Curtains Up! Lights, Camera, Action! Documenting the Creation of Theater and Opera Productions with Linked Data and Web Technologies

Thomas Steiner, Rémi Ronfard, Pierre-Antoine Champin, Benoît Encelle, Yannick Prié

Curtains Up! Lights, Camera, Action! Documenting the Creation of Theater and Opera Productions with Linked Data and Web Technologies

Thomas Steiner[1⋆], Rémi Ronfard[2] Pierre-Antoine Champin[1], Benoît Encelle[1], and Yannick Prié[3]

[1]CNRS, Université de Lyon, LIRIS – UMR5205, Université Lyon 1, France
{tsteiner, pierre-antoine.champin}@liris.cnrs.fr, benoit.encelle@univ-lyon1.fr
[2] Inria Grenoble Rhône-Alpes / LJK Laboratoire J. Kuntzmann - IMAGINE, France
remi.ronfard@inria.fr
[3]CNRS, Université de Nantes, LINA – UMR 6241, France
yannick.prie@univ-nantes.fr

Abstract. For this paper, in the context of the French research project *Spectacle en Ligne(s)*, we have recorded the entire set of rehearsals of one theater and one opera production using state-of-the-art video equipment. The resulting raw video and audio tracks as well as manually generated annotation data were then preprocessed in order to localize actors and detect their dialogues. Based on these preprocessing steps, we have built a Web-based hypervideo application that allows for navigation through performance time, performance space, and rehearsal time using modern HTML5 Web technologies like the emerging Web Components standard. We publish and consume the annotation data as so-called Linked Data Fragments, a novel way to make triple-based structured data available in a scalable way. As a direct outcome, researchers interested in the genetic analysis and the creation process of live performances can, thanks to this application, freely zoom in and out of scenes, rehearsal sessions, and stage locations in order to better understand the different steps on the way to a chef d'œuvre. A live demo of the application is publicly available at the URL http://spectacleenlignes.fr/hypervideo/.

Keywords: Hypervideo, Web Components, Linked Data Fragments, video analysis, audio analysis, theater, opera, rehearsal, live production

1 Introduction

1.1 Project Background

The objective of the *Spectacle en Ligne(s)*[1] project is to create a video corpus of live theater and opera rehearsals and to explore the uses of this archive for pedagogic, research, and mediation purposes. The project is funded by the French

⋆ Second affiliation: Google, Hamburg, Germany, tomac@google.com
[1] Project website: http://spectacleenlignes.fr/

National Agency of Research (ANR) as part of the project call *"Corpus, data and research tools in human and social sciences"*.[2] Adopting an interdisciplinary approach, the project is structured around three complementary areas of research: *(i)* sociological research for the study of public and existing performance archives, *(ii)* technological research for the chained capturing and publishing of challenges of Open Access, *(iii)* mediation research of audiences for the design of new usage scenarios of the archive. The project ended in December 2014.

1.2 Hypervideo Background

The term *hypervideo* is commonly used to refer to *"a displayed video stream that contains embedded user-clickable anchors"* [25,26] and annotations, allowing for navigation between the video and other hypermedia elements. In a 2006 article in *The Economist*, the authors write *"[h]yperlinking video involves the use of "object-tracking" software to make filmed objects, such as cars, clickable as they move around. Viewers can then click on items of interest in a video to watch a related clip; after it has played, the original video resumes where it left off. To inform viewers that a video is hyperlinked, editors can add highlights to moving images, use beeps as audible cues, or display still images from hyperlinked videos next to the clip that is currently playing"* [30]. In standard literature, hypervideo is considered a logical consequence of the related concept of *hypertext* [3]. In contrast to hypertext, hypervideo necessarily includes a time component, as content changes over time. In consequence, hypervideo has other technical and aesthetic requirements than hypertext, the most obvious one being appropriate segmentation in scenes or even objects. The opportunities for feature-rich semantic hypervideos are endless, only limited by feasibility and ease of their creation. In this paper, we share our approach to affordably and practically document the creation of theater and opera productions with video and Web technologies.

1.3 Paper Contributions

Our contributions with this paper are two-fold. First, we show how modern HTML5 [2] Web technologies and the emerging Web Components [7] standard can be used for the documentation of theater and opera productions; the resulting hypervideo Web Components[3] [28] as well as the demo application[4] created for *Spectacle en Ligne(s)* based thereon are made available publicly as open source. Second, we make use of Semantic Web technologies, namely Linked Data Fragments [34], to publish *and* consume[5] the annotation data that was created during the recording phase. This approach allows us to make our structured data reusable by others as Linked Data [4] on the one hand, and shows its feasibility by "eating our own dog food" through using this data ourselves on the other.

