code{4}lib JOURNAL

Issue 31, 2016-01-28

Video Playback Modifications for a DSpace Repository

This paper focuses on modifications to an institutional repository system using the open source DSpace software to support playback of digital videos embedded within item pages. The changes were made in response to the formation and quick startup of an event capture group within the library that was charged with creating and editing video recordings of library events and speakers.

This paper specifically discusses the selection of video formats, changes to the visual theme of the repository to allow embedded playback and captioning support, and modifications and bug fixes to the file downloading subsystem to enable skip-ahead playback of videos via byte-range requests.

This paper also describes workflows for transcoding videos in the required formats, creating captions, and depositing videos into the repository.

by Keith Gilbertson and Liz McVoy

Motivation

VTechWorks is the institutional repository of the University Libraries of Virginia Tech, which launched in late 2011 using the DSpace software. At launch, the plan for VTechWorks was for it to act as a general-purpose repository for digital collections at the library, including theses and dissertations, scholarly articles, historical images and digital scans of cookbooks from the special collections department, and digitized materials from the archives. During the mass import of legacy content, it was noticed that several digital video files had entered into the collection, including promotional videos for the cooperative extension office and tribute videos that were sent to Virginia Tech after the campus mass shooting tragedy of April 16, 2007.

We had interesting video content in the repository, but the initial use cases for DSpace institutional repositories focused primarily on textual content: for example, collections of journal article pre-prints in PDF format. DSpace has been described as largely format agnostic, in that it allows users to deposit any type of file and to make these various file types available for download, if not in-browser display. However, the repository group at Virginia Tech thought that it would be fun to allow playback of videos from within the item page. Also, popular video streaming sites such as YouTube have conditioned users to expect embedded video playback in the web browser, instead of clicking on a video filename to download it first. Thus, video playback in DSpace was quickly hacked together, and the initial release of the video playback project in VTechWorks happened in August of 2012. Closed caption and subtitle support was added in November of 2012, in anticipation of future needs. The service was advertised in internal (but public) blog postings (Video playback …, 2012; VTechWorks Closed Caption …, 2012), with a few sample videos converted to the required formats.

By advertising the capability on a blog posting and listing the format requirements, the hope was that others would continue to submit video content to VTechWorks. There was collaboration with computer science students for a class project to record computer science seminars, but outside of this successful project, few new videos were submitted to VTechWorks.

Then, in 2013, the library event capture group was formed. The event capture group provides, free of charge, video recording services for scholarly events up to two hours in length (Event Capture, 2015). From the time of its formation, this group has deposited videos of over one hundred events to VTechWorks, with more pending.

Functionality

The project allows for playback of digital videos within a viewer embedded in the item pages of VTechWorks. The customized software detects whether an item has videos encoded in the necessary formats, and if so, renders an HTML5 video player within the web browser. If the web browser does not support HTML5 video, then a fallback option using the Adobe Flash player is presented.

Our implementation of this project uses the video.js (Video.js Player, 2015) player to customize the embedded HTML5 video player and to present the Flash fallback option in the theme. With theme changes, the project is also compatible with alternative players, such as Flowplayer, or can be made to rely entirely on the built-in controls for video playback in HTML5 capable web browsers. The system supports the use of a “Movie Poster”, an image file that is displayed in the embedded viewer before playback is requested. This can be a still from an interesting frame of the video, or a title frame for the video, and is meant to avoid display of a blank canvas in the video playback area.

Playback support is intended to work on popular browsers on popular operating systems, including Firefox, Internet Explorer, Chrome, and Opera on OS X, Windows, and Linux, and on mobile devices based on Android and iOS operating systems. Some basic requirements of video playback are supported; for example, the video playback can begin even as just the beginning of the file has been downloaded, and the viewer can skip ahead to view parts of the video that have not yet been downloaded. These features are especially important for mobile devices, which may have limited storage space.

If a WebVTT file is present in the item, the file is used to provide text indexing of the video for the search discovery system, and to present subtitles on screen.

Selection of Video Formats

After the event capture group became involved and began producing videos frequently, more thought was put into standardizing the video formats for ease of processing. In this project, the video formats can be conceived of as three types.

