PowerPoint Natively on the Web, a JavaPPTX Extension

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 begin by thanking Eric, rather obviously for his help on thesis, but more importantly, for the conversations and guidance outside of thesis. The readings for classes, and the books suggested outside of class, have changed what I want to do with my career and I am very grateful for that.

To Indra and Hannah. While there is more to be said then what I can place on this page, I want to thank you both for keeping my spirits high when thesis felt overwhelming. Your laughter and friendship helped me get through thesis, and Reed.

List of Abbreviations

API	Application Programming Interface
CS	Computer Science
HTML	Hypertext Markup Language
JS	JavaScript

Table of Contents

List of Figures

Abstract

Computer Science higher education is facing a growing crisis, there aren't enough faculty members to support student demand. This problem shows no sign of getting any better in the years to come as the number of people getting Ph.D.s annually who intend on working as a professor is far fewer than the number of unfilled faculty positions. Various tools have been created for Computer Science professors that free up time so that they can focus on students. There are tools for automating grading, animation creation and presentation creation. This thesis focuses on one package, JavaPPTX. This package allows people to make complicated PowerPoint animations in Java. Packages to easily make complicated animations are important because teaching topics such as recursion, stacks, and sorting algorithms can be graphically demanding. Making these presentations in JavaPPTX allows the user to have precision that would be very difficult to get using Microsoft PowerPoint. My contribution to this package is the addition of compilation of the Java code to JavaScript and HTML. The addition of one line of code on the users part makes a web native version of the presentation. This additional functionality allows presenters to make websites that are identical to their presentations with marginal additional effort. From the student's perspective, having a web native output gives them the option to view the presentation on any internet connected device without needing any additional software. In an age where cellphones are increasingly the way people access the internet, my extension adds flexibilty and convenience for everyone and makes the presentation to accessible to audiences who wouldn't otherwise be able to access it.

Dedication

I'd like to dedicate this to my parents, who have sacrificed so much over their entire lives to allow me to get this education. I love you both and will always be grateful for this gift.

Introduction

In order to support the already strained Computer Science professors, I created an extension to the JavaPPTX package that allows presentations to be web native. This presentations would be programmed in Java, but the output is JavaScript and HTML. My extension to package can be accessed at

https://github.com/dylanhuff/thesis-code

The most up-to-date version of JavaPPTX and any further expansion or extensions can be accessed at

https://github.com/eric-roberts/JavaPPTX

The rest of this thesis will be broken down as follows. Chapter 1, I define algorithm animation and explain its usage in Computer Science education. In Chapter 2, I describe the state of computer science higher education in 2019 and where it is forecast to go in the coming years. This chapter also includes some proposed solutions to combat the current faculty shortage. In Chapter 3, I explain the functionality I added to JavaPPTX and how it works. This chapter also includes some of the technical challenges I faced along with way and how my extension could be expanded.

Chapter 1

Algorithm Animation

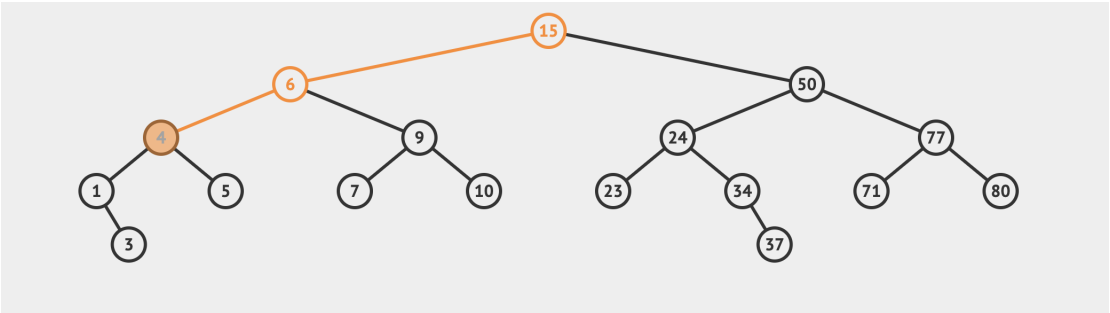
1.1 What is Algorithm Animation and What are its Benefits?