[2] *French: "Corpus, données et outils de la recherche en sciences humaines et sociales"*
[3] Polymer Hypervideo: `https://github.com/tomayac/polymer-hypervideo`
[4] *Spectacle en Ligne(s)* demo application: `spectacleenlignes.fr/hypervideo/`
[5] *Spectacle en Ligne(s)* data portal: `http://spectacleenlignes.fr/query-ui/`

2 Related Work

Related work can be regarded under the angles of online video annotation creation, large-scale Linked Data efforts for video, and video documentation of theatrical performances. Many have combined Linked Data and video, typical examples are [15] by Lambert *et al.* and [13] by Hausenblas *et al.* There are several text track enriching approaches [18,19,20,27] based on named entity recognition. The online video hosting platform YouTube lets publishers add video annotations in a closed proprietary format. From 2009 to 2010, YouTube had a feature called Collaborative Annotations [1] that allowed video consumers to collaboratively create video annotations. In [32], Van Deursen *et al.* present a system that combines Media Fragments URI [31] and the Ontology for Media Resources [17] in an HTML5 Web application to convert media fragment annotations into a WebVTT [23] file that can be used by HTML5-enabled players. Building on their work, in [29], we additionally allowed for writing annotations by letting annotators create WebVTT cues with an editor. Popcorn.js[6] is an HTML5 JavaScript media framework for the creation of media mixes by adding interactivity and context to videos by letting users link social media, feeds, visualizations, *etc.* to moving images. PopcornMaker[7] is an interactive Web authoring environment allowing for videos to be annotated on a timeline. McAuley reports in [22] findings from ten years of experimentation with recording formats and analysis for the documentation of theatrical performances. In [16], Lan and Morgan investigate the effects of retroactive and focused self-monitoring through videotaping, on children's theater performance and found that retroactive self-monitoring enhanced theater performance. Giesekam examines in [9] the use of film and video in theaters and evaluates the impact and effectiveness of such developing multimedia technologies on practices in dramaturgy and performance.

3 Hypervideo Web Components

In this section, we first provide necessary background on the emerging Web Components standard and then describe the generic hypervideo Web Components that were created in the context of the *Spectacle en Ligne(s)* project.

3.1 Introduction to Web Components

Web Components is a set of specifications, which let Web developers leverage their HTML, CSS, and JavaScript knowledge to build widgets that can be reused easily and reliably.[8] According to a (recently discontinued) W3C Working Draft introductory document,[9] the component model for the Web ("Web Components") consists of five different pieces that we will list in the following.

[6] Popcorn.js: `http://popcornjs.org/`
[7] PopcornMaker: `https://popcorn.webmaker.org/`
[8] Web Components: `http://www.chromium.org/blink/web-components`
[9] Discontinued W3C Working Draft document: `http://www.w3.org/TR/2013/WD-components-intro-20130606/` [7]

Imports which defines how templates, decorators and custom elements are packaged and loaded as a resource [12].
Shadow DOM which encapsulates a DOM subtree for more reliable composition of user interface elements [11].
Custom Elements which let authors define their own elements, with new tag names and new script interfaces [10].
Decorators which apply templates based on CSS selectors to affect rich visual and behavioral changes to documents.
Templates which define chunks of inert markup that can be activated for use.

At time of writing, partial native support for Web Components has landed in a number of Web browsers, however, for the majority of browsers, a so-called polyfill solution is still required. A polyfill is a piece of code that provides the technology that developers expect the browser to provide natively in the near future. We rely on the Polymer project[10] that provides Web Components support for older browsers. Polymer allows us to create reusable widgets that introduce a number of new custom HTML elements for our task of hypervideo creation.

3.2 Implementation Details

We have developed a number of Web Components for the creation of hypervideos. These Web Components are behaviorally grouped together by a common naming convention. In Polymer, all element names have to start with the prefix `polymer` and contain a dash in order to add a namespace which avoids conflicts with existing elements. However, this requirement is seen as a bad practice by some, as it makes Web Components seem like being "owned" by the Polymer framework.

`<polymer-hypervideo>` is the parent element of all other elements. It accepts the attributes `src` for specifying a set of space-separated video sources (to support different encodings), and—analog to the native HTML5 video attributes—`width` and `height` for specifying the video's dimensions, then `poster` for specifying the video's poster frame, and finally `muted` to specify if the video should be initially muted.
`<polymer-data-*>` is a set of data annotation elements that includes the two shorthand annotation types `<polymer-data-actor>` for annotating video actors and `<polymer-data-overlay>` for annotating visual overlays, and the generic `<polymer-data-annotation>` for other annotations.
`<polymer-track-*>` are the two elements `<polymer-track-chapters>` and `<polymer-track-subtitles>`, which rely on WebVTT [23] text tracks of the types "chapters" and "subtitles" that they enrich with automatically generated chapter thumbnails and a full text subtitle view.
`<polymer-visualization-*>` currently provides the following two visualization elements `<polymer-visualization-timeline>` on the one hand and `<polymer-visualization-toc>` on the other that create a timeline view and a table of contents that put all encountered `<polymer-track-*>` and `<polymer-data-*>` elements in a temporal context.

[10] Polymer project: `http://www.polymer-project.org/`

We have made an online demo application available at `http://hypervideo.herokuapp.com/demo.html` that showcases these Web Components and recall that we share their implementation as open source. As both native Web Component support in Web browsers and the Polymer project are constantly evolving and still in flux, the demo currently works best on the latest versions of the Chrome browser. A screenshot of the application can be seen in Figure 1, the corresponding underlying code sample is shown in Listing 1. These components communicate with each other through standard JavaScript events, so when a components needs to communicate its state to another, *e.g.*, `<polymer-hypervideo>` the current time of the video to one of the visualization components like the `<polymer-visualization-timeline>`, it fires an event that components can subscribe to and react upon. Listing 2 shows the relevant code snippets.

