Addressing Challenges in Creating Math Presentations

David Schweitzer (Liberty University)

David Schweitzer (B.S., Liberty; M.A., Houston; Ph.D., Lehigh) is an associate professor of mathematics at Liberty University, with prior positions at the University of Arkansas at Pine Bluff and the University of Central Florida. His varied interests include applied probability, actuarial education, math accessibility, operations research, and computing. His advisee Donald Honeycutt (now enrolled in a Ph.D. program at the University of Florida) contributed to this paper.

Abstract

For creators of mathematical content, all slide presentation composition platforms have at least one significant drawback. These can include complex learning curves, inefficient user interfaces, restrictive font selection, limited screen reader compatibility, and sometimes complete lack of math support.

We present various workarounds and solutions for the most popular platforms, including the authors' recently released software tools, SymbOffice and the TEX Math Here add-on for Chrome and Firefox. As a unified solution for math composition in almost any online platform or desktop app, TEX Math Here also has uses that extend beyond just math-centric presentations.

1 Introduction and Context

This paper is part of an ongoing research project addressing various issues that plague the electronic composition of math today. The project's initial focus was creating math slides in PowerPoint, and focus has extended to improving both the math capabilities of Web platforms in general and compatibility with screen readers. This project has also been a viable avenue for highly productive undergraduate research, including both software and peer-reviewed paper publication.

Due to its highly symbolic nature, the typesetting of mathematics has historically been a challenge. As the dawn of the computer age initiated the transition to a digital society, solutions to these challenges were soon forthcoming. Donald Knuth's groundbreaking 1970's research resulted in the revolutionary TEX system [1] whose syntax for electronically typesetting mathematics was so robust, yet intuitive, it is still used today. A decade later, Leslie Lamport's enhancement of TEX (dubbed LATEX) ultimately became "the *lingua franca* of the scientific world" [2]. These solutions (which, moving forward, will largely be treated as one entity) have now served mathematicians well for decades.

Unfortunately, limitations to this system do still exist.

- As a code-based implementation, a moderately steep learning curve results, especially for fairly standard tasks like page formatting and typesetting non-mathematical text. This learning curve often intimidates potential new users to the point of simple avoidance.
- Because writing mathematical documents was the initial focus of TEX/LATEX, branching out to other applications (such as presentation slides, spreadsheets, or web platforms) generally requires even more coding knowledge, and thus, greater challenge.

- Because the most common output of TEX/LATEX is PDF files, little flexibility for use in other platforms exists, at least by default.

Shifting focus specifically to presentation slides, several goals are desirable from the start.

- Ease of creation
- Accessibility for both the readers and content creators
- Robust capabilities
- Consistent appearance

For those interested, a further unpacking of these goals is presented in [3].

A Note on Scope
It would be impossible to grant full consideration to every available slide creation platform, including the rather obscure, and maintain a manageable scope for this paper. As such, we will clearly define how we will narrow our focus.

- **Widely adopted tools:** It is difficult to determine the exact market share of the various presentation software composition platforms, but inferences are possible. PowerPoint did claim 95% market share at one time [4]. Even if that number has declined in more recent years, it still held at least a 10:1 advantage over a prominent standalone competitor (Prezi) as recently as 2016 [5]. Also, if considering usage as part of a larger office suite, only two options have a significant market share: Microsoft Office 365 (44.44% combined from online and local installation usage) and Google's GSuite (55.37%) [6]. Therefore, PowerPoint and Google Slides will be the only prominent platforms explored.
- **Free tools:** In an effort to facilitate accessibility for content creators, we will focus only on add-ons for the GSuite and Office 365 platforms that are free; though it should be noted that even prominent commercial products (such as MathType) still suffer from many of the drawbacks we are trying to address.

We realize this narrowing of scope results in the exclusion from extensive discussion of several potentially useful solutions. These would include (but are not limited to) LibreOffice, reveal.js, stack.js, and Beamer. Depending on one's comfort level with learning new platforms and/or new coding processes, as well as any institutional requirements for accessibility, these tools could provide beneficial alternatives for some readers.

