Multipresenter++

A Presentation System for Multiple Display Screens

by

Huan Li

B.Sc., Zhejiang University, 2010

AN ESSAY SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August, 2014

Abstract

Multipresenter++ (MPR++) is a presentation system designed to support visual presentations on multiple display screens. In addition to supporting multi-screen slide presentations, it also supports content-rich presentations that include audio, video, and live applications. MPR++ allows users to arrange the components of a presentation by specifying the screens on which each component will appear and the order in which the components will be shown. A multi-screen presentation can then be rehearsed on a single screen prior to giving the presentation in a multi-screen environment.

The features in MPR++ allow a presenter to take advantage of the facilities that are now commonly available in modern university lecture rooms, with multiple large display screens that can show content provided by a computer. Rather than using a single screen and switching between different types of content, each of the available screens can be used to display a different type of information in a content-rich manner, or multiple screens can show more relevant information around the same topic.

Table of Contents

List of Figures

Acknowledgements

Thanks my supervisor Kellogg Booth, for being very patient with my essay and gave me so many good instructions, which I really appreciate.

Also, I want to thank Dr. Joel Lanir's great effort designing and developing the original Multipresenter system provided the basis for MPR++. His initial implementation, deployment and testing lead to the possibility of adding the additional features that are in Multipresenter++.

Finally, I want to thank my parents and my girlfriend, since they give me so much encouragement and support and make me push harder to finish my degree. :)

Introduction

Electronic presentation is a practice of combining multimedia of image, animations and electronic files with the spoken words of the presenter. Nowadays, presentation is everywhere from business world to lecture rooms. In order to generate and display the presentation content, slideware tools, like PowerPoint, Libre Office, have gained enormous popularity. It is estimated that more than 30 million PowerPoint presentations are made every day, and over 6 Million Teachers around the world use PowerPoint for classroom lectures [Saurabh P., 2010].

Basically, all kinds of slidewares contain three parts: text editing, multimedia manipulating and slide-show presentation system. We use the first two parts to generate and design content, and use the presentation system to display content. However, more or less, everyone gained difficulty or dissatisfaction in all parts. Studies suggest that in lectures, PowerPoint best suits static and liner content while instructors prefer blackboard or whiteboard in some scenarios such as referring to the some previous key points, or comparing two or more similar concepts [Frey and Birnbaum, 2002].

Bad slideware tools also prevent user from manipulating presentation flow [Paradi, 2012]. One of the reasons lies in the lack of help during preparation period. Presenters could not fully practice their presentation without plugging their computer onto the real projectors in the conference room. So anything unexpected in real presentation could trigger the loss of the presenter.

Our motivation is also inspired by the availability of large display space in lecture rooms. With the growing number of audience in big conference rooms and lecture halls, multiple high resolution display systems are equipped to support electronic presentation (Figure 1 on page vii). However, the utilization of the equipment is far less than expected; many lecture rooms simply display same content on all screens.

Figure 1: Room with multiple external displays.

To solve these problems in presentation, we designed and implemented Multipresenter++, which extended the current Multipresenter .

- Multipresenter++ realized the philosophy that decoupled the reliability of presentation content on the external displays setup, such that it provides the most flexibility for presenters during authoring the presentation. The features and key advantages are:
- Multipresenter++ becomes more like an industry product, by supporting all software features like installation, one click project save/load. This greatly eases the use of the product.
- Multipresenter++ provides the ability to do content-rich presentation: Multimedia resources like videos, audios can be easily added as part of the presentation content, which will increase the dimension of regular presentation to make it more lively.
- Multipresenter++ realized its name as it supports real multiple displays. In theory, it can handle an unlimited number of external screens while the old version only supports two.
- The rehearsal mode in Multipresenter++ assists presenter for better preparation. They could view and practice the real speech under the simulated slide flow.
- Multipresenter++ makes it possible that you don't need to base your content on the number of external screens, or the screens resolution. It's basically "setup once and used anywhere".