3.3 Evaluation of the Web Components Design Choice

The creation of new Web Components is a not too difficult task for an experienced Web developer. Especially Polymer's `<seed-element>`[11] makes getting started straight-forward. The biggest challenge and at the same time the biggest chance is the dynamic nature of Web Component development. Support for Web Components has partially landed natively in Web browsers, which means the polyfill has to do less and less work emulating native support. Existing bugs are generally fixed in a timely manner in the frequent new releases of Polymer. Communication between Web Components can be subject to race

[11] Polymer `<seed-element>`: `https://www.polymer-project.org/docs/start/reusableelements.html`

```
<polymer-hypervideo src="big_buck_bunny.mp4␣big_buck_bunny.webm"
     width="400" height="225" muted>

  <polymer-data-actor start="10" end="35" name="Chirp␣(Owl)" xywh="170,20,70,80"
       url="http://commons.m.wikimedia.org/wiki/File:Chirp1_-_BBB_-
            _reduced_snapshot.png">
  </polymer-data-actor>

  <polymer-track-subtitles src="subtitles.vtt" displaysubtitlesgroup>
  </polymer-track-subtitles>

  <polymer-track-chapters src="thumbs.vtt" displaychaptersthumbnails>
  </polymer-track-chapters>

  <polymer-visualization-timeline orientation="landscape">
  </polymer-visualization-timeline>

</polymer-hypervideo>
```

Listing 1: Web Components mark-up for the hypervideo in Figure 1, including subtitles, chapters, timeline, and table of contents; the actor annotation contains a spatial fragment (`xywh`) [31] and a link (`url`) to Wikimedia Commons

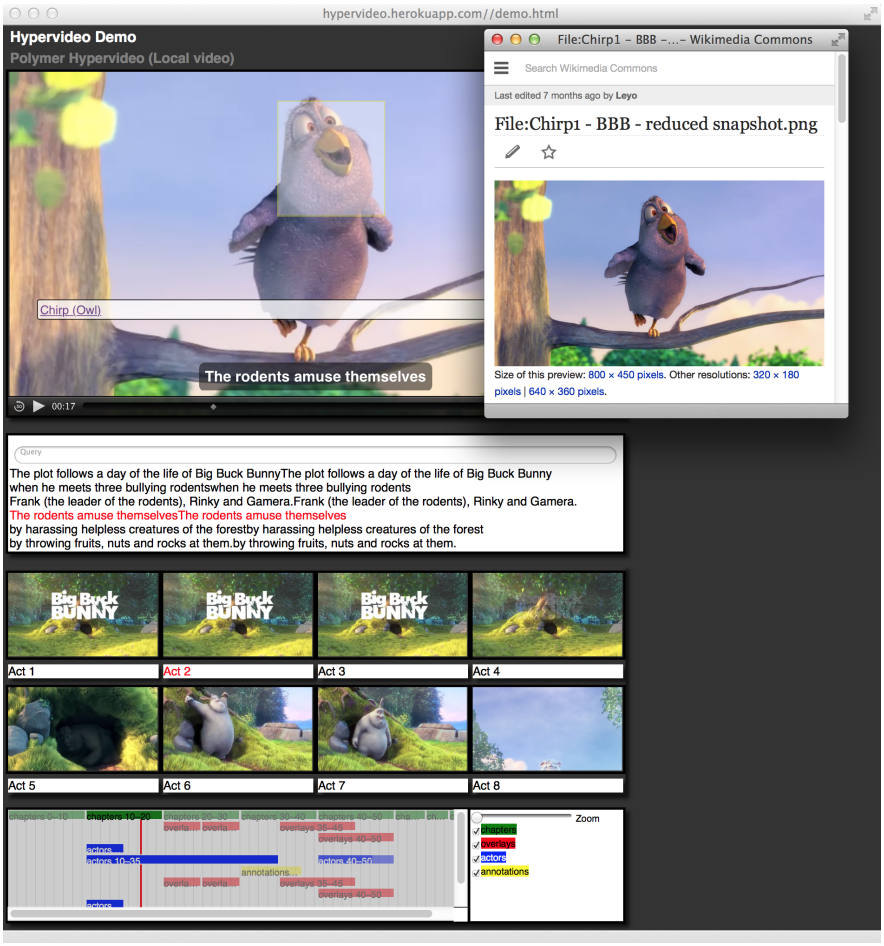


Fig. 1: Generated hypervideo based on the mark-up in Listing 1, including subtitles, chapters, timeline, and table of contents; the actor annotation contains a spatial fragment [31] (owl's head) and a link to a Wikimedia Commons page

conditions, as event listeners may not yet have been created at the time an event is being sent. Especially with dynamically created Web Components this can be an issue, also across browsers. We had to introduce short timeouts for certain Web Components before they propagate their properties up to their parent Web Component. Concluding, Web Components were nevertheless the right design choice. The ease of use of the finished Web Components (see Listing 1) and the fact that Web Components can be created and interacted with using JavaScript (see Listing 2 and Listing 4) just like regular HTML elements greatly outweigh the initial development efforts as we will show in more detail Subsection 5.3.

4 Linked Data Publication and Consumption

In this section, we first introduce the concept of Linked Data and Tim Berners-Lee's Linked Data principles, and then Ruben Verborgh's Linked Data Fragments. This finally leads us to the *Spectacle en Ligne(s)* Linked Data portal.