1.1 General Deficiencies in Current Popular Platforms

Two significant issues affect the major platforms to varying degrees. Additional platform-specific challenges exist as well and are discussed in Section 1.2.

- **Font selection:** Most computer users are accustomed to switching fonts simply by choosing a new one from some drop down menu. For math content, font switching is a much more

involved process, if it is possible at all. Considering that default fonts for text often differ significantly from default fonts for math, creating a consistent appearance between the fonts can be extremely challenging.

- **Screen reader compatibility:** Accessibility is becoming an increasingly important consideration when developing course materials, particularly as it relates to accommodation for disability. Whether due to the presence of students requiring such accommodations or universities proactively adopting institutional requirements for such, screen readers play a significant role in assisting students with visual impairments. While reading most traditional text is relatively straightforward, the same cannot be said for most math content. The most widely accepted accessibility solution for math is MathML; however, MathML code is inherently complicated.[1] Its composition requires yet another platform, and integration with Office 365 and GSuite is very poor anyway. Image alt text is far easier to implement while remaining a perfectly viable solution for screen reader compatibility. Unfortunately, embedding either of these solutions into a PDF (such as those created by LaTeX or Beamer) is still prohibitively difficult; while, at the same time, despite Microsoft's claims of screen reader compatibility with their native math content, both [8] and our own experiences would seem to indicate at least some debate of this point still exists. Lack of accessibility is a significant concern that casts at least some shadow over all current widely adopted implementations.

1.2 Platform-Specific Deficiencies

In addition to the above issues, the major slide creation platforms exhibit other drawbacks as well.

- **Google Slides and PowerPoint Online:** As of this writing, neither of these platforms offers any native math capabilities.
- **Local PowerPoint installation:** For tasks dominated by typing (such as slide creation), interaction with a graphical user interface (GUI) is inherently inefficient and greatly contributes to propensity for user input error [9, 10]. Because PowerPoint's math composition is, in general, GUI-driven, these issues are thus present. Going beyond the font selection issues plaguing all platforms, math font selection in PowerPoint is so rigid that creating a consistent visual appearance is nearly impossible (see [3] for examples).

2 A Preliminary Solution

An early result of this project, [3] presents a means of addressing (if not fully meeting) the four initial goals mentioned in Section 1 (ease of creation, accessibility for both the preparer and the reader, robust capabilities, and a consistent appearance). In summary, that solution used a local Windows PowerPoint installation, a free third-party PowerPoint add-in named IguanaTeX [11], and a pair of free fonts (Computer Modern's CMU Serif and Design Science's Euclid Symbol).

IguanaTeX, originally created by Zvika Ben-Haim and currently maintained by Jonathan Leroux, offers two features that really distinguish it from the many other similar offerings that exist (including Microsoft's own math tools).

[1] For example, Presentation MathML requires 56 characters spread across eight lines of code just to produce the extremely simple expression $x + 1$ [7].

- **Direct connectivity in PowerPoint:** There are multiple TEX-based solutions for Microsoft Office (such as TEXsword, LATEX To Word Equation, and Microsoft's own Math Environment); however, attention is predominantly focused on Word. IguanaTeX integrates easily and directly into PowerPoint, a feat with which even Microsoft's own TEX-based tools struggle significantly [12].
- **Editability of TEX-based expressions:** Once the mathematical expression is created from the TEX, it is exceedingly helpful if it can be edited for purposes of both error correction and the relatively quick construction of long derivations using copy/paste/edit. IguanaTEX (unlike some of the alternatives) fully supports editing the TEX code of an existing expression.

While the use of IguanaTEX provided an excellent foundation for meeting our goals, several shortcomings with the solution presented in [3] became apparent with use.