Previous Work

PowerPoint, one of the most popular slideware tools, has gained a lot of criticism over the years. As pointed by Edward Tufte, professor and pioneer in the field of data visualization in Yale university, in his essay "The Cognitive Style of PowerPoint" [Lenth, 2004]:

"PowerPoint is used to guide and to reassure a presenter, rather than to enlighten the audience"

Slide tools are also prominent in education field. Usually, instructors use them as visual aids to organize and present their material. However, the presenter-orientated slideware tools only ease the instructor in preparing and presenting rather than bringing more effectiveness for students. Some researcher found slideware tools bring no significant improvement for students over traditional visual aids like whiteboards or blackboards [Levasseur and Kanan Sawyer, 2006, Szabo and Hastings, 2000].

Another flaw is traditional slide tools enforce the audience's lockstep linear progression through that hierarchy (whereas with handouts, readers could browse and relate items at their benefits). However, it's not only the lockstep for the audience but also a limitation on presenter's thought flow since it would subsequently prevent a lively and multi-layered presentation.

Apart from preparation authoring and delivery, presentation rehearsal also plays a really great role. Rehearse or not, sometimes determines the presentation result. However, not many slideware tools provide the rehearsal functionality especially when multi-dimension presentation increases the complexity for preparation, thus a useable rehearsal tool becomes crucial.

On the other hand, with the decreasing cost on technical infrastructure, more and more classrooms and lecture halls are equipped with multiple, high resolution display system. Researches have been conducted on making use of multiple screens to improve the quality and effectiveness of presentation.

There have been some projects focusing on creating more effective and interesting presentation. "NextSlidePlease" [Spicer and Kelliher, 2009] developed a prototype hyperpresentation system, which facilitates the creation of dynamic presentations and assists presenters in time management, by replacing linear deck flow with hypertext navigation. They use graph visualization to provide a platform for

clearly displaying the relationships among slides.

Andrés Lucero and Dzmitry Aliakseyeu [Lucero et al., 2009] introduced a wall-mounted display tool to do asynchronous presentation. They believe the challenge lies in the lack of overview of whole presentation slides. And the single linear manner of slides flow limits the versatility of presentation styles. They overcome these by introducing a set of presentation tools which could analyze the presentation layers (gesture, sound and visual), and convey the intended message or ideas of the presenter. One of the features is using gestures to skip slides, jumping back and forth, which does bring multi-dimensions into presentation.

Joel Lanir concluded [Lanir et al., 2010] that presentation using two screens with related content do significantly increases learning effectiveness, compared to duplicate content on both screens. The reason rooted in the fact that visual comparisons were improved with parallel viewing using extra screen real estate. Through the separation and integration process on information, long-term memory is established and cognition ability is enhanced [Mayer, 2002].

Current use of old Multipresenter

After Observing current use of visual aids, Joel found out the importance of having information persist for longer times, which would greatly improve the understanding of lecture structure in education scenarios. And spontaneous nature of blackboard inspired Joel with the idea of first version of Multipresenter, which directly leads to enhanced version: Multipresenter++.

Multipresenter (Figure 2 on page x) is a presentation system which takes advantage of multiple displays to improve mulch-dimensional learning.