Algorithm animation is the process of taking algorithms and giving them graphical representations. 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.
- It provides another method for teaching students fundamental algorithms, it adds a more visual element rather than just showing code.
- It allows for another avenue of debugging.

Take Figure 1.1 as an example. The algorithm animation is showing what happens to an AVL tree when the number two is inserted into it. The animation currently shows the path the program takes to find the appropriate location to insert two. Figure 1.2

Figure 1.1: Example of an Algorithm Animation

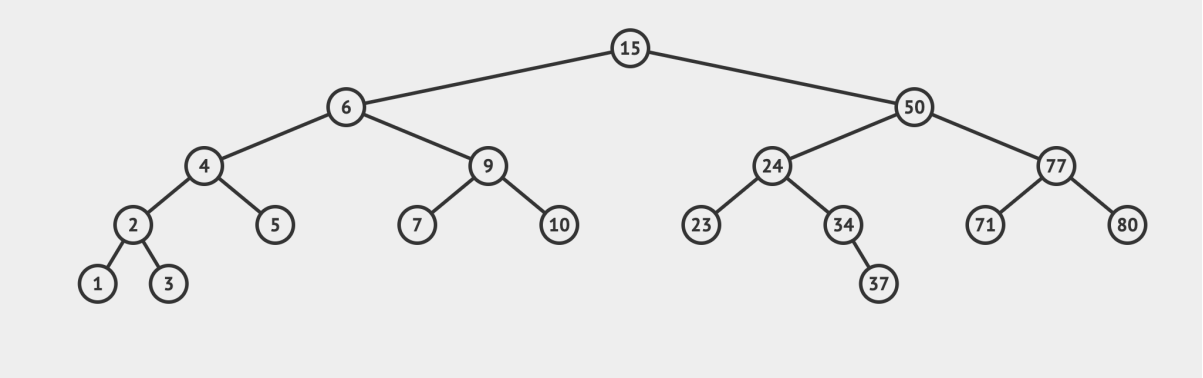


Source: https://visualgo.net/en/bst

shows what the AVL tree looks like after the animation is completed and two has been added.

[1] Hundhausen et al. (2002)

Figure 1.2: Example of an Algorithm Animation Completed



Source: https://visualgo.net/en/bst

1.2 Use in Computer Science Education

1.2.1 History

Algorithm animation began in the 1970s and has been increasingly used since. In the early days of algorithm animation, teachers used tools to make animations for their presentations. Often, 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 have progressed to the point where they are able to dynamically represent the algorithms that students make.

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. The algorithm would run, and then theapplication would make the animation. Real time animation on the other hand 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 approach allows teachers to predefine how an animation will execute, then present the animation in real time.[3]

Along with this shift from predefined to dynamically defined tools, other features were added that improved the animations. One particularly 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. The later addition of color and sound allowed animations even more avenues to convey information.[4]

[2]Hundhausen et al. (2002)
[3]Brown (1987)
[4]Najork & Brown (1994)

1.2.2 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 incongruity 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 student's algorithms, the centrality is being moved from the teacher to the student. The teachers polled were uncomforable with the centrality shifting from them to the students. They note that this phenomenon is present with highly confident and experienced teachers through not-confident, inexperienced teachers.[5] It is 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 new work being published, as of 2018.[6]

[5]Levy & Ben-Ari (2007)
[6]Kucera (2018)

Chapter 2

State of Computer Science Higher Education

This chapter describes the state of CS Higher Education, its problems, some solutions and how JavaPPTX fits into that picture.

2.1 Background

While there is a lot to be said about the entire history of Computer Science higher education, this section will focus on previous booms and bust of enrollment in Colleges.

In the following sections, I outline the various problems facing CS Higher Education today. These are not the first time these problems have been faced though. Rather, this current crisis could be seen as the third boom that has happened.[1] In the past two booms, there was subsequent decrease in the number of students gradutating with CS. There is no one factor that can account for the historical decreases in the past.[2] Figure 2.1 shows one boom and bust that occurred in the 1980s. Without an intervention, the current massive growth in enrollment will likely be followed by another period of sharp decrease in enrollment.