Limitations

- **Inline math symbols:** Because IguanaTEX generates images, placing such inline presents typesetting challenges when editing the surrounding text. We would prefer to easily type symbols if possible.
- **Font and color selection:** While the appearance was consistent (one font used for all math and text), changing that font to another font, or changing its color, was exceedingly difficult and unintuitive, especially to a novice.
- **Screen reader compatibility:** The initial solution addressed one of the many definitions of "accessibility" by providing a no-cost solution for content creators. However, screen reader compatibility was still not present.
- **Platform restrictions:** The solution existed only for a local PowerPoint installation on Microsoft Windows. Though the LATEXiT PowerPoint add-in should suffice for a local PowerPoint installation on Mac, other platforms were not addressed.
- **Local installation of LATEX (and Ghostscript, Image Magick, etc.):** IguanaTEX relies on a local installation (and configuration) of LATEX. Additionally, most common usages of IguanaTEX would also require the installation and configuration of Ghostscript and/or ImageMagick. Other configurations require yet more. When considered as a whole, this process is very user-*un*friendly, and proves prohibitively onerous for many novice users.

The primary focus of the remainder of this paper will be addressing these shortcomings. Section 3 will do so within the PowerPoint/IguanaTEX environment (using workarounds and a new tool named SymbOffice). Section 4 will present another new tool (TEX Math Here) useful for math composition that extends to other platforms, including those with little or no current math support (Office 365 Online, Google Slides, etc.).

3 Enhancing the Preliminary Solution

Because the preliminary solution in Section 2 provides a good starting point, we desired to address as many of its limitations as possible, while also instantly recognizing that certain issues (such as

the significant platform restrictions and local LaTeX installation requirements) would forever be beyond our control.

Balancing these considerations prompted an approach that focused more heavily on finding quick LaTeX and PowerPoint-based workarounds, even when enhancement of IguanaTeX was theoretically possible and could have resulted in a far more polished user experience. The appropriateness of this more limited approach to development was later confirmed when an update to one of IguanaTeX's dependencies (Ghostscript, specifically) created the false perception of multiple bugs in IguanaTeX.

Under this more limited approach, some software development did still occur, with the result being a useful macro for Microsoft Office named SymbOffice. Additionally, by ultimately avoiding modification of IguanaTeX itself, resources were freed up, thus leading to the development of TeX Math Here (see Section 4).

3.1 Inline Math Symbols

When one types typical American English text using a standard keyboard, generally all of the symbols one could possibly need are directly accessible via the keyboard. However, the same is not true for math and its hundreds of symbols that could be of interest (for example, $\div$ or π). While these symbols are included in almost all fonts via the Unicode Standard, the text encoding system that has had almost universal acceptance internationally [13], many of the symbols that are used in mathematics are not directly present on keyboards due to space constraints. Thus, while the characters exist, the historical challenge has been in accessing them.

The access method that is likely most familiar to most users is some form of symbol menu, which is opened (and searched through) every time a math symbol is to be inserted by the user. Of course, this GUI-based approach has the inefficiency issues mentioned earlier [9, 10].

A more generalized method for inserting Unicode symbols exists: keyboard codes, which are built into all major operating systems to allow the full Unicode character set to be used. Windows uses a set of alt-codes, which are a combination of holding down the Alt key and then typing a numerical code on the numpad that corresponds to the desired symbol. MacOS uses a similar system with its Option key and alphanumeric codes. While Linux also provides similar functionality, its implementation varies based on distribution and user configuration. Problematically for all platforms, these codes are neither elegant nor effective in their solution when needed on a regular basis. The associated codes are unintuitive to memorize (e.g., π is 227) and attempting to remember the appropriate code can break a user's workflow just as GUI use would. This is particularly an issue when considering the vast number of mathematical symbols that one might need.

TeX (specifically Knuth's Plain TeX) addressed this issue by assigning short, relatively intuitive names (prefaced by a backslash) to just about all of the symbols one could want (for example, `\pi` yields π and `\pm` yields $\pm$). Because these symbol names are far more intuitive and easy to learn, this methodology provides a useful framework for symbol access in any platform, provided it can be implemented.

SymbOffice

By repurposing existing Microsoft Office functionality, we can implement the use of TeX symbol names with relative ease. While most people would typically use autocorrect as a helpful tool for

correcting spelling errors, Microsoft allows user-defined entries into the autocorrect table, and these entries can be populated in such a manner that text strings (specifically, the TEX symbol names) autocorrect to corresponding symbols. Really, only two significant challenges must be overcome.

