

Annotating Proprietary Streaming Video

Open Annotation Collaboration (OAC) Annotation Demonstration Experiment Report

James Smith
MITH, University of Maryland

1. Executive Summary

MITH and Alexander Street Press collaborated on an experiment to provide web-based annotation capabilities for streaming video targeting time intervals as well as areas within the video frame with the ability to import and export annotations using the Open Annotation data model. We have developed a set of developer tools that enable anyone to add an annotation client in their web page and allow annotation of embedded streaming video. The tools provide import/export capabilities for annotations using the Open Annotation data model in RDF/JSON. While the library supports only HTML5 video for now, it has a well-defined API for adding support for additional video players.

2. Use Case Context

Video annotation presented a unique challenge among the experiments in that the annotations targeted two different constraints: time interval and spatial. Other experiments used constraints that required a single constraint, such as an area of an image or a span of text in a document.

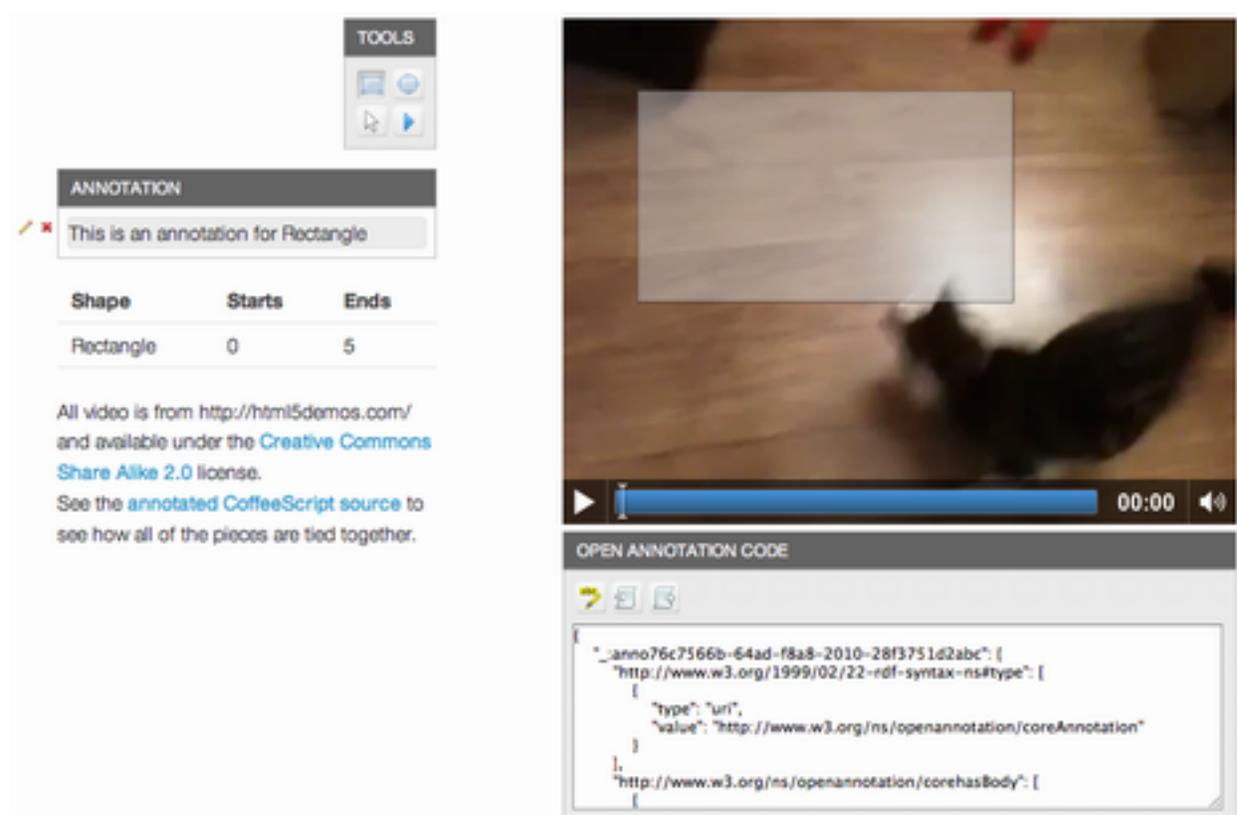
Alexander Street Press publishes a number of video collections using a proprietary video player that protects their intellectual property contained in the videos. The video annotation experiment had to respect that protection while allowing annotations.