2.2 Problems facing Higher Education

2.2.1 Lack of PhDs

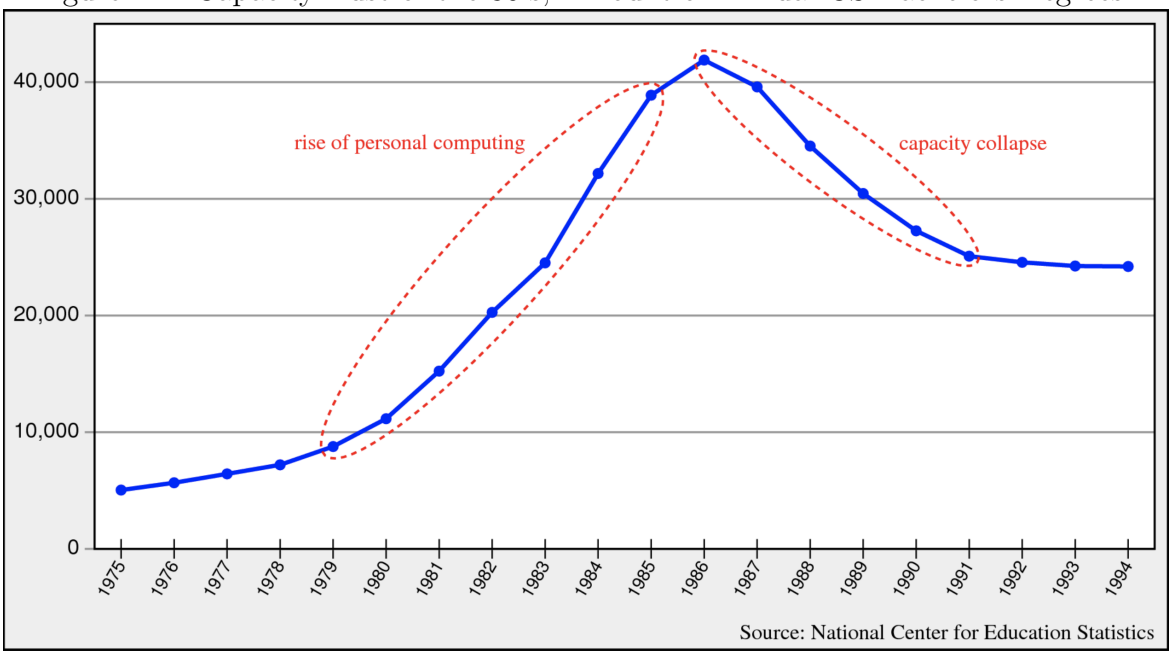
One of the major issues facing CS higher education is the amount of PhDs being produced and the percentage of them going to industry versus professor positions. Currently, 57% of the new PhD graduates go to industry.[3] With wages in industry being considerably higher than wages for CS professors, there is no sign of this trend decreasing.

[1] Roberts (2016)

[2] Committee on the Growth of Computer Science Undergraduate Enrollments et al. (2018)

[3] Zweben & Bizot (2018)

Figure 2.1: Capacity Bust of the 80's, Amount of Annual CS Bachelors Degrees



Currently about 30% of PhD graduates go into academia, but that doesn't mean they all take tenure track positions. About 10% of all graduates choose to go into Post Doctoral positions while another 2% go into research positions.[4] In the end, only 18% end up in teaching positions. This translates to 320 graduates per year. To put that in perspective, there are currently 1577 institutions that offer a Bachelor's in CS, meaning, on average, a program could hire a new faculty member every 5 years. While those numbers seem bleak, the picture only gets worse. Consider that when large prestigious universities have openings, the graduates are much more likely to go there compared to smaller local universities with less prestige.

2.2.2 Increasing Demand from Students

Between 2009 and 2015, the number of people graduating with a bachelor's degree from not-for-profit institutions has increased 74% - a rate significantly higher than the general rate of increase in bachelors degrees awarded over the same period of time. This rate is the average increase. There is large variation between institutions, with some seeing rates much higher than the average. In particular, research institutions tend to see higher than average rates of increase. Another consideration with this average is that some institutions also work to cap the amount of prospective CS students they admit.

On top of increased degree production, enrollment of CS courses from CS majors and non-majors has also increased since 2005. This trend also shows no sign of slowing down within the next few years, without any institutional discouragement.