First, desired Unicode symbols must be entered into the autocorrect table, a process that would typically require knowledge of the numeric codes for all such symbols. Second, additions to the autocorrect table are typically entered one by one, as there is no built-in method to save or restore a custom list. With the sheer number of math symbols one could desire, this process would quickly become quite tedious for some quantity of symbols, as well as necessary on each computer one may desire to use.

SymbOffice [14] is a freely downloadable, macro-enabled Word document that automates much of this setup for a local Microsoft Office installation. It contains a VBA script that, when run, adds over 200 math symbols to the autocorrect table. Because of the tight integration of Microsoft Office's various products, these additions to Word's autocorrect table are automatically transferred (upon Office restart) to Excel, PowerPoint, and Outlook as well. This means inline math symbols can be created in PowerPoint simply by typing the corresponding TEX name followed by a space or punctuation mark. The conversion takes place automatically and is clearly visible to the user.

Because Microsoft's primary objective for autocorrect differs from ours, one peculiarity required a workaround. Symbols that, in TEX, have names that are capitalized (most notably, the capital Greek letters, but also various double arrows) must append the suffix "cap" to see the desired behavior in Microsoft Office. This is necessary because autocorrect, in its primary role as spelling tool, makes no distinction between different capitalizations of the same word. As an example, using SymbOffice (and unlike TEX), both `\Gamma` and `\gamma` would yield γ. If one desires Γ when using SymbOffice, they would need to type `\gammacap` (or `\Gammacap`).

There are several options additional symbols that are not currently included.

- Add the symbol using the standard method for adding autocorrect entries [15].
- Edit the SymbOffice macro by adding a new row to the file's symbol table, populating it with the desired auto correction, and re-running the macro.
- Make requests for symbol inclusion in future updates via the www.mathaddons.com website (or its linked GitHub repository).

3.2 Font and Color Selection

In a recent version of IguanaTEX, functionality was introduced to automatically import the font size information from the user's current text box to IguanaTEX for equation sizing purposes. It was immediately apparent that having similar functionality for the text box's font selection and font color would also be highly desirable. While implementing such technically is possible (thanks to IguanaTEX's open source nature and Microsoft's VBA functionality), significant differences in the way these parameters are handled by LATEX exist. Namely, font size is handled in the command line, while font selection and color are handled in the code's preamble. This difference prevented the solution from being as straightforward as it was for font size, as GUI changes would also be necessitated by the addition of any such feature.

Still, changing these properties remains possible using a code-based solution:

1. Download and install the `mathspec` and `xcolor` LaTeX packages. Doing so in advance will avoid otherwise likely PowerPoint/IguanaTeX program crashes. These occur because if one's LaTeX installation is configured to prompt prior to downloading and installing new packages (as is MiKTeX's default), IguanaTeX suppresses the prompt and IguanaTeX will crash due to timeout. In this scenario, resolution would then only be possible with reconfiguration of LaTeX to download automatically.

2. Change the LaTeX engine used by IguanaTeX to `xelatex`.

3. Add the following code to the preamble of IguanaTeX's code

```
\usepackage{mathspec}
\usepackage{xcolor}
\setallmainfonts{Font Name Here}
\definecolor{mycolor}{RGB}{0-255, 0-255, 0-255}
```

4. Replace "`Font Name Here`" with whatever display name the desired font uses in Windows (for example, "`Times New Roman`").

5. Replace each `0-255` with a number from that range for the red, green, and blue components of the desired color, respectively.

6. Place whatever content should have the new color within the second set of curly braces in the following statement: `\textcolor{mycolor}{Colored Text or Math here}`. The content within the curly braces can include math expressions and span multiple lines.

7. Click the "Make Default" button to avoid the need to perform steps 2 and 3 again with every new expression.

Note that multiple custom colors (for example, `mycolor1`, `mycolor2`, etc.) can be defined and used similarly. Also, as an alternative to defining a custom `mycolor`, LaTeX also recognizes 68 standard color names as detailed in [16]. Figure 1 summarizes steps 2 through 7.

