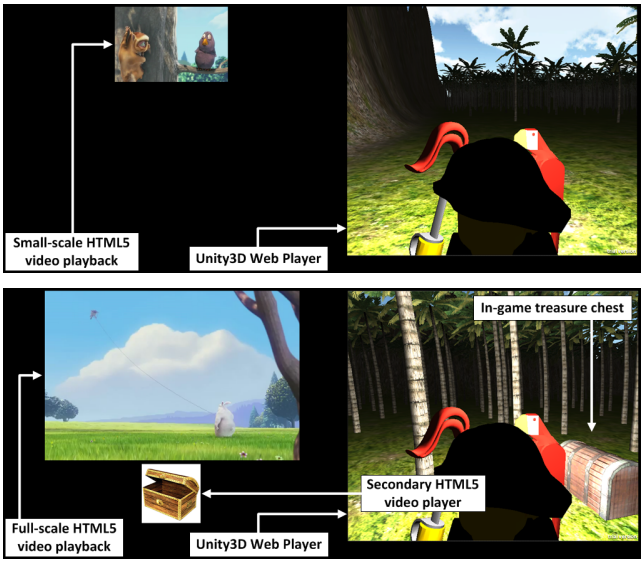

Figure 4: Dynamically dimensioned video playback

and the video playback is rewound correspondingly. At the same time, when the player diverges from the path, he has a fixed amount of time to return to it. Failure to do so results in a gameover situation and a complete halt of the video playback. To provide the player with visual feedback and a hint of the amount of time he has left to return to the trajectory, the video is increasingly obscured while the player is off-roading, up to the point where it turns completely opaque (i.e., at gameover). Figure 3 presents an annotated screenshot of this Video Racing Proof-of-Concept.

Implementation-wise, all logic and coordinating functionality is clustered inside the Unity3D application. As an example, it maintains the timers that dictate the pace at which checkpoints need to be crossed. In order to keep the state of respectively the webpage and the game logic synchronized, the Unity3D application exploits its communication interface with the embedding HTML environment (see Section 2.2.1). The game logic for instance invokes JavaScript instructions to appropriately reset the video playback when a checkpoint was missed, or to dispatch the required level of video opaqueness when the player is deviating from the racing trajectory. The video opaqueness functionality is hereby realized by overlaying a HTML5 canvas element on top of the entire video player using the JAVV API.

3.4 Video Treasure Hunting

The final demonstrator revolves around treasure hunting gameplay (see Figure 4). Here, a number of treasure chests are scattered around the virtual environment; all of them are empty, except for one, which holds the actual treasure.

Players need to locate this treasure, and the spatial dimensions of the video content thereby act as a guiding compass: video playback starts out small and gradually enlarges as the player approaches a treasure chest. The video's sound output affords similar feedback, in the sense that the audio volume is also related to the in-game distance between the player and the nearest treasure chest in a directly proportional manner. In case a chest is discovered but turns out to be empty, the video playback spatially scales down again and the sound volume is reset to its minimum value. At the same time, a secondary video player shows a video fragment of a void chest to inform the player. This process is repeated until the chest that actually contains the treasure is found.

As was the case with the Video Racing Proof-of-Concept, the game engine acts as dominant component in the implementation and orchestrates the game-video interplay.

4. RELATED WORK

SMIL (Synchronized Multimedia Integration Language) is a W3C recommended markup language that has been intentionally designed for the authoring of interactive multimedia presentations on the web (`http://www.w3.org/TR/SMIL3/`). Unfortunately, the recommendation is plagued by limited adoption by the Internet community. This manifests itself in poor native SMIL support in web browsers, which substantially hampers its practicality. A more realistically feasible category of related work is found in the academic literature on the hypervideo methodology. A hypervideo is defined as an online video document in which one or more user-clickable anchors are embedded. Notable hypervideo research efforts include *HyperCafe* [5], *Hyper-Hitchcock* and its *detail-on-demand* video concept [6], *HyLive* [1], and the *Component-based Hypervideo Model* (CHM) by Sadallah et al. [4]. Finally, at the most tangible end of the spectrum of related work, there exists a number of both free frameworks and commercial products that encompass web-based interactive and/or augmented video viewing properties. Examples include cacophony.js (`www.cacophonyjs.com`), Mozilla's Popcorn.js HTML5 media framework (`popcornjs.org`), Gravidi (`www.gravidi.com`), and YouTube Video Annotations (`www.youtube.com/t/annotations_about`).