4.1 Introduction to Linked Data

Linked Data [4] defines a set of agreed-on best practices and principles for interconnecting and publishing structured data on the Web. It uses Web technologies like the Hypertext Transfer Protocol [8] and Unique Resource Identifiers (URIs, [5]) to create typed links between different sources. The portal `LinkedData.org` defines Linked Data as being "about using the Web to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods." Tim Berners-Lee defined the four rules for Linked Data in a W3C Design Issue as follows.

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

Linked Data uses the Resource Description Framework (RDF, [14]) to create typed links between things in the world. The result is often-times referred to as the Web of Data. RDF encodes statements about things in the form of triples that take the form subject, predicate, object. In this context, Heath and Bizer [6] also speak of RDF links, *i.e.*, dereferencing the URI that appears as the destination of a link yields a description of the linked resource in question.

```
// === In polymer-hypervideo.js: ===
// listen to native html5 video timeupdate events
video.addEventListener('timeupdate', function() {
  that.currentTime = video.currentTime;
  // publish hypervideotimeupdate events
  that.fire('hypervideotimeupdate', { currentTime: that.currentTime });
});


// === In polymer-visualization-timeline.js: ===
// listen to hypervideotimeupdate events
document.addEventListener('hypervideotimeupdate', function(e) {
  var currentTime = e.detail.currentTime;
  // update the time marker
});
```

Listing 2: Native JavaScript event communication between Web Components

4.2 Linked Data Fragments

Various access mechanisms to Linked Data exist on the Web, each of which comes with its own trade-offs regarding query performance, freshness of data, and server cost/availability. To retrieve information about a specific subject, you can dereference its URL. SPARQL [24] endpoints allow to execute complex queries on RDF data, but they are not always available. While endpoints are more convenient for clients, individual requests are considerably more expensive for servers. Alternatively, a data dump allows interested parties to query locally. However, especially with dynamic datasets, data dumps risk to get outdated quite quickly. Users then always have to download the entire updated data dump again or work with data diffs. Linked Data Fragments [34] provide a uniform view on all such possible interfaces to Linked Data, by describing each specific type of interface by the kind of fragments through which it allows access to the dataset. Each fragment consists of three parts that we will list in the following.

data: all triples of this dataset that match a specific selector;
metadata: triples that describe the dataset and/or the Linked Data Fragment;
controls: hypermedia links/forms that lead to other Linked Data Fragments.

This view allows to describe new interfaces with different trade-off combinations. One such interface is triple pattern fragments [33], which enables users to host Linked Data on low-cost servers with higher availability than public SPARQL endpoints, or even for free [21] on consumer cloud storage products. Such a light-weight mechanism is ideal to expose mid-size datasets on commodity hardware in a scalable and ad hoc manner using triple pattern fragments.

4.3 Linked Data Portal

We use the Linked Data Fragments server implementation Server.js[12] by Ruben Verborgh. Installing the server requires Node.js 0.10 or higher and is tested on OSX and Linux. We also make a browser interface available that is based on the Linked Data Fragments client implementation Browser.js[13] by Ruben Verborgh. We expose the data at the URL `http://spectacleenlignes.fr/query-ui`, which is host to a user interface that allows for SPARQL queries to be executed. A screenshot of the query interface can be seen in Figure 2.

4.4 Linked Data Fragments Web Component

Verborgh's original `ldf-client` library was written for a Node.js environment. We have compiled it using browserify,[14] a tool that allows modules designed for Node.js to be used from a Web browser context. This allows us to query our Linked Data portal from a browser context using a declarative Web Component

[12] Server.js: `https://github.com/LinkedDataFragments/Server.js`
[13] Browser.js: `https://github.com/LinkedDataFragments/Browser.js`
[14] Browserify: `http://browserify.org/`

called <polymer-ldf-client>[15] that we have also released as open source. The HTML markup of a Web Component that allows for declaratively accessing the data in the Linked Data portal is depicted in Listing 3, the JavaScript source code for obtaining streaming and polling data is shown in Listing 4.

[15] <polymer-ldf-client>: https://github.com/tomayac/polymer-ldf-client

```
<polymer-ldf-client
   id="polymer-ldf-client-streaming"
   responseFormat="streaming"
   query="SELECT DISTINCT ?frag WHERE {
      ?a a &lt;http://advene.org/ns/cinelab/ld#Annotation&gt; ;
        &lt;http://advene.org/ns/cinelab/ld#hasFragment&gt; ?frag ;
        &lt;http://advene.org/ns/cinelab/ld#taggedWith&gt;
          [ &lt;http://purl.org/dc/elements/1.1/title&gt; 'personnages: Maggie'],
          [ &lt;http://purl.org/dc/elements/1.1/title&gt; 'personnages: Brick'];
        .
    }"
   startFragment="http://spectacleenlignes.fr/ldf/spectacle_en_lignes">
 </polymer-ldf-client>
```