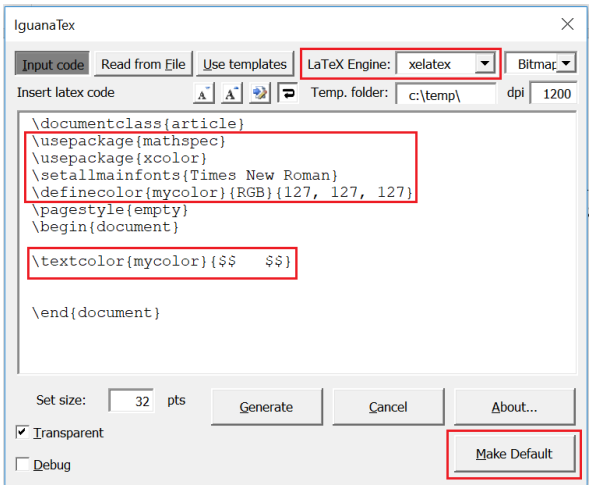

Figure 1: Screenshot of IguanaTeX highlighting the changes necessary to make the default be a math expression (enclosed by the $$'s) in a 50% gray Times New Roman font.

3.3 Screen Reader Compatibility

As a MathML-based approach would be near impossible to implement under this structure, using an image's alt text field is really the only realistic path to solution. Populating this field automatically would be ideal, but similar to the font and color selection issue, doing so necessitates changes to both the GUI and deeper functionality, thus raising issues of project scope. Similarly, a work around will be proposed instead.

However, in order for the question of *how* to populate the alt text field to even be relevant, a decision must first be made regarding with *what* it should be populated. The most straightforward solution (and the one proposed here) would be simply to populate it with the LaTeX code of the original math expression. As a strictly text-based, linear approach to math representation, LaTeX code is acknowledged by the MAA as a perfectly viable solution for facilitating screen reader compatibility [17]. Adding this compatibility manually is possible, as the following illustrates.

1. Create the image with math content using IguanaTeX.
2. Right click the image and select "Format Picture..."
3. In the new toolbar on the right, select the third picture in the top row (labeled with a ToolTip as "Size & Properties").
4. Populate the Title field, Description field, or both with the LaTeX code (or a relevant portion thereof).

Figure 2 summarizes steps 3 and 4.

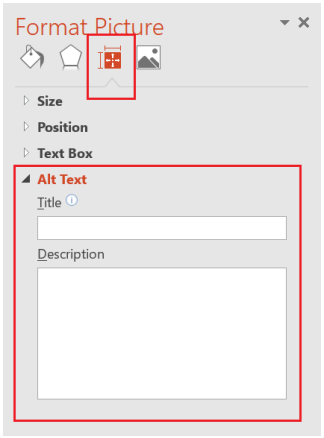

Figure 2: PowerPoint's alt text dialog with highlighting

4 TeX Math Here

As the limitations of the PowerPoint/IguanaTeX solution became more noticeable, it seemed the best approach moving forward was to attempt to build an alternative solution from the ground up. Not only does this provide a framework for implementation of all desired features, but it also addresses some longstanding issues regarding math on the Internet.

TeX Math Here [18, 19] is a browser add-on that converts TeX code to a .png image that can be pasted anywhere images are supported. In fact, even if direct image pasting is not supported by a

particular platform (as is the case with Blackboard, among others), the end result can be the same by embedding in the image metadata the URL for a remotely hosted version of the image. TEX Math Here is currently available for Chrome and Firefox, and the images generated can be pasted into not only Web platforms, but into desktop apps as well. This solution introduces math capabilities to Google Slides and PowerPoint Online, and also expands beyond the boundaries of presentation tools to addresses math composition needs in most other platforms as well (Blackboard, the rest of Google Drive's office suite, Microsoft Office Online, etc.).

In addition to providing a single, unified interface that increases the number of Web platforms that one can now use to compose math content, TEX Math Here also relies on an online LATEX renderer, thus eliminating all of the LATEX installation, configuration, and dependency issues discussed earlier, with the trade-off being that an online connection is required for functionality. However, allowing for a local rendering option (with the associated benefits and drawbacks) is a goal for future release.