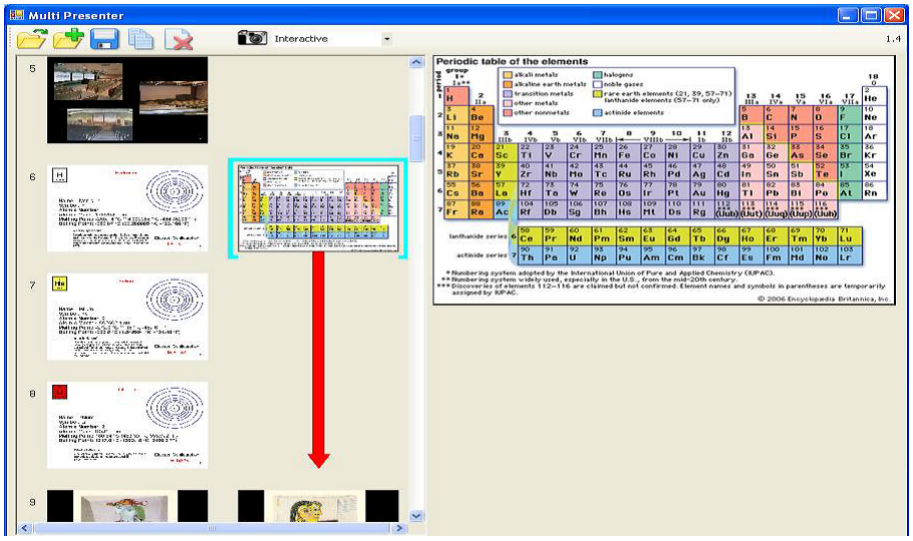


Figure 2: old Multipresenter

Information persistence. Current slideware tools do not provide the functionality to save slides for later use, which is common practice in traditional teaching process with blackboards. According to some statistics, 12 percent of material is referred later during presentation, while the longest refer-back is more than 20 minutes. Multipresenter takes use of one extra screen, and uses it to display earlier slides. This practice brings spontaneous, dynamic and non-linear advantage over traditional slideware tools. Common uses of this feature include comparing the

relationship two different objects with similar definition, or display the slide with questions or discussions on one screen while the solution stream is displayed on the other screen.

Multifarious modes. This feature greatly enriches the choice space during presentation. Presenters can change between different modes to get the most flexibility. For example, users can use the secondary screen to show previous slides by specifying the modes, according to their relevance. Usually, presenters can display the regular stream of slides on one screen while showing one, two or four previous slides on the other screen, or display two similar slides for comparison, or show the regular stream on one screen while persist the overview on the other one. More importantly, Multipresenter offers the mode to use secondary screen as the sketching book, which can be used to support improvised design and dynamic commentary. Also, Multipresenter supports instant change between these modes, which bring more convenience during presentation.

High adaptability. Screens fragmentation remained a problem in display system. There is no standard for size or resolution of display screens in lecture rooms. Any system without considering this problem would bring disaster during presentation. Multipresenter would detect all screens first, save the specification, and use different strategies for different screens. Multipresenter innovatively offers the ".mpr" file, which is the configuration file for the presentation slides. Users could easily get the mpr file from any version of PowerPoint slides. Every time after they design the dynamic presentation over Multipresenter, they can save the streams as mpr file, which can be read into the tool from one click, even on a computer without MS PowerPoint installed. These features make Multipresenter more portable.

Spontaneous presentation style. To receive the consistency nature of information flow, Multipresenter simulates whiteboards to preserve some previous slides, which could fully make use of multiple screens, for the benefit of the audience. While on the users (like instructors) side, Multipresenter offers the cut-and-paste tool, which makes the presentation authoring process on dual-screen more spontaneous and straightforward. Users can modify the content on one screen, with cut-and-paste action from the other screen, before or during the presentation. This feature outstrips traditional whiteboards and provides more dynamic ingredient into presentation.

Multipresenter greatly improves the concept of presentation, with the help of multiple displays. However, here are some **LIMITATIONS**:

Barely no extensibility. With the popularization of multiple screens in lecture halls and conference rooms, three or more large resolution displays are available to the presenters and audience. This brings the potential of more flexibility and versatility in presentation; however, it would also demands challenge to authoring part, which needs to put more all screens into consideration. But, Multipresenter is

not designed at the first place to support more than two screens or do any feature extension, so the source is hard-coding everywhere which increase the difficulty for easy integration. Also, no source control is setup for collaboration.