[4] Zweben & Bizot (2018)

One of the major reasons for this increased demand is that in the U.S., the number of jobs related to computing have been increasing steadily for the past 40 years. Currently, there are not enough students graduating with degrees relating to the field to fill open the current open job positions. Looking down the road, the U.S. Bureau of Labor Statistics estimates that the number of jobs in this area is estimated to continue to grow rapidly through at least 2026.[5] Beyond the data predicting continued growth, given the increasing pervasiveness of computers in the world, it is clear that the amount of jobs will continue to grow for the foreseeable future.

2.2.3 Budget Constraints

Getting an appropriate budget is often a challenge for schools, but CS higher education today has some unique problems. As I mentioned earlier, wages in industry are particularly high for CS related Ph.Ds. Compounding this problem is that many universities have internal rules that enforce equal pay for professors regardless of their area of study. This internal wage equality means that the pay for CS faculty cannot simply be increased to lure people away from industry. In many institutions, it would require raising the pay for faculty across the board, which is in-feasible.

The other major budgeting issue is simply the under-funding of computer science departments, especially at state funded schools. In recent years, many states have decreased there budget for higher education.[6] This decrease in spending comes during a boom in enrollment in CS, meaning that while departments need to expand their budget to hire new faculty, they are unable to. The under-funding of departments in response to budget constraints is particularly short sighted because these schools train a large fraction of the CS related work force, which has benefits for the state and federal economy.[7]

2.3 Proposed Solutions

Currently there is no definitive solution to the problems facing CS higher education. Most of the problems stem from the job market, and there is no sign that the job market will ease up on the pressure that it is applying. There are a few solutions that have been proposed, none of which will be the sole answer.

Stanford recently implemented a program to retrain Ph.D.s from others disciplines.[8] The program is a Masters of Education which will teach these people fundamental CS skills. Key to the faculty shortage is the requirement that most universities and colleges have that professors must have a Ph.D. The goal of this program is not to have these people teaching advanced CS courses, but rather have them be the teachers for some of the introductory courses. These retrained faculty also do not need to participate in outside research; rather their main focus would be teaching courses. This

[5] BLS (2018)
[6] Committee on the Growth of Computer Science Undergraduate Enrollments et al. (2018)
[7] Committee on the Growth of Computer Science Undergraduate Enrollments et al. (2018)
[8] Starkman & University (2016)

solution should help free up the faculty that has more advanced knowledge, allowing them to teach the high level courses and do their research. As an added benefit, it maybe the case that since the people being retrained are committed academics, they could be less likely to leave for industry.

Another partial solution is the creation of supporting tools for existing faculty. In the next couple of years, there will not be enough professors to meet student demands. Even if a solution is found to get more professors, there will be years of delays before that solution is felt since it takes years to train people. Although it is clear that more tools cannot replace faculty, in terms of immediacy, tools have the advantage. There are existing auto grading tools for programming assignments which have been helpful. Another aspect of teaching that could be added is presenting. Tools could aid in the time it takes to create a presentation and the effectiveness that said presentation has. One tool in particular, JavaPPTX, is the focus of this thesis.

2.4 JavaPPTX

JavaPPTX, created by Eric Roberts, is a library for Java that aids in the creation of PowerPoint Presentations. The package allows someone to write a program in Java that will create a PPTX file. This differs from the usual method of creating PowerPoints using Microsoft PowerPoint, which can become very cumbersome when making detailed animations that have specific movements, require many moving objects, or require specific timing of animations.

2.4.1 Background

The package supports many of the features in PowerPoint and adds functionality in other cases. Here are some of the features supported:

- Creation of many shapes, including rectangles, ovals, lines, and more.
- Creation of text objects, including titles and long form text.
- Animation of objects. This includes fades, appearing, disappearing, linear motion, and more. All of these can be event based (clicks or delays).
- Support for multiple slides and slide transitions.
- Importing of images.