Listing 3: Linked Data Fragments Web Component <polymer-ldf-client>

```
var ldfClient = document.querySelector('#polymer-ldf-client-streaming');

/* Streaming example */

// Process data as it appears
ldfClient.addEventListener('ldf-query-streaming-response-partial', function(e) {
  alert(e.detail.response);
});

// Get notified once all data is received
ldfClient.addEventListener('ldf-query-streaming-response-end', function() {
  alert('Received all data');
});


/* Polling example */

// Poll for data
ldfClient.addEventListener('ldf-query-polling-response', function(e) {
  alert(e.detail.response);
});

// Manually trigger data polling
var button = document.querySelector('#button');
button.addEventListener('click', function() {
  ldfClient.showNext();
});
```

Listing 4: Obtaining streaming and polling data from the Linked Data Fragments Web Component <polymer-ldf-client>

4.5 Evaluation of the Linked Data Fragments Design Choice

Linked Data Fragments turned out to be a good design choice. Our mid-size dataset fits the use case perfectly well. Building upon Linked Data Fragments' promise to keep the server simple and enable smart clients, we can support complex queries on our dataset even on commodity hardware. While the processing speed may not in all cases compete with a performant native SPARQL query engine, the streaming nature of Linked Data Fragments results delivery allows the user to see and process partial results as they come, or simply to wait for the complete result. Our `<polymer-ldf-client>` Web Component supports both kinds of operation through a `responseFormat` attribute.

5 The *Spectacle en Ligne(s)* Demo Application

We recall the objective of the *Spectacle en Ligne(s)* project, which is to create a video corpus of live theater and opera rehearsals and to explore the uses of this archive for pedagogic, research, and mediation purposes. The demo application should facilitate the navigation through performance time, performance space, and rehearsal time of the recorded œuvres. It was built on top of the hypervideo Web Components that were introduced in Section 3 and uses the project's Linked Data portal described in Section 4 through a dedicated Web Component. From the recording phase, we have several video tracks for each rehearsal day as well as WebVTT text tracks and manual annotations from the preprocessing phase. Manual annotations mainly contain act and scene data, sparse mise en scène data, and in some cases information on the acting persons in a particular scene.

5.1 Data Flow

Starting with a list of all available videos (the demo application does not contain the full list of all recordings), we first obtain the relevant WebVTT text track for the selected video and via JavaScript create an empty `<polymer-hypervideo>` container. The text track is of type chapters (see [23] for available types) and is converted to a `<polymer-track-chapters` element that gets appended to the `<polymer-hypervideo>` container. We interpret chapters as text cues in the œuvres, which allows us to navigate directly into them. All chapters from the `<polymer-track-chapters` element are automatically displayed on a dynamically inserted `<polymer-visualization-timeline>` element. In continuation, we then query the Linked Data portal for all annotations available for this video using a dynamically inserted `<polymer-ldf-client>` Web Component. Incoming annotations are converted to `<polymer-data-annotation>` elements that are then placed on the `<polymer-visualization-timeline>` element. Finally, we obtain a WebVTT text track of type subtitles (again see [23]) that contains the spoken text of each text cue of the œuvre in the video in question. We dispose of HTML documents of the recorded œuvres that show the text and mise en scène instructions in a human-friendly way. Using common CSS selectors, we identify text cues in this document and align them with the text cue

data from the WebVTT subtitles text track. This allows us to visually highlight text cues in the human-friendly documents upon cue change events that are sent by the `<polymer-hypervideo>` element. Figure 3 shows all Web Components and the human-readable documents in action for three different iterations.

5.2 Enabled Navigation Patterns

As outlined earlier, we have several videos for each rehearsal day. Actors rehearsed the acts of each œuvre on different days and not necessarily in chronologically correct order. In the simplest form, we allow for consuming the videos in sequential order to revive the rehearsal days and to see act by act, scene by scene, text cue by text cue how the actors and the metteur en scène worked on them. Each annotation, represented through yellow blocks in the timeline beneath the video in Figure 3, acts as a hyperlink that makes the video jump right to the annotation's time code. The same holds true for the chapter annotations that represent the text cues, displayed as green blocks in the timeline. Most interesting to the user probably is the navigation by text cue, where the user can see how a certain text cue evolved over time. Figure 3 shows act 1, cue 7.9 on three different days, starting with the first lecture of the text at a table on day 1, over to an early rehearsal on day 8 on a private stage, and ending with the technical mise en scène on day 40 on the public stage. Upon mouse-over on the main video, a hovering semi-transparent navigation control gets displayed that lets the user navigate to the next or previous rehearsal day, or the next or previous text cue. Additionally, three select boxes on top of the main video allow for focused by-text-cue, by-video, and by-rehearsal-day navigation.

5.3 Evaluation of the Demo Application

Through our groundwork with the hypervideo and Linked Data Fragments Web Components, building the final application was a rather straight-forward task. All that was missing was the application logic that orchestrates the different Web Components. The whole application required no more than 500 lines of JavaScript code.[16] As we make all source codes of the involved Web Components and the application itself available, future reuse of our work in other theater or opera performances is rendered possible. The opera and theater partners of the *Spectacle en Ligne(s)* project particularly appreciated the complete freedom of navigation in the whole recording archive. By dissolving the temporal order of the rehearsals in the application, they could analyze and study the progress of the different scenes, acts, and even text cues on a level of detail not seen before.