1. The original format, as captured by the camera
2. The preservation format, an archival copy that can used to create other formats
3. Presentation formats, used for display in the web browser

The preservation format was selected after research by the video event capture group at the time of the initial implementation. Members were Liz McVoy, Scott Pennington, and Therese Walters. The event capture group produces the preservation copy after the video has been edited. The specification is as follows:

Preservation Format:
Multimedia container file: .mov
Video Codec: ProRes 422
Resolution: 1920×1080

The presentation formats were selected based on the capability of web browsers at the time of the original implementation, and based on experimentation. Browser support for HTML5 video has improved greatly, but there are still differences between browsers and platforms. An updated chart of HTML5 compatible multimedia formats on both mobile and desktop is on the Mozilla Developer Network. Some web browsers, such as Safari on OS X and iOS, natively supported mp4 files with h264 encoding, while other browsers, such as Chrome, supported webm files with VP8 encoding. Therefore, each video has two presentation files stored in VTechWorks, an mp4 file and a webm file.

Presentation Format, .mp4:
Multimedia container file: .mp4
Video Codec: h264 video, two-pass encoding
Resolution: 854×480 target (do not scale up if the original is smaller)
Audio: AAC
Pixel format: yuv420p
Moov atom: Located at front of file
Bitrate: 1200 kbps variable

Presentation Format, .webm:
Multimedia container file: .webm
Video Codec: vp8
Resolution: 854×480 target (do not scale up if the original is smaller)
Audio: Ogg Vorbis
Pixel format: yuv420p
Bitrate: 1200 kbps variable

There are some peculiarities to these specifications. The pixel format of “yuv420p” was selected for compatibility with Google Chrome for Windows. When the videos were encoded with other pixel formats, they did not play back on Chrome in Windows. Our specification for the .mp4 file states that the moov atom should be located at the beginning of the file. The moov atom serves as a time-based index to the rest of the file. If the moov atom is not located at the beginning of the file, then skip-ahead functionality is not enabled, and in some cases the video will not play at all. (Placing moov atom … , 2010)
If an mp4 file and a webm file are both present in an item, the video playback features will be enabled in the theme. In addition to the video files, it is possible to include other files to support optional features in the playback functionality. Image files support thumbnails and movie posters; webvtt files support subtitles.

Thumbnail:
File format: .jpg
File naming convention: video_filename.mp4.jpg, video_filename.webm.jpg
Size: 100 pixels, maximum width
Location: THUMBNAIL bundle in VTechWorks

Movie Poster:
File format: .jpg
File naming convention: video_filename.jpg
Size: Same dimensions as 1 video frame (target 854×480)
Location: MOVIEPOSTER bundle in VTechWorks

DSpace organizes bitstreams, or files, into logical groups called bundles. Typical bundles in a DSpace repository include ORIGINAL, THUMBNAIL, and LICENSE. The MOVIEPOSTER bundle is a special bundle that was added for this project.

Subtitles:
File format: .webvtt, specification at http://dev.w3.org/html5/webvtt/
Location: ORIGINAL bundle in VTechWorks

With the current implementation, the video playback functionality will only detect and use one webvtt file for each item. Due to this limitation, it is not currently possible to display subtitles in an alternate language, for example.

Code Changes

Overview

The code changes that were necessary to support this feature modified several subsystems of DSpace, including the file download system, web theming, and configuration of indexing and file types. Code additions are stored in the VTUL (Virginia Tech University Libraries) group GitHub repository.

File type Registry

DSpace utilizes a file format registry that lists file types known to DSpace. Registry entries for the mp4, webm, and WebVTT file formats were added to the registry, bitstream-formats.xml.

```
   <bitstream-type>
   <mimetype>video/mp4</mimetype>
   <short_description>MP4 Container</short_description>
   <description>MP4 Container format for video files</description>
   <support_level>0</support_level>
   <internal>false</internal>
   <extension>m4v</extension>
   <extension>mp4</extension>
</bitstream-type>

<bitstream-type>
   <mimetype>video/webm</mimetype>
   <short_description>webm video</short_description>
   <description>The webm video container format</description>
   <support_level>0</support_level>
   <internal>false</internal>
   <extension>webm</extension>
</bitstream-type>

<bitstream-type>
   <mimetype>text/vtt</mimetype>
   <short_description>WebVTT caption file</short_description>
   <description>Closed caption or subtitle file for HTML5 video</description>
   <support_level>1</support_level>
   <internal>false</internal>
   <extension>vtt</extension>
</bitstream-type>
```

These entries in the file type registry allow the theme to detect the presence of these files and to customize its view accordingly. The xml file shown here will configure file entries for new DSpace installations. For previously installed repositories, there is a web interface to add new format entries.