3. Annotation Types

This experiment explored annotating video with text, but the toolkit can be extended to allow arbitrary bodies as long as appropriate display and editing capabilities are provided. The annotations target a timespan of the video and a shape within the play surface that can be described using an SVG shape or path. The toolkit provides support for rectangles and ellipses, but the API allows the web developer to add other shapes by defining handlers that parse or construct the SVG constraint and draw the shape on the play surface.

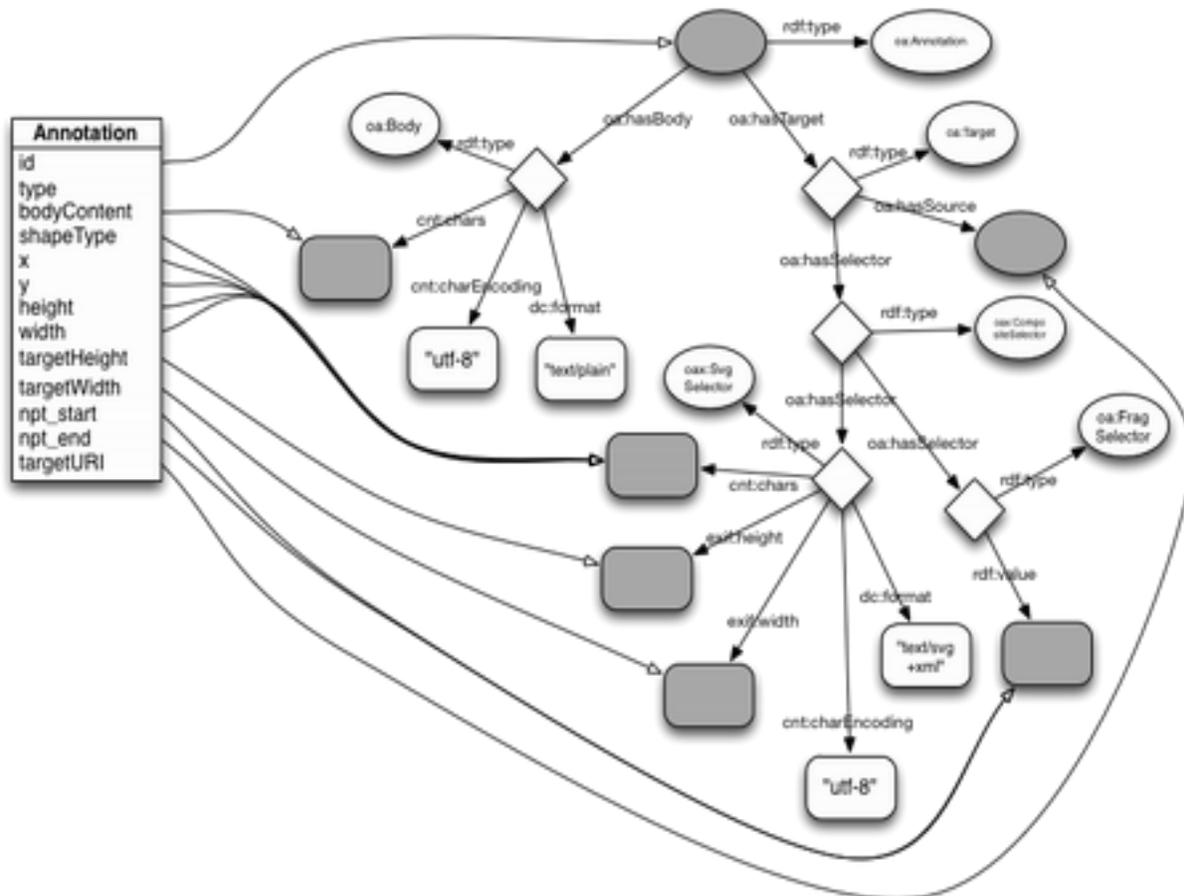
4. Example Annotations

Four components make up the interface for managing annotations: the Tools box, the Annotation box, and Open Annotation Code box, and the annotation presentation over the video play surface. The tools allow the selection of the shape type to draw, the use of the mouse as a selection and editing tool, and video playback. The Annotation box allows editing of the annotation body and deletion of the annotation. The selection tool allows selection of a shape in the play surface and editing of its extents and position. The video playback tool allows playback of the video regardless of the accessibility of the controls provided by the video player.



Here, the annotation consists of the combination of the target shape and play position visible in the video player and the comment body in the Annotation box.

A turtle serialization of the annotation is provided in Appendix A. The following image shows how the internal data model of the annotation client maps to the Open Annotation data model expressed as RDF.



In the demonstration application, we assume that the annotation body will be simple text. By adding a bodyType attribute to the internal data model, the annotation application could support arbitrary body content types.

The other attributes describe the temporal and spatial constraints on the target part of the video. The shapeType, x, y, height, and width attributes are used to build the SVG element describing the spatial constraint. The targetHeight and targetWidth attributes are used to describe the extents of the play surface for which the SVG constraint applies. The npt_start and npt_end attributes are used to build the fragment selector describing which time segment the annotation targets.

We believe that most applications will work like this, translating between an internal data model designed for the application and the Open Annotation data model designed for exchange with other applications.

5. Summary of Progress

The video annotation developers toolkit and accompanying demonstration show how to provide web-based annotation of proprietary video through the use of video player drivers that isolate

the proprietary nature of the video player from the annotation management and display. The demonstration lacks controls for editing the time aspect of the annotation. All annotations are assumed to be for five seconds starting at the time of the play head when the annotation is created. The annotation toolkit API allows for additional controls and presentations, but these details are out of scope for this document. They will be documented on the Github site for the code.

