Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Informatica
en Toegepaste Informatica

# Content Migration and Layout for the MindXpres Presentation Tool

Proefschrift ingediend met het oog op het behalen
van de graad van Master in de Toegepaste Informatica

Joris Vandermeersch

Promotor: Prof. Dr. Beat Signer
Begeleider: Reinout Roels


Mei 2015

Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

# Content Migration and Layout for the MindXpres Presentation Tool

Graduation thesis submitted in partial fulfillment of the
requirements for the degree of Master in Applied Computer Science

Joris Vandermeersch

Promotor: Prof. Dr. Beat Signer
Advisor: Reinout Roels


May 2015

# Abstract

Microsoft PowerPoint continues to be used worldwide in staggering numbers. We try to provide an alternative with MindXpres, facilitating the switch by converting existing Microsoft PowerPoint presentations into MindXpres presentations, and automatically fixing the layout in the process.

# Acknowledgements

*"Simplicity is a great virtue,*
*but it requires hard work to achieve it and education to appreciate it.*
*And to make matters worse: complexity sells better."*

— *Edsger W. Dijkstra*

# Contents

# Chapter 1

# Introduction

For over 25 years, Microsoft PowerPoint has been the market leader in digital prsentations. Admittedly, it was a revolutionary software package when it was first introduced, and its ease-of-use combined with its supreme graphical capabilities – at least compared to other software in the same era – quickly made it one of the most popular software packages in history. 25 years later, Microsoft PowerPoint can claim over 90% market share in presentation software, and on average 30 million Microsoft PowerPoint presentations are created every day.

In this time, Microsoft PowerPoint has gotten many new features, and certainly improved and grew with every new version, but it never really changed its core approach. It started out mimicking the then-popular and widespread use of dia and overhead projection slides, which was at the time a good way to convince people of its purpose, allowing them to feel comfortable with a familiar format instead of alienating potential customers with a new and potentially confusing interface.

However, this interface is quite restricting, and in recent years different approaches have seen the light of day. The zoomable user interface of Prezi is probably the most well-known, but apart from abandoning the traditional slide format it does little to improve or extend the concept of presenting information to an audience.

This is where MindXpres comes in. Its extensible plugin system allows anyone with some knowledge of programming to create new functionality to use in presentations. Examples are interactivity with the audiencer through various means, controlling the presentation from another device – or several! – and (re)modelling data while presenting it, based on feedback from the audience.

While this is obviously a big improvement on the traditional presentation model of Microsoft PowerPoint and the likes, it remains hard to convince

the general public of its merits. People are generally afraid of change, and it is important to make the transition as smooth as possible. On top of that, people are often worried that the work they did in the past may be lost – or worse, irrelevant – after switching to something new. This alone may be a huge factor in deciding wether or not to start using new software, or to stick with what they know.

That is where the subject of this thesis comes in. We aim to provide a way for people to convert their existing Microsoft PowerPoint presentations into MindXpres presentations, allowing them to take their previous work with them in their switch to MindXpres. This way, we lower the treshold for them to make the decision to start using MindXpres as their presentation software of choice. Once all their existing Microsoft PowerPoint content is available, usable and editable in MindXpres, it should be obvious to anyone why MindXpres is the better option for their presentations.

Another common problem with Microsoft PowerPoint presentations is the way they look. This is not necessarily the fault of the software; most people just are not trained in graphical design, and as such they know very little about proper layout, color choices, or slide content limits. Everyone has probably encountered slides with full paragraphs of text, too small to read and / or too much to process in the short time the slide is visible — (too) many people have made those slides themselves.

When we say this is not the fault of the software, that is mostly true, as the creators of these slides obviously made a conscious choice to make their content appear like that. It could be said however that Microsoft PowerPoint and other presentation tools are guilty through inaction. We believe it is possible to have software either warn its users against these choices and practices, or – even better – have the software fix these problems automatically.