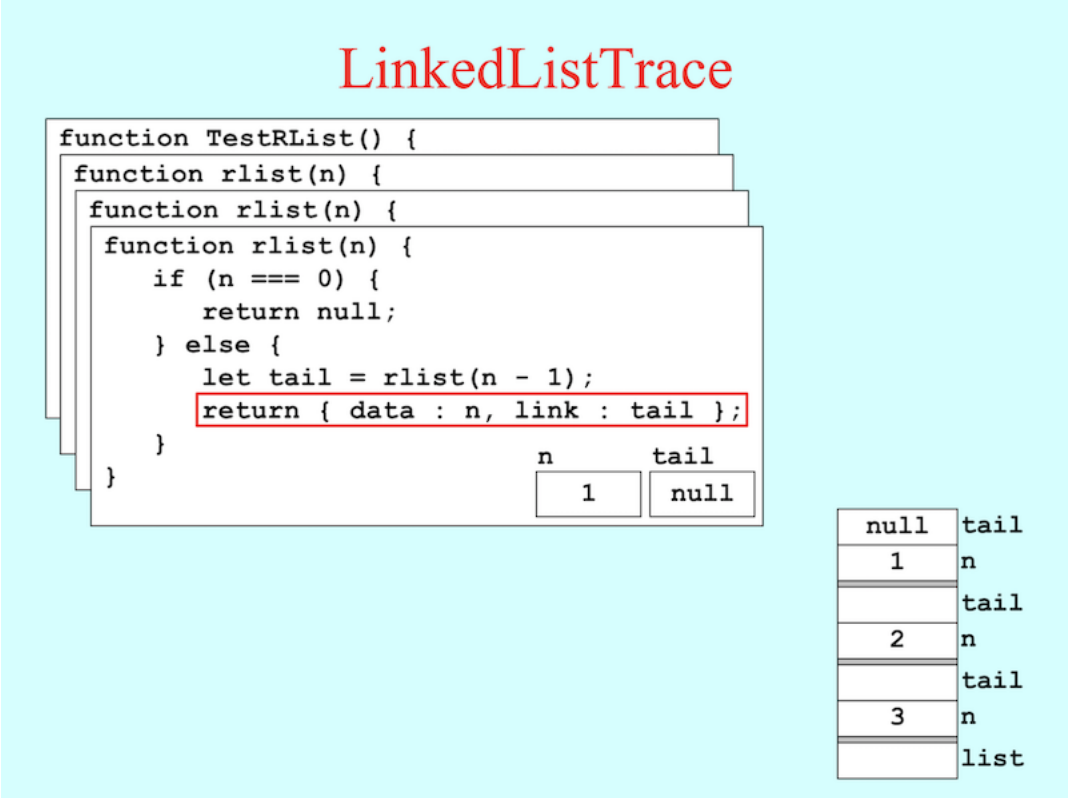
Some of the extended functionality includes:

- An animated terminal function.
- Exact animations including bezier paths.
- An animated stack tracing function.

2.4.2 Classroom Usage

One of the major benefits of the package is that it unobtrusively saves professors' time while allowing them to animate hard to teach concepts. The package fits in well because it doesn't require professors to change their teaching methods, as many professors already use PowerPoint during their lectures. While there is a learning curve associated with the package, it isn't particularly difficult to use and would save time in the long run. The package can be used to make simple slides, but the package was designed to, and excels at, making complicated PowerPoints. Some topics that can be more easily explained are recursion, loops, and debugging.

Figure 2.2: Example of an Algorithm Animation in JavaPPTX



Chapter 3

JavaPPTX to JavaScript

3.1 Statement of Work

3.1.1 New Features

The main addition of this work is the ability to export the presentation to HTML and JS without having to add any additional logic. This added functionality means that a professor could use JavaPPTX to create a PowerPoint for lecture, and with only adding one line 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.

Take the code in Figure 3.1 an example, which makes the previously shown Linked List Animation. The only line that needs to be added to export to JS and HTML is the line shown boxed, line ten, in Figure 3.1.

Another major addition was the automatic scaling to screen resolution. When a PowerPoint is created with this package, the slides will be a fixed size, defined in pixels. A major benefit of using JavaScript is the ability to scale the presentation without any additional work on the presenter's part. This is an important feature because one benefit of outputting to web is that phones can natively view the presentation. With auto-scaling built in, the presentation will adjust to phones and desktops.

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 without any additional software. 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.