[16] *Spectacle en Ligne(s)* demo application: `https://github.com/tomayac/postdoc/tree/master/demos/polymer-hypervideo/spectacle-en-lignes`

6 Conclusions and Future Work

In this paper, we have first described the objectives of our project *Spectacle en Ligne(s)*. Second, we have introduced the concept of hypervideo, followed by a look at related works in the areas of online video annotation creation, large-scale Linked Data efforts for video, and video documentation of live theatrical and musical performances. In continuation, we provided necessary background on the emerging Web Components standard. As the first contribution of our paper, we have implemented and evaluated hypervideo Web Components that can be used to create hypervideos in a declarative way through nothing but custom HTML tags. Objects or temporal points of interest in the hypervideos can be annotated, such annotations then appear on an interactive timeline. We have then looked at Linked Data sharing principles and introduced Linked Data Fragments as an appropriate way to share our annotation data for others and ourselves to consume in a scalable manner. The second contribution is a Web Component that allows for interacting with Linked Data Fragments in a streaming or polling way again purely declaratively. Finally, we have combined all generated Web Components in a demo application that showcases the power and simplicity of our approach. The source codes of all involved Web Components and the application itself are available as open source to encourage broad reuse.

Future work will mainly concentrate on the hypervideo Web Components. We will evaluate their usefulness for further use cases like, for example, Web video courses and improve them accordingly if necessary. As we have repeatedly expressed, the Web Components standard is still in flux. As more and more Web browsers will gain native support for Web Components, existing timing challenges that we encountered occasionally will be resolved. Work on more hypervideo features has already started like camera selection or inter-hypervideo communication for synchronized hypervideo experiences. On the Linked Data side, we want to further improve the retrieval speed of Linked Data Fragments in our Web Component by allowing for parallel multiplexed query requests.

Concluding, we are content of how far the limits of the Web have been pushed. Tasks like hypervideo creation were a thing of native applications that did not allow for their results to be easily shared on the Web. Through our hypervideo Web Components, creating a basic hypervideo has become accessible enough that standard Web developers can do it. The promise of Linked Data in the end are synergies between data that was not interlinked before. By following Linked Data principles and thanks to our Linked Data Fragments Web Component, these synergies can be reached in the context of a Web application with very reasonable effort, as we have shown in this paper. We encourage others to reuse our Web Components in their own applications and to further extend and improve them. We cannot wait to see what they will create.

— Curtains down. And cut.

References

1. S. Bar et al. YouTube's Collaborative Annotations. In *Webcentives '09, 1st International Workshop on Motivation and Incentives*, pages 18–19, 2009.
2. R. Berjon, T. Leithead, E. D. Navara, et al. HTML5: a vocabulary and associated APIs for HTML and XHTML. Working Draft, W3C, 2012.
3. T. Berners-Lee. WorldWideWeb: Proposal for a HyperText Project. Proposal, W3C, 1990. `http://www.w3.org/Proposal.html`.
4. T. Berners-Lee. Linked Data, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`, accessed July 15, 2013.
5. T. Berners-Lee, R. T. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, IETF, 2005.
6. C. Bizer, T. Heath, and T. Berners-Lee. Linked data – the story so far. *International Journal On Semantic Web and Information Systems*, 5(3):1–22, 2009.
7. D. Cooney and D. Glazkov. Introduction to Web Components. Working Draft, W3C, June 2013. `http://www.w3.org/TR/components-intro/`.
8. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, 1999.
9. G. Giesekam. *Staging the Screen: The Use of Film and Video in Theatre*. Palgrave Macmillan, 2007.
10. D. Glazkov. Custom Elements. Last Call Working Draft, W3C, Oct. 2013. `http://www.w3.org/TR/custom-elements/`.
11. D. Glazkov and H. Ito. Shadow DOM. Working Draft, W3C, June 2014. `http://www.w3.org/TR/shadow-dom/`.
12. D. Glazkov and H. Morrita. HTML Imports. Working Draft, W3C, Mar. 2014. `http://www.w3.org/TR/html-imports/`.
13. M. Hausenblas, R. Troncy, Y. Raimond, and T. Bürger. Interlinking Multimedia: How to Apply Linked Data Principles to Multimedia Fragments. In *Linked Data on the Web Workshop (LDOW 09), in conjunction with the 18th International World Wide Web Conference (WWW 09)*, 2009.
14. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommendation, W3C, 2004.
15. D. Lambert and H. Q. Yu. Linked Data based Video Annotation and Browsing for Distance Learning. In *SemHE '10: The Second International Workshop on Semantic Web Applications in Higher Education*, 2010.
16. W. Y. Lan and J. Morgan. Videotaping as a means of self-monitoring to improve theater students' performance. *The Journal of Experimental Education*, 71(4):371–381, 2003.
17. W. Lee, W. Bailer, T. Bürger, et al. Ontology for Media Resources 1.0. Recommendation, W3C, Feb. 2012. `http://www.w3.org/TR/mediaont-10/`.
18. Y. Li, G. Rizzo, J. L. Redondo García, R. Troncy, M. Wald, and G. Wills. Enriching Media Fragments with Named Entities for Video Classification. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 469–476, 2013.
19. Y. Li, G. Rizzo, R. Troncy, M. Wald, and G. Wills. Creating Enriched YouTube Media Fragments with NERD Using Timed-Text. In *11th International Semantic Web Conference (ISWC2012)*, November 2012.
20. Y. Li, M. Wald, T. Omitola, N. Shadbolt, and G. Wills. Synote: Weaving Media Fragments and Linked Data. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *LDOW*, volume 937 of *CEUR Workshop Proceedings*, 2012.