At the beginning of the project, we hoped to make use of the player controls as a way to provide editing of the annotation's time span in the video. However, we found that we have limited access to player controls, or to knowledge of where they might be on the play surface. With HTML 5, browsers are free to render controls as they see fit. Because we couldn't count on having this information for one video player type, we abandoned our expectation of having it for any video player type. HTML 5 is too integrated into the web for us not to support it as a standard video player type. By the end of the project, we didn't have time to implement custom time editing controls, but we didn't need to in order to test the affordances of the OAC data model for annotating streaming video.

Several technical hurdles were not considered in scope for the core parts of the software.

Handling communication between the browser and the server went out of scope as we shifted to a generalized toolkit. Instead, we provide an API that allows the import and export of the RDF/JSON representation of the annotations. The developer using the toolkit can transport these between the browser and server in a way that fits the context in which they are using the toolkit.

We demonstrated early in the project that we could pass control information between the toolkit's execution context and any of three other contexts: same execution context as the toolkit, the context of a Flash player, and the execution context of an embedded iframe. Limited developer resources precluded us building support for all three into the initial HTML 5 video driver.

We were able to move from the earlier Open Annotation Collaboration data model to the W3C Open Annotation data model with ease. All of the results in this report and the code in the toolkit use the newer data model.

6. Discussion of Results and Conclusions

6.1 Resolving the Appropriate Target URI for HTML5 Video

The Open Annotation data model doesn't have a good way to specify the resolution of a target video. The data model assumes that a target URL uniquely identifies a resource's representation or that the state of a resource can be used to identify to which representation the annotation should apply. This simply doesn't work with streaming video.

The first example of a situation that Open Annotation can not accommodate is the [HTML5 video tag](#). The specification states that the following is a valid use of the tag to embed a video in a web page (slightly simplified for illustrative purposes):

```
<video>
  <source src="http://www.example.com/foo" />
  <source src="http://www.example.net/bar" />
</video>
```

The browser should use the first source that it knows how to use because the author is asserting that these sources are different representations of the same video resource.