No support for content-rich presentation. To make the presentation more attractive and versatile, almost all modern slideware tools support authoring rich contents, such that videos, audios become part of the presentation content. However, Multipresenter only allows adding static images, so if you want to launch a video clip, you need to quit your presentation, open the video player, load your file and drag it to the external screen.

No support for rehearsal. Rehearsal is always an essential part for modern professional slideware tools. Especially in the scenario of multiple screens, with the highly dynamic nature increases the complexity during presentation, if the presenter cannot get some practical assistance to rehearse, the multi-screen style might have the exactly opposite effect.

Highly coupled content authoring with external setup. With Multipresenter, you can never give a two-stream presentation onto one external screen. This can be very frustrating to presenters since they will lose the opportunity to give multi-layered presentation just because they don't have enough external displays.

Usability issues. Some usability problems prevent Multipresenter being easily operated. For example, to save your current workspace, the save button will create you two files (.mpr, .ink) and one folder(contains all images), and if you want to use them on some other computers, you need to copy all of them over. Also, to launch a premade Multipresenter project, you need to click on the load menu, and select the .mpr file. What if both save and load is just one click on one single file, like current PowerPoint supports?

Design philosophy

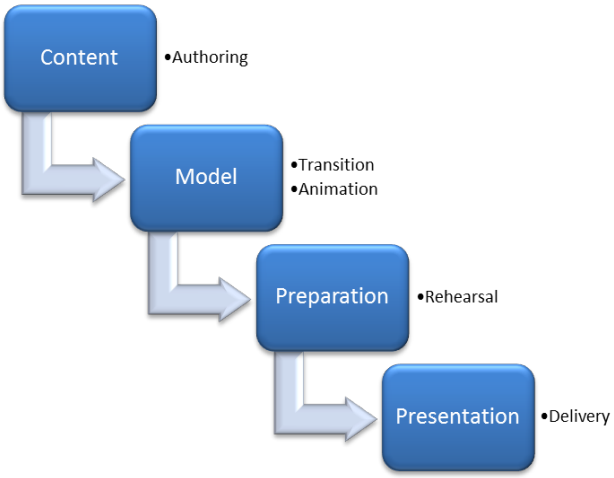


Figure 3: Workflow of a presentation.

Separate presenting from authoring

PowerPoint, like most of other slideware tools, offers great assistance in preparing and authoring presentation slides. Various slide elements, like textbox, table, flash and video, greatly enrich the content generation. Highly sophisticated layouts and templates also ease the process of organizing these elements. All these features increase the flexibility into the authoring part in preparing presentation.

However, as mentioned above, PowerPoint has very weak functionalities to impose a dynamic presentation which could support all sorts of presenting styles. Instead, PowerPoint has most of its advantages in sequential or highly structured content, which is not a popular choice in active and improvisational presentation. The main reason of this failure lies in the fact that PowerPoint wants to integrate authoring with presenting, which are two most important but also quite different parts in the whole presentation practice. These inconsistencies inspired the idea of separation presenting from authoring, and led the implementation of a presentation

system that assists more in final presenting.

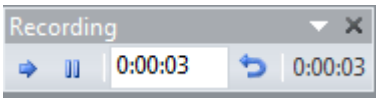
Separating content and presentation is a common design philosophy and methodology applied in lots of fields. On the web, we use HTML or similar markup language to generate content and use CSS to style web pages, including elements such as layout, colors. The common design objective behind this is to improve content accessibility, provide more flexibility and control in the specification of presentation characteristics. This separation defines different layers, and increases the independence between them, such that, each layer can be developed by different people/method for different use cases. The same idea applies to presentation, for example, the same content can be presented differently just by specify different style in a presentation tool, like what we achieved in Multipresenter++.

In order to implement this philosophy, we integrated with Microsoft PowerPoint seamlessly for content authoring and focus Multipresenter++ more on presentation such that same content can be presented differently, based on presenters' preference.