21. L. Matteis and R. Verborgh. Hosting queryable and highly available Linked Data for free. In *Proceedings of the ISWC Developers Workshop 2014*, Oct. 2014.
22. G. McAuley. The video documentation of theatrical performance. *New Theatre Quarterly*, 10:183–194, 5 1994.
23. S. Pfeiffer and I. Hickson. WebVTT: The Web Video Text Tracks Format. Draft Community Group Specification, W3C, Nov. 2013.
24. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Recommendation, W3C, 2008.
25. N. Sawhney, D. Balcom, and I. Smith. HyperCafe: Narrative and Aesthetic Properties of Hypervideo. In *Proceedings of the the Seventh ACM Conference on Hypertext*, HYPERTEXT '96, pages 1–10, New York, NY, USA, 1996. ACM.
26. J. Smith and D. Stotts. An Extensible Object Tracking Architecture for Hyperlinking In Real-time and Stored Video Streams. *Department of Computer Science, University of North Carolina at Chapel Hill, Technical Report*, 2002.
27. T. Steiner. SemWebVid – Making Video a First Class Semantic Web Citizen and a First Class Web Bourgeois. In *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010*, volume 658 of *CEUR Workshop Proceedings*, pages 97–100, Nov. 2010.
28. T. Steiner, P.-A. Champin, B. Encelle, and Y. Prié. Self-Contained Semantic Hypervideos Using Web Components. In R. Verborgh and E. Mannens, editors, *ISWC Developers Workshop 2014*, CEUR-WS, pages 96–101, Oct. 2014.
29. T. Steiner, H. Mühleisen, R. Verborgh, P.-A. Champin, B. Encelle, and Y. Prié. Weaving the Web(VTT) of Data. In *Proceedings of the 7th Workshop on Linked Data on the Web*, Apr. 2014.
30. The Economist. From Hypertext to Hypervideo, Sept. 2006. `http://www.economist.com/node/7904166`.
31. R. Troncy, E. Mannens, S. Pfeiffer, et al. Media Fragments URI 1.0 (basic). Recommendation, W3C, Sept. 2012. `http://www.w3.org/TR/media-frags/`.
32. D. Van Deursen, W. Van Lancker, E. Mannens, et al. Experiencing Standardized Media Fragment Annotations Within HTML5. *Multimedia Tools and Applications*, pages 1–20, 2012.
33. R. Verborgh, O. Hartig, D. Meester, et al. Low-cost queryable Linked Data through triple pattern fragments. In *Proceedings of the 13th International Semantic Web Conference: Posters and Demos*, volume 1272, pages 13–16, Oct. 2014.
34. R. Verborgh, O. Hartig, D. Meester, et al. Querying datasets on the Web with high availability. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *Proceedings of the 13th International Semantic Web Conference*, volume 8796 of *Lecture Notes in Computer Science*, pages 180–196. Springer, Oct. 2014.