TEX Math Here also streamlines solutions to the issues that, in the PowerPoint/IguanaTEX solution, required the workarounds outlined in Sections 3.2 and 3.3. In TeX Math Here, the image's underlying TEX code is automatically injected into both the title and the alt text fields of the image's metadata for screen reader compatibility. Additionally, a drop down menu (albeit with rather limited choices currently) is available for choosing a different font.

As of this writing, version 0.7 (screenshot in Figure 3) is available in the Chrome Web Store. Features include support for True Color in a variety of standard math fonts, the automatic saving of font characteristics within a browser session, and keyboard shortcuts that can allow one to avoid mouse use altogether. Mozilla's recent changes to permission handling have delayed version 0.7 release on Firefox, though version 0.6 is still available in Add-Ons. The only practical drawbacks of the earlier version are the loss of color functionality and fewer choices for font selections.

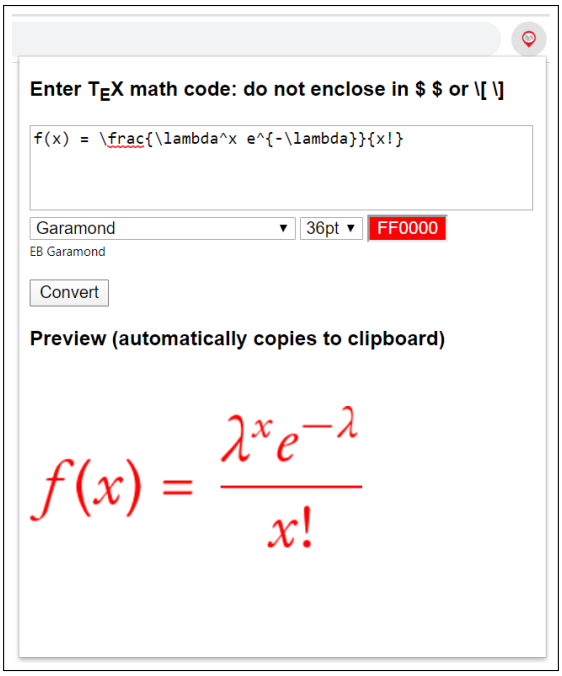

Figure 3: Screenshot of TEX Math Here converting TEX code to a math expression in red 36 point Garamond font. The expression's image is automatically copied to the clipboard for pasting elsewhere.

A Word on MathJax

A very common question raised with regard to this project has been about why MathJax [20] is not part of the solution. While MathJax is an excellent solution for writing mathematical content on the Web, it does have limitations, particularly in interfacing with other platforms. Specifically, the only image type that MathJax is capable of outputting is a scalable vector graphics (.svg) file. While .svg files have a host of advantages over other file formats (including those used by this project), overall support for .svg files is unfortunately rather limited. For example, neither Google Drive's office suite nor Microsoft Office (online or locally installed) support .svg files directly. The same holds true for both of the other output types MathJax can generate (MathML and HTML with CSS). Because of interfacing issues, such support would almost certainly need to come from Google, Microsoft, and/or MathJax developers. As such, implementing a solution based on MathJax is simply well beyond the scope of this team's capabilities.

5 Concluding Remarks

Even though much further development is needed, the early returns of this project have been most promising. Thanks to SymbOffice and some workarounds, a local PowerPoint installation can now provide a combination of efficiency, flexibility, simplicity, and accessibility that any other math presentation composition solution would be hard pressed to match. Additionally, the composition of mathematical content in almost any relevant Web platform or desktop application is now possible with a single browser add-on.

Another significant component of this project has been the ability to fully integrate undergraduates as researchers. The student co-author of this paper developed SymbOffice and the early functional prototype of TEX Math Here during the 2018-19 academic year. Two other undergraduates worked as research assistants over Summer 2019 developing the first fully featured release version of TEX Math Here and laying much groundwork for version 0.7. A fourth student has been enhancing and finalizing version 0.7, as well as configuring our own server during 2019-20.