Theming

VTechWorks uses the XMLUI user interface for DSpace 5.4. This interface uses XSLT to transform XML representations of DSpace objects and metadata into HTML for display in a web browser. The XMLUI allows themes to customize views for the entire repository, or for individual collections. The current theme for VTechWorks is a lightly customized version of a theme created with responsive design principles, called “Mirage2”, from the @mire company. The main customization to the Mirage2 theme for this project is a modification to the item view page that displays an embedded video playback area when the user is looking at video items. There are also changes to include and style the video.js plugin.

First, the page-structure.xsl file was modified to include the necessary css and Javascript files for the video.js tool.

```
<!-- include css and javascript for video playback -->
<link type="text/css" rel="stylesheet">
    <xsl:attribute name="href">http://vjs.zencdn.net/c/video-js.css</xsl:attribute>
</link>
<script src="http://vjs.zencdn.net/c/video.js"> </script>
```

The page links to the files on the video.js project server, instead of linking to copies that could be stored on the VTechWorks server. This potentially introduces performance issues, or instability when the files are modified. However, it also allows VTechWorks to automatically and effortlessly benefit from the latest improvements to the video.js project.

The item-view.xsl file checks for the existence of an mp4 and a webm file, and if and only if both exist, displays the video playback area in the item view page. If a webvtt file is provided for subtitles, or an image file for use as a movie poster, these are also detected and handled. The code uses the standard HTML5 video, source, and track elements. Video.js detects these elements and customizes them accordingly.

```
   <!-- V.T. Add HTML for the video in a videojs frame -->

          <xsl:if test="(./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='video/mp4']) and (./mets:fileSec/me

               <!-- V.T. Best guess at aspect ratio of most of our videos -->
               <video controls="controls" preload="none" width="853" height="480" class="video-js vjs-default-skin" data-setu

                   <xsl:if test="./mets:fileSec/mets:fileGrp[@USE='MOVIEPOSTER']">
                       <xsl:attribute name="poster">
                           <xsl:value-of select="./mets:fileSec/mets:fileGrp[@USE='MOVIEPOSTER']/mets:file/mets:FLocat[@LOCTY
                       </xsl:attribute>
                   </xsl:if>

                   <source type="video/webm" >
                   <xsl:attribute name="src">
                       <xsl:value-of select="./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='video/webm']/me
                   </xsl:attribute>
                   </source>

                  <source type="video/mp4">
                  <xsl:attribute name="src">
                      <xsl:value-of select="./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='video/mp4']/mets
                  </xsl:attribute>
```

```
                    </source>

                   <!-- V.T. display captions -->
                   <xsl:if test="./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='text/vtt']">

                   <track kind="captions" srclang="en" label="English" default="default">
                   <xsl:attribute name="src">
                   <xsl:value-of select="./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='text/vtt']/mets:FLoca
                   </xsl:attribute>
                   </track>
                   </xsl:if>

                  </video>

 <hr />

             </xsl:if>
```

The theme is not able to detect if the moov atom is located at the front of the mp4 file, as is required per proper operation. The site superuser.com has suggestions about how to detect the location of the moov atom from scripts. Also, all videos are given a viewport of the same dimensions, regardless of their aspect ratio. In our current revision, this size is set to 854×480, which matches the dimensions of our current presentation formats in VTechWorks, and is a 16/9 aspect ratio.

BitstreamReader

In DSpace repositories using the XMLUI interface, there is a Java class called the BitstreamReader that is responsible for reading deposited files from storage and sending the files back to the requesting web browser. The BitstreamReader class at one time contained byte range support that enabled downloading of arbitrary portions of files. In the stock DSpace distribution, this code had been commented out because it was reported to cause problems with some downloads, in particular with some external PDF viewers.
Changes to the BitstreamReader.java file include enabling these byte range downloads again, and fixing bugs related to arithmetic and HTTP headers.