One of the primary goals of MindXpres is to provide automatic layout, much like LaTeX does, ensuring that the content creator only has to worry about the actual content, while the software takes care of layout. In practice, both LaTeX and MindXpres currently use template-based layouts, where the contents' position is predefined in the template and not related to or based on its size, shape or nature. In the end, everyone who has ever used LaTeX knows that sooner or later you will struggle to get a certain image incorporated in the text correctly, ending up doing the layout yourself anyway, because the predefined template just doesn't work properly for you specific content.

As such, the second part of this thesis focuses on implementing true automatic layout in MindXpres. Again primarily to convince Microsoft PowerPoint users to switch, showing that their presentations actually could look better in MindXpres, while thus also providing new functionality to existing MindXpres users.

# Chapter 2

# Problem statement

## 2.1 Terminology

## 2.2 Microsoft PowerPoint

Microsoft PowerPoint was officially released in 1990, with Windows 3.0 (Austin, 2009). It had originally been developed as Presenter, but trademark issues caused a name change early on. It was also originally build for the Macintosh, which may seem surprising nowadays but was actually common practice back then since the Macintosh was widely regarded as a better development environment, more mature, more stable and capable of far better performance and visualisations. Some may argue this still rings true today.

Since then, it has grown to be the world's most popular slide show presentation program, alledgedly having been installed on over 1 billion computers worldwide, and being used on average 350 times *per second* (Parks, 2012). In 2012, it had a market share of 95%, leaving the other 5% to be shared by alternatives such as Apple's Keynote, Prezi, SlideRocket and others. While this number is declining, it may not be going as fast as many people think. As most readers of this thesis have heard before, over 30 million Microsoft PowerPoint presentations are created every day, for all kinds of purposes, with good and bad results both presentation-wise and goal-wise.

TODO this probably needs more content.

## 2.3 MindXpres

*This chapter's content is largely based on "MindXpres: An Extensible Content-driven Cross-Media Presentation Platform" (Roels, 2014).*

### 2.3.1 Introduction

The importance of digital presentations in this day and age cannot be understated. Millions of presentations are created every day, supporting the oral transfer of knowledge and playing an important role in educational settings. Their origins as tools for creating physical media such as photographic slides or transparencies for overhead projectors are still reflected in the underlying concepts and principles of slide-based presentation tools. The rectangular boundaries of a slide, and the linear navigation between slides, are still restrictions we face today in digital presentations. Tufte argues that these concepts of slideware have a negative impact on the effectiveness of knowledge transfer (Tufte, 2003). While the presenter is compelled to squeeze complex ideas into a linear sequence of slides, those ideas are rarely sequential by nature, resulting in a loss of relations, overview and details. An initial approach to address these issues might involve creating minimalistic presentations or introducing some structure via a table of contents. Unfortunately, this does not work in the domain of learning, where complex knowledge or other pieces of rich information need to be presented as is (Farkas, 2006).

It is important to point out the monolithic nature of slideware presentations where content is spread over many self-contained presentation files. In order to reuse previous work, the presenter has to switch between files while giving a presentation or duplicate some slides in the new presentation. Note that the issue is not limited to reusing single slides since there is a wealth of resources available, spread over a wide spectrum of distribution channels and formats. The inclusion of content by reference or transclusion (Nelson, 1995) might help to cross the boundaries between different types of media and be beneficial in the context of modern cross-media presentation tools.

There also seems to be an imbalance between the functionality for the authoring and visualisation of content. The main authoring views consist of toolbars and buttons to specify how content should be visualised while there is less support for the authoring of the content itself. While we have seen the addition of basic multimedia types such as videos to modern slideware, most content is still rather static. During a presentation we can, for example, not easily change from a bar chart to a pie chart data visualisation or dynamically change some values to see the immediate effect, which could be beneficial for knowledge transfer (Holzinger et al., 2008). Finally, the audience can be more actively involved via audience response and classroom connectivity systems which provide multi-device interfaces for sharing knowledge and results during as well as after a presentation. The evolution of presentations can be compared with the Web 2.0 movements where users have become contributors, content is more dynamic and interactive and where we

have a decentralisation of content via service-oriented architectures.

