Tech Review

CS461 - Senior Software Engr Project

Instructor: D. Kevin McGrath

Instructor: Kirsten Winters

Fall 2018

Matthew Jansen

November 19th, 2018

Abstract

This document will discuss the different technologies used within our project, specifically how our project will export pictures and videos from within the web application. These technologies are a necessary component to the functionality and usability of our web application, as the project is supposed to allow the user to obtain snapshots or movies of their results and use them for presentation. Our project entails converting data stored within a CSV file to a 3D environment with timeline capabilities. To explain how the web application will give the exporting feature to its users, we will first discuss the different formats of files that can be exported, such as PNG and GIF files for pictures and MOV and MP4 for movies. Finally, we will show how various Javascript frameworks can be implemented to export the files from the application to the user.

I. User Presentation of Export Module

Before discussing how we will be exporting these pictures, a brief overview of the prerequisites for this functionality should be considered. Moving forward, when talking about the logistics of the webpage that contains the export module, we will make the following assumptions.

First, the user is able to view their 3D environment, which contains all of the data points and general plot information related to the CSV file that they provided. Second, the user is able to view an export module which allows them to select one or more snapshots of their 3D environment. Third, an export button is available to the user, which they can click after selecting one or more snapshots of their 3D environment. After clicking this button, the user can expect that the web application will convert the selected snapshots into the picture or movie file that they choose.

For pictures, the user can choose between a PNG file type for a single snapshot of the 3D environment, or a GIF file type to create an animation using several snapshots. As for movie files, after the user selects more than one snapshot of their 3D environment, the user can choose between downloading a movie file as a MP4 file type, or as a MOV file type. After these selections are made and the user clicks the export button, the user can expect the web application to download the respective file using their web browser.

II. Exporting Pictures

With the ability to export pictures, the user is able to contain a summary of their desired data points in a 3D environment, such that a customer presentation can be done without the need to be online.

A. Picture file types

In this web application, the module to export images of the 3D environment will support PNG and GIF file types. PNG file types are universally used, along with GIF file types, for single-image formats (PNG), as well as animations that span over several images (GIF).

B. html2canvas.js

html2canvas[1] is a javascript module which allows a user to screenshot a specific frame of a web browser. This occurs by the scripts rendering a screenshot by reading the DOM and the different styles applied to the elements. To do this, the *html2canvas()* method is called with the DOM element (to be converted to a HTML canvas element) as a parameter. After the canvas element has been converted from

the DOM, the next step will be downloading the canvas element as a PNG file. This can be done by including a *download()* method in the Javascript file we previously used, which does not involve using any external frameworks[2]. Together, these methods will provide the webpage the functionality to download PNG images of the users 3D environment in a single-frame format.

C. gif.js

gif.js[3] is a javascript framework which encodes GIFs, and uses typed arrays and web workers to render the frames within the animation. This functionality gives gif.js the ability to perform GIF encoding very quickly, all while in the background of the web browser. This framework will come into play when the user wishes to save one or more snapshots into the form of a GIF animation, rather than a movie file. As we saw with html2canvas, we can create a snapshot of the current 3D environment using the *html2canvas()* method within our web application. After saving these snapshot(s) that the user would like to convert into a GIF, we can pass them into the gif.js framework using the *gif.addFrame()* method, which can take in parameters of either image elements or canvas elements. Once we finish adding frames into the GIF, we can use the *gif.render()* method to render the gif, and then download it to the users web browser. Doing this gives the user the ability to save animations, alongside image and movie files, giving the user a wider array of options to choose from.

III. Exporting Movie Files

Having the ability to export movie files allows the user to explore more options when deciding how they would like to present their 3D environment to a group of people.

A. Movie file types

In this web application, the module to export videos of the 3D environment will support MOV and MP4 file types. These two movie file types are used universally for playing videos, so if a user is able to play videos on their device, then chances are that they will be able to export videos from this web application.

B. whammy.js

In order for the web application to export MOV and MP4 file types, we must first convert the snapshots that we obtain using the *html2canvas()* methods into a movie format. To do this, we will use the

whammy.js[4] framework to convert these canvas elements into a webm format.

This can be done by first defining a new whammy video using the *whammy.video(x)* method, where x is the number of frames we wish to add into the video. Next, we can begin adding canvas elements into the whammy video using the *whammy.add(canvasElement)* method, where *canvasElement* is the canvas element that we obtained through html2canvas. Once we are finished adding snapshots of the 3D environment into the video, we can compile the frames using the *whammy.compile(false, function(output))* method. This will create a webm file type which contains the compiled snapshots that we wish to turn into a video. The next steps include converting this webm type file into a MOV or MP4 file type, which we will do using *ffmpegserver.js*.

C. ffmpegserver.js

This javascript framework, *ffmpegserver.js[5]*, is a simple node server that begins a capture, collects canvas frames, and converts them to a MP4 format. To use it, a *CCapture* object needs to first be created using

```
var capturer = new CCapture({
    format: "ffmpegserver" });
```

Depending on the users choice of video format, we can choose to do a more in-depth definition of the capturer object and choose to capture the frames as a MOV file type, however the default file type when defined as stated above is a MP4 file. After the object is defined, we can begin capturing using the *capturer.start()* method. Next, we define a *render()* function containing the *capturer.capture(canvasElement)* method, which will collect the canvas elements (collected using html2canvas) into the capturer object. Lastly, we can call *capturer.stop()* to cease the collection of canvas elements, and then use the *capturer.save()* method to save the captured frames as the movie file type which was specified earlier by the user. After this is done, the web application can begin downloading the movie file using the users browser.

IV. Conclusion

In conclusion, through the use of several Javascript frameworks, we are able to add the functionality of converting and exporting snapshots of the 3D environment from the web application to the client. Furthermore, we are able to do so in a multitude of file formats, ranging from PNGs and GIFs to MOV and MP4 movie file types. This gives the customer a wide range of options to choose from, and can be very beneficial when deciding the best method to present the data.

REFERENCES

[1] N. von Hertzen, “html2canvas,” 2018, (Accessed on 11/18/2018. [Online]. Available: https://github.com/niklasvh/html2canvas

[2] user1693593, “Stack overflow,” 2018, (Accessed on 11/21/2018. [Online]. Available: https://stackoverflow.com/questions/18480474/how-to-save-an-image-from-canvas

[3] J. Nordberg, “gif.js,” 2018, (Accessed on 11/18/2018. [Online]. Available: https://github.com/jnordberg/gif.js

[4] K. Kwok, “whammy.js,” 2018, (Accessed on 11/19/2018. [Online]. Available: https://github.com/antimatter15/whammy

[5] Greggman, “ffmpegserver.js,” 2018, (Accessed on 11/19/2018. [Online]. Available: https://github.com/greggman/ffmpegserver.js