The AVV paradigm evidently satisfies the hypervideo definition and also exhibits considerable overlaps with the cited web-based interactive video services, either conceptually, implementation-wise or both. A distinguishing feature of the AVV methodology however is its exploration of (3D) synthetic environments as a medium to "activate" classic video consumption. To the best of our knowledge, this approach has been left virtually unconsidered in academic research as well as commercialized offerings, and hence represents an important scientific contribution of our work. In addition, our AVV proposal is, one way or another, more expressive, broadly applicable, powerful and/or practical than each of the mentioned related systems. As an example, the CHM project, although being founded on a solid theoretical framework, sacrifices low-level, fine-grained control over video overlay element definition in favor of ease of specification. The AVV codebase makes no such concessions, while its specification terminology is still readily adopted by designers and artists who have only the slightest familiarity with web development. As another example, the YouTube Video Annotation tools do not include constructs to interact with the HTML page that surrounds the video player.

5. CONCLUSIONS AND FUTURE WORK

Even in this era of continuous technological innovations, only a relatively negligible minority of video content suppliers factor (inter)active features in as an essential facet in video consumption. This article has promoted the *Augmented Video Viewing* mindset, a broadly applicable paradigm that allows video viewing to be transformed into a dynamic experience. Two distinct yet interoperable AVV concepts have been introduced that are designed to fundamentally integrate (inter)activity in the video consumption process. Firstly, a JavaScript API scaffolds the extension and augmentation of a traditional HTML5 video player with overlays that enable users to interact with the video footage in a wide variety of manners. The second procedure consists of exploiting full-duplex programmatic bridges between game engines and (external) HTML5-based video visualizations to incorporate elaborate gameplay features in video viewing environments. We claim that, either separately or combined, these concepts afford content suppliers the necessary toolbox to author compelling, arousing and richly interactive video experiences for end-users. This proposition has been enforced by the presentation of a total of four largely divergent prototypes that give a taste of the myriad of creative possibilities (with regard to the "activation" of video viewing) that are unlocked by the AVV implementation.

A first future research direction involves the implementation of a logging and analytics framework that anonymously records the (inter)actions performed by AVV consumers (to enable quantitative analysis and mining of users' AVV behavior). Secondly, we plan to organize user studies to collect qualitative feedback regarding the AVV paradigm in general, and the developed prototypes in particular.

6. ACKNOWLEDGMENTS

Part of the research described in this article was performed in the context of the iMinds MiX project Wanagogo. This project is cofunded by iMinds (Interdisciplinary institute for Technology), a research institute founded by the Flemish Government, and by IWT.

7. REFERENCES

[1] P. Hoffmann, T. Kochems, and M. Herczeg. HyLive: Hypervideo-Authoring for Live Television. In *Changing Television Environments*, volume 5066 of *Lecture Notes in Computer Science*, pages 51–60. 2008.

[2] N. Narasimhan. When the Shift Hits the (Television) Fan: A Growing Opportunity for Companion Devices. *IEEE Internet Computing*, 15(5):83–86, September-October 2011.

[3] A. A. Raney, L. M. Arpan, K. Pashupati, and D. A. Brill. At the Movies, on the Web: An Investigation of the Effects of Entertaining and Interactive Web Content on Site and Brand Evaluations. *Journal of Interactive Marketing*, 17(4):38–53, Autumn 2003.

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

[5] N. Sawhney, D. Balcom, and I. Smith. Authoring and Navigating Video in Space and Time. *IEEE MultiMedia*, 4(4):30–39, October-December 1997.

[6] F. Shipman, A. Girgensohn, and L. Wilcox. Combining Spatial and Navigational Structure in the Hyper-Hitchcock Hypervideo Editor. In *Proc. Hypertext 2003*, pages 124–125, Nottingham, UK, August 2003.