The rapid prototyping and evaluation of new concepts for the representation, visualisation and interaction with content is essential in order to move a step towards the next generation of cross-media presentation tools. After introducing existing slideware solutions, we discuss the requirements for next generation presentation tools. This is followed by a description of the extensible MindXpres architecture and its plug-in mechanism. The web technology-based implementation of MindXpres is validated based on a number of use cases and MindXpres plug-ins and followed by a discussion of future work.

MindXpres tries to be an alternative unlike other alternatives, stepping away from the classic slide format and introducing a plugin system to allow literally anything you can think of. The obvious choices are visualisations, animations, layout and the embedding of various media such as video, but it can also provide interaction with the audience, allow different presenters to take control from their own device, arbitrarily and ad-hoc change the way data is presented allowing to play into the audience's reactions unhindered by design decisions made beforehand, and so many other novel ways of presenting that captivate the audience instead of lulling them to sleep.

MindXpres was first created by my advisor, Reinout Roels, in 2012 as part of his Master thesis, and has since been built upon by himself and other students.

# Chapter 3

# Approach

## 3.1 Compilation process

The first part of the approach is fairly straightforward in its basic explanation: we had to convert Microsoft PowerPoint presentations into MindXpres presentations. This involves finding out how Microsoft PowerPoint presentations are structured, getting the parts wee need out of that structure, and then putting those parts together in de MindXpres structure. Since the author of this thesis has a small background in compilers (Vandermeersch and Timbermont, 2009) it did not take long to see the resemblance of this process to that of a compiler.

A compiler takes source code and transforms it into a working program with the semantics described by that source code. The compilation process consists of several steps. First the source code is tokenized, which means the symbols in the code are identified one by one and classified in certain categories.

Then the tokens are processed by a parser into an intermediary form called a parse tree. A parser looks for certain predefined patterns in the source code. These patterns are part of the source code's language syntax. As such, these two steps analyse and validate the source code's syntax. If part of the code does not match any pattern, the parser and the compilation process stop and the user gets a message saying the code's syntax is invalid.

When a parse tree is constructed, the compilation process can alter it, to improve it. Certain patterns in the parse tree may be replaceable by different patterns with the same outcome, but with more optimal execution. This part of the compilation process is optional, and is called compiler optimization. Optimizations can consist of many things, depending on the language. For example, some languages guarantee tail call optimization, where infinite

loops can be constructed by letting a function call itself as its last statement without causing a stack overflow. This is something the compiler (or interpreter) can optimize during this part of the compilation process.

After this, the parse tree can be written out to produce the desired output. Every node in the tree has a well-defined equivalent in the target language's syntax. The target language can be Assembly, which consists of the exact instructions a CPU needs to carry out a program, or it can be another programming language. Many compilers of higher-level languages translate their language into C, for several reasons: the C compilers that translate C into Assembly have been optimized so much that it is easier to rely on them than to put an enormous amount of effort into optimizing another language; C compilers exist for most – if not all – CPU architectures, which means translating a language into C makes it compatible with all those architectures, while it would cost a lot more effort to write different compilers for every architecture you would want to make your language available on.

The conversion tool that is the purpose of this thesis, can be described in a similar succession of steps. First, we take a Microsoft PowerPoint presentation and tokenize and parse it into an intermediary structure that allows us to perform other operations, or 'optimizations', on it. The intermediary form consists of a 'parse tree' containing the components of the original presentation — a component tree, if you will.

With this structure, we can construct a MindXpres presentation containing the same components in the same place, essentially creating a 'program' with the same semantic meaning as the original 'source'.