Figure 3.1: Java Code for Linked List Animation

```
1  import edu.stanford.cs.pptx.*;
2
3  public class PPLinkedListTrace {
4
5     public void run() {
6        PPShow ppt = new PPShow();
7        PPCodeTraceSlide slide = new HSSlide();
8        ppt.add(slide);
9        ppt.save("LinkedListTrace.pptx");
10       PPSaveJS.save(ppt,"LinkedListTrace.js");
11       System.out.println("LinkedListTrace.pptx");
12    }
13
14    public static void main(String[] args) {
15       new PPLinkedListTrace().run();
16    }
17    ...
18 }
```

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 to the JavaScript framework.

3.2.1 JavaScript Framework

The JS framework relies heavily on the Canvas API. The Canvas API is meant to make animating in JS and HTML much simpler. The framework that I built is constructed 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 five circles and having them move using Canvas vs my 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 my framework, a programmer would only need to create the five circle objects by passing them a few starting attributes (position, color, etc) then call a move function on the desired circle.

My framework creates classes for many of the shapes supported by JavaPPTX. The way that the framework was built allows for shapes classes to be easily added. The framework was created in a hierarchical manner, putting many of the methods into the abstract classes. This way, any objects added will have many of the methods

already created for it.

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 and 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. Futhermore, the information can be stored in Java in a variety of ways.

Translating from Java to JS was one set of problems, but another set of problems was translating between PowerPoints internal logic to JavaScripts. Most of the time, logic was stored in a manner that was clearly related to the object. For example, a rectangle would contain the length, width, color, position on the screen, etc. There were times that the information for an object was stored not based on what made sense for that object vacuously, but the information was stored with regards to the way PowerPoint needed to be given the information. One particularly troublesome example was lines. PowerPoint draws lines by keeping track of what quadrant of the slide the line is in. JS however doesn't render lines according to quadrant, so the logic needed to be translated before it is output.

3.3 Future Work

3.3.1 Full JavaPPTX Support

The JavaScript compilation is not fully completed, not all of the features in JavaPPTX are supported in the JavaScript compilation. The main structure for the JS compilation is complete though. As an example of what that means, I'll explain how the program process a presentation with one slide that has a circle and a star on it. The program starts by storing all of boilerplate information, number of slides, height and width of slides, etc. The program then goes and tries to process all of the objects on a given slide. Since all of the objects are not yet supported, some objects will be skipped. In this step, because circles are supported, a circle would be processed and added to the presentation. A star object is not supported, and so it would be skipped over during this phase. All of the information would then be exported to JavaScript and rely on the JavaScript framework that I built to display the objects on the screen. The main infrastructure of processing, storing and building objects in JavaScript is

done. Multiple slides can be added. Various types of motion are available for the supported for objects.

In order to finish the JS compilation, some objects, properties and features need to be added. If a star object was to be added, the logic for processing stars specifically would be added during the object processing phase. For adding motions and background colors, the JavaScript framework needs to be extended. Figure 3.2 shows all of the classes supported by JavaPPTX and my JS extension. Figure 3.3 shows all

Figure 3.2: Supported Classes in JavaPPTX and JS Extension

Class Name	JavaPPTX Support	JS Extension Support
PPRect	Yes	Yes
PPLine	Yes	Yes
PPOval	Yes	Yes
PPTextBox	Yes	Yes
PPPie	Yes	No
PPStar	Yes	No
PPGroup	Yes	No
PPPath	Yes	No
PPPointer	Yes	No
PPRectCallout	Yes	No
PPOvalCallout	Yes	No

of the animations supported by JavaPPTX and my JS extension. For the currently supported animations, there are three triggers supported for all of them in JavaPPTX and the JS extension. The triggers supported are "On Click", "After Previous", and "With Delay".