HTTP byte range offsets start with 0; this led to an off-by-one error that was corrected:

```
/* entityRange = byteRange.intersection(
                    new ByteRange(0, this.bitstreamSize)).toString(); */

// this code fixes off by 1 error in commented line above
requestedRange = byteRange.intersection(
                     new ByteRange(0, this.bitstreamSize - 1));
                entityRange = requestedRange.toString();
```

It was also discovered that the Content-Length HTTP header was missing, and that the word “bytes” was absent from the Content-Range header.

```
//response.setHeader("Content-Range", entityRange + "/" + entityLength);
// V.T. fix for headers
response.setHeader("Content-Length",""
                       + requestedRange.length());
response.setHeader("Content-Range",
                      ”bytes " + entityRange + "/" + entityLength);
```

Properly functioning byte range downloads are necessary to enable viewers to skip ahead to a portion of the video that has not yet been downloaded, and for allowing playback on mobile devices, which typically have little storage space. After these problems were corrected, mobile device playback and the skip-ahead feature functioned properly.

At the time of publication of this article, another problem was discovered with byte range downloads. See the “Problems” section for more details.

Transcoding scripts

During the earliest stages of experimentation, videos were encoded on an individual basis as the embedded playback feature was tested on different file types. After the event capture group was started, videos were added to the repository more often. The preservation video format had been standardized, and it was an appropriate time to attempt automation for transcoding to the presentation formats.

The event capture group was created at a time when system administrator availability was difficult to come by in the library. A Mac mini and several external drives were purchased to act as a makeshift file server. A perl script was hacked together (Garbage Scripts, 2015) to create the mp4 and webm presentation files from the preservation copy, along with the thumbnails and movie poster file. The “-movflags faststart” option is used to make sure that the moov atom is placed at the beginning of the file.

This script was retired at the same time as the makeshift file server was retired and moved to a production appropriate system, and video transcoding is now handled with iffmpeg, a closed-source, commercial GUI front-end for the ffmpeg encoder that runs on OS X.

The iffmpeg program meets all of our technical needs and is more flexible, but due to a shortage of available staff time, there is now a demand for the script to be returned to service. This is more challenging than expected because of the need to find an available server node with large storage capabilities and high compute capability.

Configuration

The DSpace configuration file, dspace.cfg, has an entry that lists bundle names that appear in the web interface for item uploads.

```
xmlui.bundle.upload = ORIGINAL, METADATA, THUMBNAIL, MOVIEPOSTER, LICENSE
```

The modified line shown here adds the MOVIEPOSTER bundle so that the Movie Poster files can be uploaded from the web interface.

DSpace has a set of media filters that process common file types such as images, to build thumbnails, and PDF files, for full text indexing. Even though WebVTT is not an HTML file, we can configure the HTMLFilter to provide full text indexing of the subtitles:

```
filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats = HTML, Text, t
```

Testing

The very first item to serve as a test for video in VTechWorks is a tribute video that was presented to Virginia Tech.

Figure 1. Screenshot of Embedded Tribute Video

The video playback functionality was tested on the most popular platforms, including Android and iOS, and Safari, Chrome, Firefox, and Opera on OS X, and Firefox, Chrome, and Internet Explorer 6, 8, and 9 on Windows.

Multiple versions of Internet Explorer were tested, because the HTML5 <video> tag was not supported until Internet Explorer 9, but Internet Explorer 6 was still in wide use, even in computers within the library. For browsers that do not support HTML5, such as IE6, video.js provides a fallback mode that uses Adobe Flash. HTML5 video is now widely supported among commonly used browsers, and Flash appears to be on the decline.

During debugging, curl, as well as the Live HTTP Headers plugin for Firefox were both used extensively.

Deposit Workflow

After editing and exporting the preservation copies of the videos and saving them on a file share with nightly backups, the event capture group places a copy of the preservation file into a special file share folder (ir_prepare/incoming_mov) for transcoding into presentation formats, as described into the “transcoding scripts” section.

