My Final College Paper

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Dylan H. Huff

May 2019

Approved for the Division
(Computer Science)

Eric S. Roberts

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class.

List of Abbreviations

You can always change the way your abbreviations are formatted. Play around with it yourself, use tables, or come to CUS if you'd like to change the way it looks. You can also completely remove this chapter if you have no need for a list of abbreviations. Here is an example of what this could look like:

CS	Computer Science
JS	JavaScript

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Dedication

You can have a dedication here if you wish.

Introduction

Chapter 1

Algorithm Animation

This chapter is an overview of algorithm animation and where this thesis fits into that story.

1.1 What is Algorithm Animation?

Algorithm animation is the process of taking algorithms and giving them graphical representations.

1.2 Motivation for Algorithm Animation

Algorithm animation has benefits for teachers and students. Below are some of the most commonly cited benefits.[1]

- It allows teachers to display algorithms in lectures easily.
- Another method for teaching students fundamental algorithms, pictures and code compared to just code.
- It allows for another avenue of debugging.

1.3 Use in Computer Science Education

1.3.1 History

Algorithm animation began in the 70's and has seen usage since. In the early days of algorithm animation, teachers used tools to make animations for their presentations. Often time these were predefined short films. As tools progressed, the animations no longer had to be used exclusively by teachers and didn't need to be predefined.[2] Tools became able to dynamically represent the algorithms that students made.

[1]Hundhausen et al. (2002)
[2]Hundhausen et al. (2002)

One of the most famous and important contributions to algorithm animation was BALSA. BALSA was created in 1987 by Marc Brown. BALSA introduced several major innovations in the field. One major contribution was the addition of real time animations. Prior to BALSA, the animations wouldn't operate in real time, rather the animation would algorithm would run then the animation would be made, unlike real time animation which executes the animation and algorithm simultaneously. Another innovation was the introduction of scripts. Scripts were predefined PASCAL programs that would control the algorithm and could be executed in real time. This allows teachers to predefine how an animation will execute, then present the animation in real time.[3]

Along with this shift form predefined to dynamically defined tools, other features were added that improved the animations. One paritcularly notable improvment came in when animations transitioned from 2D to 3D. This shift from 2D to 3D allowed for more information to be simultaneously displayed. Along with this improvement, color and sound were both added to animations.[4]

1.3.2 Current State of Algorithm Animation

There are a few challenges facing modern day algorithm animation. One of these challenges is the lack of adoption of animation tools by instructors.Levy points out two main challenges of adoption from survey results of teachers. The first is that the tools being developed may be feature rich but not integrated well into the existing material or curriculum. This illuminates the fact that the tool developers aren't usually primarily concerned with integration, but rather features. Second, they cite "centrality" as the other major inhibition of teachers. They define centrality to be where the center of learning is for the students. By making animation tools that animate the students algorithms, the centrality is being moved from the teacher to the student. They note that this phenomenon is present with highly confident and experienced teachers through not confidant inexperienced teachers.[5] Its also worth noting that the teachers in this study are high school teachers and not college professors.

Along with low adoption rates by teachers, there is also a lack of work being published.[6]

[3]Brown (1987)
[4]Najork & Brown (1994)
[5]Levy & Ben-Ari (2007)
[6]Kucera (2018)

Chapter 2

State of Computer Science Higher Education

This chapter will be about the state of CS higher ed, its problems, some solutions and how JavaPPTX fits into that picture

2.1 Background

Talk about the history and formation. Talk about its periods of growth and bust. Talk about current growth and future growth.

2.2 Problems facing Higher Education

Talk about various problems, how they arose and what their effects on education are.

2.2.1 Lack of PhDs

lack of number of people and huge demand for PhDs from market

2.2.2 Increasing Demand from Students

2.3 Proposed Solutions

In each subsection, I will talk about the pros, cons and feasibility of every solution presented

2.3.1 Retraining PhDs

taking teachers with PhDs not in CS and retraining them to teach intro level classes.

2.3.2 Teaching Only Staff

Staff that teach but don't hold the professor title

2.3.3 Teaching Tools

talk about some of the existing tools to aid profs.

2.4 JavaPPTX

Overview of the package, how it helps address some of the problems in CS higher ed, and I'll end with talking about areas of expansion, which will be a good segway into my project. Also mention that this tool, and no tool, will be the solution to solving the problems facing higher ed.

2.4.1 Background

Here I will briefly talk about the features in the package. Some features include saving time animating and more effective at conveying concepts to students using pptx.

2.4.2 Classroom Usage

I'll talk about how it fits in unobtrusively and its current usage. May not be enough here to warrant a subsection.

2.4.3 Package Expansion

Here I'll talk about ways that the package could be expanded and the benefits of the specific expansions. One will be my expansion.