As mathematics in particular can be a difficult subject in which to integrate undergraduate research, development of useful software tools could lower the barrier to entry for some students. This has required not only computer science knowledge, but also an introduction to a wide variety of other disciplines not typically associated with "math research," such as typography, GUI and graphic design, and accessibility.

6 Acknowledgments

This project is made possible in part by support from the Center for Research & Scholarship, Liberty University.

References

[1] D. Knuth, "Mathematical Typography," *Bulletin (New Series) of the American Mathematical Society*, vol. 1, no. 2, March, pp. 337–372, 1979.

[2] L. Lamport, *LATEX: A Document Preparation System*, 2nd ed. Boston: Addison-Wesley, 1994.

[3] D. Schweitzer, "IguanaTEX: LATEX PowerPoint Add-in Greatly Simplifies Creating Elegant Mathematical Slides," *Proceedings of the 30th Annual International Conference on Technology in Collegiate Mathematics*, 2019. Available at https://www.pearson.com/content/dam/one-dot-com/one-dot-com/us/en/files/DavidSchweitzer-SchweitzerICTCM2018.pdf

[4] Robert Gaskins Home Page. Available at https://www.robertgaskins.com/

[5] "How Prezi's Peter Arvai Plans To Beat PowerPoint," *Forbes*. Available at https://www.forbes.com/sites/forbestreptalks/2016/06/07/how-prezis-peter-arvai-plans-to-beat-powerpoint

[6] Market Share Category: Office Suites. Available at https://www.datanyze.com/market-share/office-suites

[7] "Working With MathML," *Wolfram Language and System Documentation System.* Available at https://reference.wolfram.com/language/XML/tutorial/MathML.html

[8] S. Bahram, D. MacDonald, & CB Averitt, "Enabling Math on the Web, in Word & PDF: Emerging Solutions & Overcoming Support Problems," *30th Annual CSUN International Technology and Persons with Disabilities Conference*, 2015. Available: http://www.davidmacd.com/mathml/making-math-accessible-CSUN-2015L.docx

[9] D. M. Lane, H. A. Napier, S. C. Peres, and A. Sandor, "Hidden Costs of Graphical User Interfaces: Failure To Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts," *International Journal of Human-Computer Interaction*, vol. 18, no. 2, pp. 133–144, 2005.

[10] E. M. Altmann, J. G. Trafton, and D. Z. Hambrick, "Momentary Interruptions Can Derail the Train of Thought," *Journal of Experimental Psychology: General*, vol. 143, no. 1, pp. 215–226, 2014.

[11] IguanaTEX. Available at http://www.jonathanleroux.org/software/iguanatex/

[12] M. Sargent, "LaTeX Math in Office." Available at https://docs.microsoft.com/en-us/archive/blogs/murrays/latex-math-in-office

[13] J. M. Aliprand, "The Unicode Standard," *Library Resources & Technical Services*, vol. 44, no. 3, pp. 160–167, 2011.

[14] SymbOffice. Available at http://www.mathaddons.com/ourtools.html

[15] "Add or remove AutoCorrect entries in Word," *Microsoft Office Support.* Available at https://support.office.com/en-us/article/add-or-remove-autocorrect-entries-in-word-e7433b94-f3de-4532-9dc8-b29063a96e1f

[16] Wikibooks, *LATEX*. Available https://en.wikibooks.org/wiki/LaTeX/Colors

[17] A. Maneki, "Guidelines for Collegiate Faculty to Teach Mathematics to Blind or Visually Impaired Students," *2015 CUPM Curriculum Guide*. Available at https://www.maa.org/sites/default/files/cupm/FacultyGuidelinesForTeachingVisuallyImpairedStudents.pdf

[18] TEX Math Here. Available at http://www.mathaddons.com/ourtools.html

[19] D. Schweitzer, D. Honeycutt, A. Lofgren, and D. Serban, "TEX Math Here: A Simple, Unified, and Efficient Model for Web-based Math Composition," *Electronic Journal of Mathematics and Technology*, vol. 14, no. 1, pp. 50–66, 2020.

[20] MathJax: Beautiful Math in All Browsers. Available at http://www.mathjax.org