The outputs from the transcoding section, a webm file, an mp4 file, and jpeg thumbnails and movie posters, are placed into a different folder (ir_prepare/ready_for_deposit) on the file share. Previously, perl scripts were used to run the ffmpeg tools. Currently, iFFmpeg, an OS X GUI front-end for ffmpeg is utilized by the event capture group.

All videos are deposited to the repository by either members of the event capture group or members of the VTechWorks group, using the web upload interface built into DSpace.

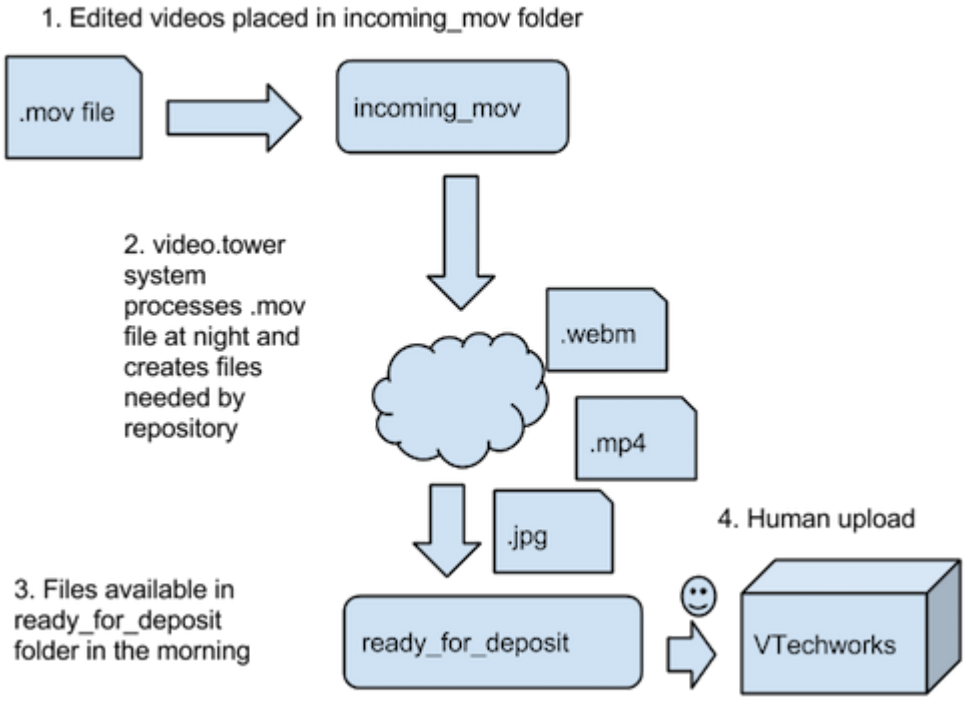

Figure 1. Video Workflow.

Subtitle workflow

At present, only three videos produced by the library have been captioned. Captions are created in the WebVTT format. WebVTT is a text-based format with time codes and corresponding words to be displayed on the screen (WebVTT, 2015) during each of these time segments.

```
01:49.025 --> 01:55.039
WE WILL PREVAIL
WE ARE VIRGINIA TECH!
```

The WebVTT file for the initial captioning experiment was created with the Microsoft Caption Maker web tool. In this tool, the user provides a link to a video file. The user can see the video in the web browser while typing in captions, and can pause, rewind, and play the video. After the video has been viewed and captioned, the user can download a WebVTT file containing the time codes and titles. Captioning is a time-consuming process, and could easily take half of a working day or more to caption a one hour video.

When the second and third videos were captioned, the Microsoft Caption Maker tool was temporarily unavailable. Instead, a similar tool called Amara was used. The Amara tool is operated much like Microsoft Caption Maker, and also has the very useful feature to save work still in progress to a user account. There's also an additional step in the Amara process that allows the user to adjust the synchronization of the titles on screen by dragging bars in the web interface.

After the WebVTT file is created, it is uploaded to the DSpace item via the web interface. The theme detects the existence of the WebVTT file and displays captions, which can be disabled via a button in the playback interface.