Chapter 3

JavaPPTX to JavaScript

3.1 Statement of Work

3.1.1 New Features

The main addition of this work is he ability to export the animations to HTML and JS without having to add any additional logic. This means that a professor could use JavaPPTX to create a PowerPoint for lecture, and with only having to add the following two lines

```
PPSaveJS testSave = new PPSaveJS(ppt);

testSave.save(”../example.js”);
```

the lecturer could create a web page that is identical to the PowerPoint. The newly created HTML file and JS file could be uploaded as is to create a web page, or could easily be embedded within an existing web page.

3.1.2 Uses of this Expansion

Having a native web output has two main benefits, the ease of viewing the animation and allowing new audiences to view the animation. PowerPoint is not web native, meaning that someone that wants to view the lecture needs to download a copy of the PowerPoint and have access to an application that can view the PPTX file. While this isn’t too large of a burden on laptops or desktops, accessing the lecture on a phone is very cumbersome and often infeasible. This barrier is easily overcome with HTML and JS as an output as almost all phones, as well as laptops and desktops, can view a web page easily. This is very convenient for anyone trying to view the lecture, but it also allows people without access to a laptop or desktop to view the lecture.

3.2 Internal Logic

The process of building this feature set and making individual algorithms can be broken down into two parts, building the framework and cross compilation of the logic.

3.2.1 JavaScript Framework

The JS framework relies heavily on the Canvas API. This API is meant to make animating in JS and HTML much simpler. The framework that I built is built on top of that and offers more abstractions to the programmer while also storing objects to be animated in a hierarchical and object oriented manner. This makes animating shapes and text objects much simpler. As an example consider drawing 5 circles and having them move using Canvas vs this framework. Canvas doesn't provide any way to keep track of drawn objects. It doesn't provide a native way to make objects move. It doesn't provide native re-rendering of drawn objects which is a problem when trying to move an object since it forces a rerender. Using the framework, a programmer would only need to create the 5 circle objects by passing them a few starting attributes (position, color, etc) then call a move function on the desired circle.

The framework is created dynamically, only building the parts that are necessary for that particular algorithm. This allows the resulting JS to be as minimal and fast as possible. This is done at compilation every time. There is some boilerplate framework that is created every time.

3.2.2 Cross Compilation

After the framework is built, the logic is processed. The native JavaPPTX package stores all of the information passed by the programmer then creates the corresponding PowerPoint. Since all of the information is stored by the package prior to being complied to a pptx file, the information can be stored in the same way but compiled to JS. After a save to JS function is called, the program will crawl through all of the stored data, and run the corresponding compilation to JS functions. This step was particularly tricky because the way that Java stores information is different from JS, so all the logic needed to be translated. Take storing color data as an example. Java may store the color as ”RED” but JS would take a hex string as a color attribute. This logic translation was possibly the most difficult part of creating this extension. It required understanding how Java and JavaPPTX stored information, then figuring out the best way to store that in JS.

3.3 Future Work

3.3.1 New Features

What features did I miss? How can this overall be improved? This section will be quick to write once I know if/what features I was unable to get to.

3.3.2 Realtime JS Animation

As this work stands, the animations that can be made are not real time and aren't reflecting algorithms that a student makes. A meta study has shown that it maybe more effective to have the animations reflect the students algorithms rather than having a teacher animate an algorithm and use it in a lecture, but this is contentious.[1] A major contribution of this thesis is a JavaScript framework that allows for animations to be created more easily. In its current state, the animations are built based off of the logic given by the person creating the animation, which is intended to be an instructor. This framework could be used to reflect animations in real time that students create. Since the visual framework has been built, someone expanding on this work would have to figure out the logic behind creating animating the algorithms. The easiest expansion could be animating JS algorithms. Since JS is native to the web, it also maybe worthwhile to have other languages have animations done with this framework since the animations could easily be made into websites and shared. Since this package is written in Java, Java would be a logical language to expand this to.

[1]Hundhausen et al. (2002)

Conclusion

Here's a conclusion

Appendix A

The First Appendix

References

Brown, M. H. (1987). *Algorithm Animation*. Phd thesis, Brown University.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *13*(3), 259–290. http://www.sciencedirect.com/science/article/pii/S1045926X02902375

Kucera, L. (2018). Visualization of abstract algorithmic ideas. (p. 8).

Levy, R. B.-B., & Ben-Ari, M. (2007). We work so hard and they don't use it: Acceptance of software tools by teachers. (pp. 246–250). Event-place: Dundee, Scotland. http://doi.acm.org/10.1145/1268784.1268856

Najork, M. A., & Brown, M. H. (1994). A library for visualizing combinatorial structures. (pp. 164–171). Event-place: Washinton, D.C. http://dl.acm.org/citation.cfm?id=951087.951119