Figure 3.3: Supported Animations in JavaPPTX and JS Extension

Animation Name	JavaPPTX Support	JS Extension Support
Appear/Disappear	Yes	Yes
Linear Motion	Yes	Yes
Bezier Motion	Yes	Yes
FadeIn/FadeOut	Yes	No
FlyIn/FlyOut	Yes	No
WipeIn/WipeOut	Yes	No
ZoomIn/ZoomOut	Yes	No
FadedZoomIn/FadedZoomOut	Yes	No
CheckerboardIn/CheckerboardOut	Yes	No
Spin	Yes	No
Grow/Shrink	Yes	No
ChangeFillColor	Yes	No
ChangeLineColor	Yes	No

Take making a insertion sort presentation as an example. Figure 3.4 shows the code required to make a presentation that sorts the array [5,4,3,2,1] to [1,2,3,4,5] in PowerPoint and JavaScript. Figure 3.5 shows the webpage that is made from the

Figure 3.4: Current Features Code Example

```
import edu.stanford.cs.pptx.*;

public class PPInsertionSort {

   public void run() {
      PPShow ppt = new PPShow();
      PPSlide slide = new PPSlide();
      sortTest(slide);
      ppt.add(slide);
      ppt.save("InsertionSort.pptx");
      PPSaveJS.save(ppt,"InsertionSort.js");
   }

   public static void main(String[] args) {
      new PPSelectionSort().run();
   }

   public void PPSlide sortTest(PPSlide slide){
      slide.addTitle("Insertion Sorting");
      double xc = slide.getWidth() / 2;
      PPTextBox text5 = new PPTextBox("5");
      slide.add(text5, xc-60, 150);
      PPTextBox text4 = new PPTextBox("4");
      slide.add(text4, xc-30, 150);
      PPTextBox text3 = new PPTextBox("3");
      slide.add(text3, xc, 150);
      PPTextBox text2 = new PPTextBox("2");
      slide.add(text2, xc+30, 150);
      PPTextBox text1 = new PPTextBox("1");
      slide.add(text1, xc+60, 150);
      text1.move(-120,0,"/onClick");
      text5.move(30,0,"/withPrev");
      text4.move(30,0,"/withPrev");
      text3.move(30,0,"/withPrev");
      text2.move(30,0,"/withPrev");
      ...
     }

}
```

code in Figure 3.4 before anyone clicks on the page. After someone clicks on the

Figure 3.5: JavaScript Insertion Sort Example, Before Click

webpage, as selection sort would dictate, one gets moved to the first position and all of the other numbers are shifted over by one position. Figure 3.6 shows the same webpage after a click. This example shows how the general infrastructure is made for slides, presentations, motions, and objects. This example should also illustrate that currently only basic features of JavaPPTX are supported.

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)

Figure 3.6: JavaScript Insertion Sort Example, After Click

Conclusion

JavaPPTX is a helpful package for Computer Science professors, saving them time and making their presentations more effective. The extension that I built should seamlessly integrate into their work. Once integrated, the compilation to JS and HTML allows the presentation to be more accessible and reach new audiences.

There are improvements to be made however. The tool currently does not have all of the features that are available in JavaPPTX. As it stands, there are some presentations that are fully supported, and others that aren't. Certain types of objects didn't get included and certain overall methods were not added. This thesis adds the main infrastructure for the JS compilation, and many of the features in JavaPPTX. The remaining work for this compilation to be completed is not trivial, but many of the major problems with compilation have been solved. Put succinctly, the structure is built, but there are gaps to be filled.

In the coming years, I hope that this feature prompts more widespread usage of JavaPPTX and allows existing presentations to reach new audiences. While the systemic problems facing CS higher education will not be solved by this new compilation, it may help professors and students continue to teach and learn Computer Science effectively.

References

BLS, U. (2018). Computer and information technology occupations :occupational outlook handbook u.s. bureau of labor statistics. https://www.bls.gov/ooh/computer-and-information-technology/home.htm

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

Committee on the Growth of Computer Science Undergraduate Enrollments, Board on Higher Education and Workforce, Policy and Global Affairs, Computer Science and Telecommunications Board, Division on Engineering and Physical Sciences, & National Academies of Sciences, Engineering, and Medicine (2018). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. National Academies Press. https://www.nap.edu/catalog/24926

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

Roberts, E. (2016). A history of capacity challenges in computer science. (p. 24).

Starkman, R., & University, C. S. (2016). Stanford computer science launches new masters of education. *HuffPost*. https://www.huffpost.com/entry/stanford-computer-science_b_9713220

Zweben, S., & Bizot, B. (2018). Another year of record undergrad enrollment; doctoral degree production steady while master's production rises again. (p. 47).