What is the target URL for the video?

Perhaps we produce an annotation with two targets, one for each source. The browser will only load one of them, so we only have the resolution of one of the videos. We don't know the extents of the play surface for the other video. OAC requires that SVG constraints be in terms of the pixel size of the video we are annotating without any reference to the size of the play surface. How do we know that the second video resource will have the same extent as the first so that our annotations will target the same part of the second video resource as the first?

If we only target the video resource that the browser plays, then how do we know to show our annotations if another user uses a browser that plays the other resource? How do we know where to target areas of the play surface?

Even with a single stream, how do we handle annotations on a play surface that is resizable?

Our video annotation tool solves both of these problems by choosing not to address them head on.

We require a 'data-oactarget' attribute on the <video/> tag that specifies the URL we are targeting in our annotations. We assume that the web page author knows how to name embedded videos uniquely so that we can track which annotations apply to which embedded videos.

6.2 Scaling Target Areas

We also add exif:height and exif:width tags to the SVG constraint to specify the extents of the play surface we are annotating. We scale the annotation bounding boxes as needed to match the extents of the SVG constraint with those of the play surface.

We recommend that the Open Annotation group pay closer attention to the streaming video use case instead of considering it an edge case with little use. Video is one of the mainstream media formats used on the web that has regular need for multiple constraints in targets. It also

presents a fluid specification with respect to extents, with browsers able to play video in full size, full screen, stretched, and original aspect ratio formats, all with different extents requiring scaling of sub-frame constraints. Consumers consider all of these different formats to represent the same video. They will expect annotations to understand this equivalence.

6.3 Generalizable Results and Conclusions

The Open Annotation data model is reasonable for exchanging annotations between applications. A few areas may need some further work to aid specific use cases, but the framework supports augmenting the standard data model as needed by user communities. While such augmentation could lead to fragmentation, it can also show where future standards might evolve if managed properly through extension namespaces and proper scoping.

The experiment showed that it is possible to respect proprietary interfaces to content and still allow annotation of that content. The annotation does not need access to the actual content being annotated as long as it has sufficient information to address the target such that the body of the annotation can be associated with the target in a reproducible fashion.

We also showed that the annotation application does not need to use the Open Annotation data model as its internal working data model. It is possible to translate between the application's data model and the Open Annotation data model. This was expected since the Open Annotation data model is designed for exchange of annotations, not the manipulation of annotations.

Acknowledgements

Appendix A - Annotation RDF Serialization

The following prefixes apply for all serializations:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix cnt: <http://www.w3.org/2011/content#>.
@prefix oa: <http://www.w3.org/ns/openannotation/core/>.
@prefix oax: <http://www.w3.org/ns/openannotation/extension/>.
@prefix exif: <http://www.w3.org/2003/12/exif/ns#>.
```

A.1 Example Annotation from Section 4

The following represents the example annotation shown in section 4. The content in bold represents information held in the annotation application's internal data model.

```
_:annotation
  rdf:type      oa:Annotation ;
```

```

    oa:hasBody    _:body ;
    oa:hasTarget  _:target .
_:body
  rdf:type          oa:Body ;
  dc:format         "text/plain" ;
  cnt:chars        "This is an annotation for Rectangle" ;
  cnt:characterEncoding "utf-8" .
_:target
  rdf:type          oa:SpecificResource
  oa:hasSource     <http://html5demos.com/assets/dizzy> ;
  oa:hasSelector   _:selector .
_:selector
  rdf:type          oa:CompositeSelector ;
  oa:hasSelector   _:svgSelector ;
  oa:hasSelector   _:fragSelector .
_:svgSelector
  rdf:type          oa:SvgSelector ;
  dc:format         "text/svg+xml" ;
  cnt:chars        "<rect x=187 y=126.5 width=268 height=151 />" ;
  cnt:characterEncoding "utf-8" ;
  exif:height      "360" ;
  exif:width       "480" .
_:fragSelector
  rdf:type          oa:FragSelector
  rdf:value        "t=npt:0,5"

```