Problems

Before this project was implemented, some videos had already been deposited into VTechWorks. The embedded playback feature doesn't work with these videos, because the theme customization requires that both an mp4 and a webm file are present to display the embedded player. The video files are still available for download, even though they can't be played back directly on the item page. There is currently no plan to correct these pre-existing files. Automating the process is not a very easy matter because the files are not in standard, identical formats – and preservation or original copies often are not locatable. Pre-existing video items are instead occasionally encoded on an item-by-item basis into the required formats as desired.

Some of the problems that we have experienced are related to the byte range request feature which had been disabled in DSpace, that we had enabled specifically for this project.

Recently, after our video encoding and deposit workflows had changed, and we experimented with encoding at a higher resolution, two larger video files were deposited into VTechWorks where playback was broken. If videos aren't encoded into the specific formats, with the moov atom at the front of the h264 file, this can cause problems with embedded playback. These particular files, though, not only had problems with embedded playback, they were also unable to be downloaded to client machines. The files were both larger than two gigabytes. The byte range request feature that was enabled in the BitstreamReader class of the XMLUI in DSpace depends on byte range code in the Cocoon project, which uses a Java int to store byte offsets (DSpace Chokes … 2015)

Another problem related to byte range requests involves the DSpace statistics features. As part of the statistics subsystem of DSpace, file downloads of each bitstream (content file) are recorded and tallied. With byte range requests enabled, each request – even if it is only for part of the file – is counted as a request. Thus, streaming a large video file on a mobile device can result in many, many file visits, even if the particular file was viewed only once. This has not been a major issue for us, because at this time we aren't concerned with the specific number of views on each file. We use the statistics for viewing general trends, such as the total change in repository-wide usage on a month-to-month basis.

Next Steps

A first priority is integrating video playback capability into the main DSpace codebase. Video playback from DSpace is a desired feature, but the functionality that we have built has thus far been limited to Virginia Tech's customized version of DSpace. This has two unwanted consequences. First, organizations that wish to stream video from their DSpace repositories, but can't dedicate resources to developing a solution are unable to try this method. Second, each time a major upgrade of DSpace is released, Virginia Tech must spend time integrating the solution and testing it again to make sure that it works as expected, which tends to take at least one or two days of development time. Things may break, because testing isn't as thorough as it was during initial development. We are using maven WAR overlays in DSpace to lessen the problem during upgrades. With the maven WAR overlay, our custom code is stored in a separate directory path and used in place of the standard DSpace code for customization. This makes it easier to incorporate our code without breakage during upgrades; however, we may miss new features or bug fixes that have been added to the standard DSpace code.

One possible complication of contributing this code to the DSpace project is that the BitstreamReader component of this experiment is part of a specific DSpace interface, the XMLUI, aka Manakin, which is based on Cocoon, while a substantial number of organizations use an interface that is based upon the JSPUI, and other interface technologies are currently under evaluation. However, the VTechWorks group has been restructured to follow agile planning processes, and is now committed to giving back to open source communities.

Our process for encoding the videos has gone from ad-hoc, automated scripts on an unofficial file server set up by the event capture group, to a people-driven process utilizing iFFmpeg. It may be desirable for us to reach some sort of compromise between automation and control. DSpace curation tasks may be one way of accomplishing this goal. A DSpace curation task is a module of code that can be run upon request by staff on a specific item or collection to accomplish a well-defined goal. Curation tasks often make use of external software services. Examples of current curation tasks are virus scan, using Clam AV, and a metadata translator, using Microsoft Translate. A DSpace curation task for video encoding could make use of ffmpeg and reasonable defaults to allow other institutions to encode videos in the proper format. FFmpeg, which is CPU intensive, would be required on the DSpace server.

Subtitles for the deaf and hard-of-hearing are currently supported using WebVTT files, but at the present moment, only two of the videos in the institutional repository have had subtitles created. Library staff are currently working on a project to add closed captions to all library produced videos as part of a campus-wide effort.

Figure 2. Caption Initiative.