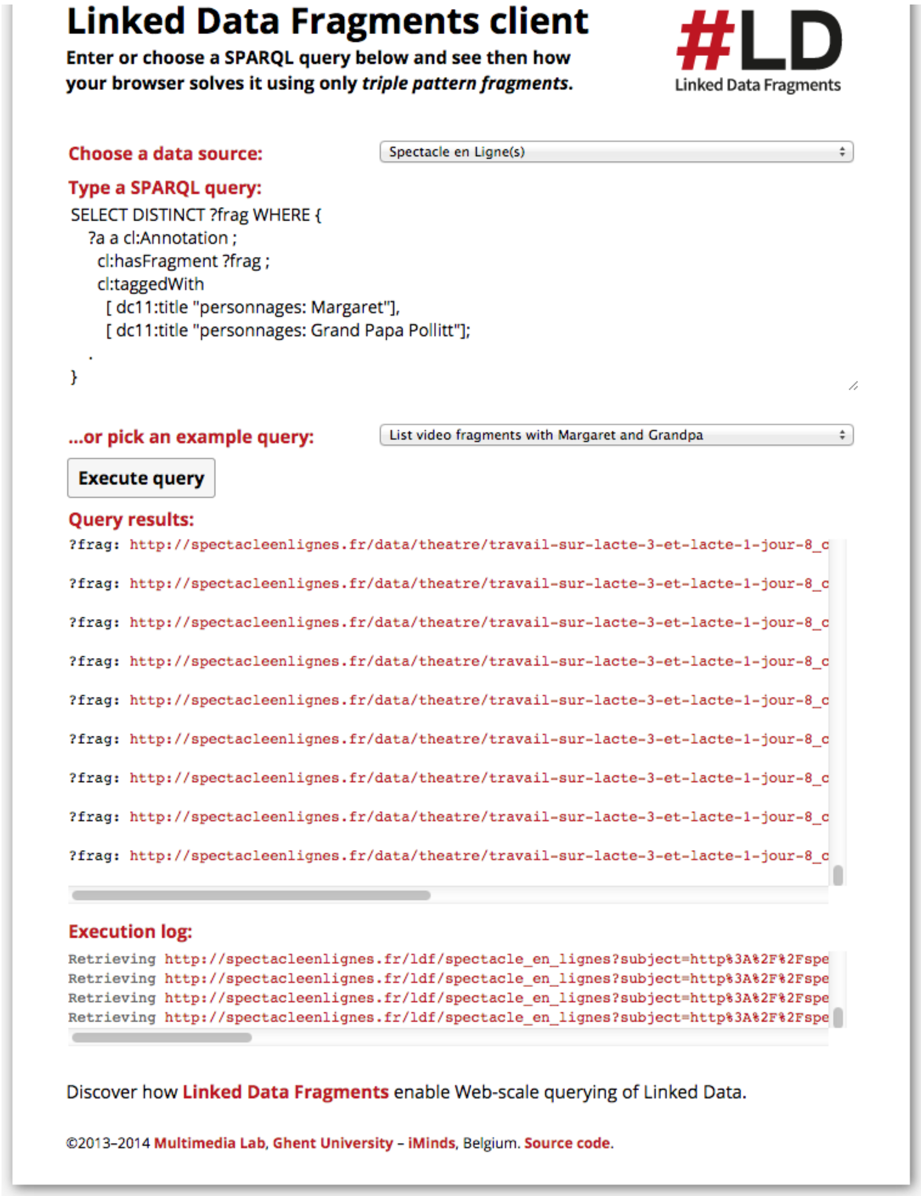
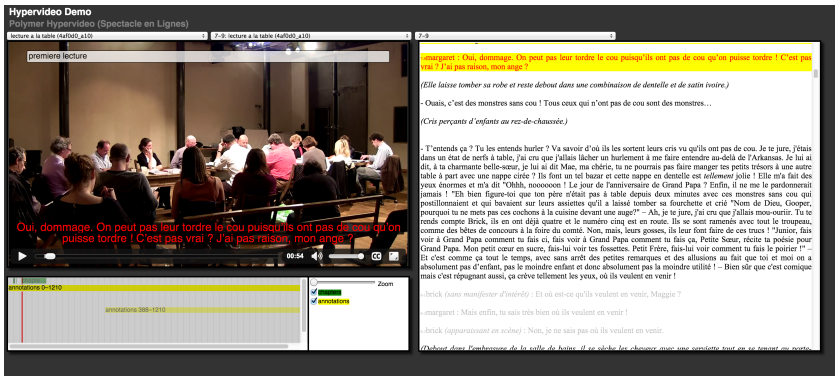
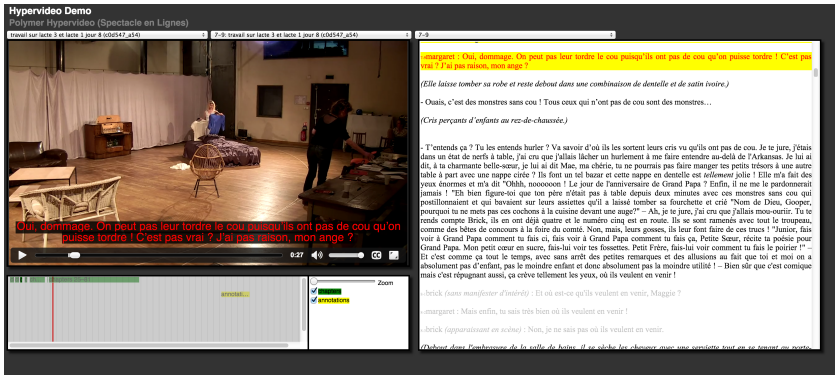


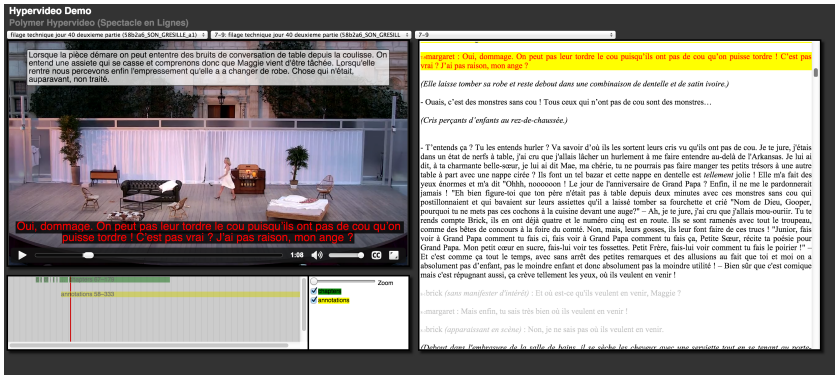
Fig. 2: Linked Data portal for the project *Spectacle en Ligne(s)*: `http://spectacleenlignes.fr/query-ui/`



(a) Day 1, first lecture at the table, `http://spectacleenlignes.fr/hypervideo/#lecture-a-la-table_4af0d0_a10/7-9`



(b) Day 8, rehearsals of act 1, `http://spectacleenlignes.fr/hypervideo/#travail-sur-lacte-3-et-lacte-1-jour-8_c0d547_a54/7-9`



(c) Day 40, technical mise en scène, `http://spectacleenlignes.fr/hypervideo/#filage-technique-jour-40-deuxieme-partie_58b2a6_SON_GRESILLE_a1/7-9`

Fig. 3: Evolution of act 1, cue 7.9 of T. Williams' *Chatte Sur Un Toit Brulant*