## 3.2 Compiler optimizations

Since the conversion process resembles that of a compiler, it seemed logical at first to make automatic layout a part of that process, as some kind of 'compiler optimization'.

At first, we tried to traverse the component tree, giving its objects new coordinates and sizes so that they would fit together on every slide as well as possible. This seemed an easy solution, but the results were sub-optimal. On top of that, we soon realised that we were in essence creating another template out of which a presentation would be made, which was exactly the opposite of what we were trying to do. As such, we abandoned this approach.

We then switched to a different method: defining constraints for every component, in the form of margins, maximum sizes and other limits, and then calculating a way to satisfy all constraints while fitting content together on each slide. While this is clearly a better method, it turned out the compiler

optimization phase was not the best place in the process to take care of this.

In the end, we decided to take a different approach, relying on the layout engine of MindXpres itself and enhancing that engine to create the automatic layout we wanteD.

## 3.3 Using MindXpres

MindXpres already takes care of layout for you, since that is one of its primary goals. The way it does this currently is however heavily based on templates, while we wanted a layout engine that could take any content and put in in an appropriate layout without any directions from the user. As such, we had to enhance MindXpres's layout engine to use constraints, based on the size of the content, and try to find an optimal position for every component it is given.

# Chapter 4

# Implementation

## 4.1 Taking Microsoft PowerPoint apart

We found Apache POI library very helpful in this part of the implementation. The POI Library – formerly "Poor Obfuscation Implementation" (Sundaram, 2004) – is a Java library that provides an API to access Microsoft document formats. The most mature (and most popular) part of it is HSSF, which stands for Horrible SpreadSheet Format, and which is used by Java developers worldwide to access Microsoft Excel spreadsheet data.

For our purposes, we relied on HSLF ("Horrible SLideshow Format"), which gave us access to a Microsoft PowerPoint presentation's contents in many ways. We could access all images at once, or every bit of text from the whole presentation, but the most interesting to us was the ability to access contents on a per-slide basis.

This allowed us to loop over the presentation's slides, converting them one by one, by placing the contents of each slide in a MindXpres slide equivalent.

## 4.2 Generating MindXpres

### 4.2.1 Plain HTML5

Since the MindXpres compiler was not functional during most of this thesis' implementation, we decided to generate an html file much like the MindXpres compiler would, including the MindXpres JavaScript library and plugins. This required us to first find out how MindXpres works on the inside, which proved to be a steep learning curve but gave us more insight into the software than we would've gotten if we only had to generate MindXpres XML and leave the rest to the compiler.

### 4.2.2 MindXpres XML

## 4.3 Creating layouts

### 4.3.1 Using constraints

### 4.3.2 Other ways

# Chapter 5

# Conclusions and Future Work

## 5.1 Contribution

## 5.2 Future Work

# Bibliography

D. Austin, *Beginnings of PowerPoint: A Personal Technical Story*, 2009.

B. Parks, *Death to PowerPoint*, 2012.

R. Roels, *MindXpres: An Extensible Content-driven Cross-Media Presentation Platform*, 2014.

E. R. Tufte, *The Cognitive Style of PowerPoint: Pitching Out Corrupts Within.*, 2003, graphics Press.

D. K. Farkas, *Toward a Better Understanding of PowerPoint Deck Design.*, 2006, information Design Journal + Document Design 14(2).

T. H. Nelson, *The Heart of Connection: Hypermedia Unified by Transclusion.*, 1995, communications of the ACM 38(8).

A. Holzinger, M. D. Kickmeier-Rust, and D. Albert, *Dynamic Media in Computer Science Education; Content Complexity and Learning Performance: Is Less More?*, 2008, educational Technology & Society 11(1).

J. Vandermeersch and S. Timbermont, *Compiling Mutually Tail-Recursive Functions Into Iterative Loops*, 2009.

E. Sundaram, *Excelling in Excel with Java*, 2004.