Additionally, an exploratory app for the Apple TV, known as VTechWorks Videos, or VT TV, has been released (VTechWorks Video …, 2015). The app presents a selection of videos from the VTechWorks institutional repository in a simple menu. The videos are streamed directly from VTechWorks, using the BitstreamReader portion of the project described in this paper. The videos in the repository were encoded with settings considered appropriate a few years ago for display in a web browser or in mobile devices. Seeing these videos on a large screen with 1920×1080 resolution, which has become relatively common, has spurred a desire to re-encode the archival videos into a higher resolution, higher quality setting. For the present time, DSpace will serve as the main video presentation system for academic works created at Virginia Tech, with the exception of some video exhibits put together by the special collections department that are stored in Omeka, while library marketing videos are placed on YouTube and Vimeo. The experimental VT TV app may evolve as an aggregation point to make all of these videos accessible from a single location. Preservation copies of videos are currently on a large storage system awaiting deposit into an upcoming dark archive that is likely to be based on Archivematica.

While nearly all of these videos in the VT TV app are streamed from the repository, one title is currently being streamed from Amazon S3 to the Apple TV. This was done to allow fast experimentation with playlists and content delivery networks without placing additional experimental code into the institutional repository codebase. There are a few items in the repository that have multiple videos; in the current implementation, only the first video is played back in the embedded window. Therefore it makes sense to also add playlist support to the repository implementation. In addition to playlist support, the VT TV application utilizes adaptive bitrate streaming, so that an appropriate video is selected according to user bandwidth. If the bandwidth is not available for quality playback of the 1920×1080 video, the Apple TV selects a video at a lower resolution. This feature is also implemented through m3u8 playlists, and is a likely candidate for future inclusion in the VTechWorks institutional repository video playback features.

Links

DSpace [Internet]. [Retrieved 2015 Dec 02]. DuraSpace. Available from: http://www.dspace.org

DSpace chokes on large uploads (over 2GB) [Internet]. [Retrieved 2015 Dec 02]. Mello99(GitHub username), Virginia Tech University Libraries. Available from: https://github.com/VTUL/vtechworks/issues/91

Event Capture [Internet]. [Retrieved 2015 Dec 02]. Center for Digital Research and Scholarship, Virginia Tech University Libraries. Available from: http://www.cdrs.lib.vt.edu/services/event-capture.html

Garbage scripts [Internet]. [Retrieved 2015 Dec 02]. Available from: https://github.com/keithgee/garbage_scripts/blob/master/event_capture/1process_mov_for_ir.pl

Placing “moov atom” at the beginning of an MPEG-4 video with FFMPeg [Internet]. [Updated 2011 April 28]. Cut from the North. Available from: http://cutfromthenorth.com/placing-moov-atom-at-the-beginning-of-an-mpeg-4-video-with-ffmpeg/

Video.js Player [Internet]. [Retrieved 2015 Dec 02]. Available from: http://videojs.com

Video Playback in VTechWorks [Internet]. [Updated 2012 Sep 07]. Digital Library and Archives, Virginia Tech University Libraries. Available from: https://blogs.lt.vt.edu/dlablog/2012/09/07/video-playback-in-vtechworks/

VTechWorks – Closed Caption Support [Internet]. [Updated 2012 Nov 07]. Digital Library and Archives, Virginia Tech University Libraries. Available from: https://blogs.lt.vt.edu/dlablog/2012/11/07/vtechworks-closed-caption-support/

VTechWorks Video: an Apple TV App for our institutional repository [Internet]. [Updated 2015 Nov 02]. Marooned Librarian. Available from: https://maroonedlibrarian.wordpress.com/2015/11/02/vtechworks-video-an-apple-tv-app-for-our-institutional-repository/

WebVTT [Internet]. [Retrieved 2015 Dec 02]. Wikipedia. Available from: https://en.wikipedia.org/wiki/WebVTT

About the Authors

Keith Gilbertson (keith.gilbertson@vt.edu) is a Digital Technologies Development Librarian at Virginia Tech.
Liz McVoy (lizmcvoy@vt.edu) is the Digital Media Specialist on the Creative Services Team at Virginia Tech’s University Libraries. A combination of education, internships, and previous work endeavors shaped her love of video production, specifically video editing and graphic design. She uses her videography and design skills to promote the Libraries’ spaces, services, and resources. a Digital Media Specialist in the Public Relations and Marketing Department of the University Libraries at Virginia Tech.