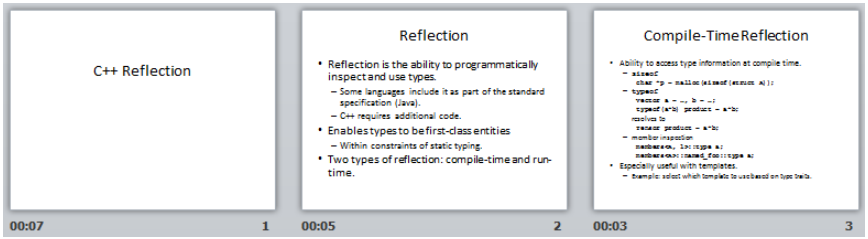
Separate rehearsal from delivery

As the saying goes "If you fail to prepare, you are prepared to fail". Rehearsal is a popular practice, commonly used by people in all public presenting activities. In theatre, performers usually ensemble a dress rehearsal before the audience several rounds before the final show. A professional orchestra or chamber ensemble rehearses a piece in order to coordinate the rhythmic ensemble and ensure that the pitches of the different sections match. So, separating rehearsal from final delivery has been widely accepted and used, which also plays a really essential part in preparing a presentation. It is commonly performed to ensure that all details of the subsequent performance before the final delivery are adequately prepared and coordinated. "The best presentations are rehearsed, not so that the speaker memorizes exactly what he or she will say, but to facilitate the speaker's ability to interact with the audience and portray a relaxed, professional, and confident demeanor." [Collins, 2004] Also, a lot of tips have been given to guide an effective rehearsal [Holzl, 1997].

To better help rehearsal, PowerPoint offers the Slide Timing feature (Figure 4a on page xv) to record the time spent on each slide during rehearsal, and then give a summary about time usage on all slides (Figure 4b on page xv). Also, users can set the recorded time to advance slides automatically when performing the real presentation.



(a) Timing in PowerPoint



(b) Each slide timing information

Figure 4: Rehearsal in PowerPoint

This feature did, to some degree, assist rehearsal. However, in dynamic presentation scenarios, it's completely different. Presenters need to take control of multiple streams of slides instead of the singular flow in PowerPoint, so only timing the rehearsal is not nearly enough, they want to visually rehearse all streams, which is quite a challenge in dynamic presentation. In Multipresenter++, we developed rehearsal mode to get high-level and all-directional assistance both in performing and preparing the presentation.

Separate content preparation from external hardware setup

In multi-screens presentation world, this is a very possible use case: presenter has been using Multipresenter for a while, and he is very familiar with his premade two stream slides which always works very well on two external screens. He suddenly was required to do the same presentation but in an old-style room, which only has one external screen for use. He became very pressured since he hasn't prepared any slides for one single external screen. It is also very challenging to combine two stream slides into one stream without harming the original good presentation effect.

The root cause for the presenter's pain described above is: his prepared content highly relies on the external display restriction. If the number of content streams doesn't match the number of external screens, the presenter needs to update the premade content accordingly.

In Multipresenter++, we decoupled this dependence, such that, the premade

presentation content is once-for-all[1] solution. The presenter can prepare the content without thinking about any external hardware restrictions, which greatly increase the flexibly of doing real "multi-dimension" presentation.

[1] Once setup, used anywhere.

Implementation

Design Goals

Separation of rehearsal and delivery

In old Multipresenter, the primary target is the separation of content and presentation to support most flexibility. Through Multipresenter, multiple presentations can be created and stored referring to the same source content. With the support on more large displays, users have the opportunity to make more dynamic presentation which would take advantage of extra displays. But how to deliver fluent presentation becomes a challenge, with the growth of dimensions in control space (multiple slides are not only related in space and time, but also in content). The most effective solution is to provide professional rehearsal tools in current presentation system, to separate rehearsal from delivery. In Multipresenter++, users can simply turn on "rehearsal mode", which could assist a detailed walk through on their laptop, without connecting the laptop to any external displays. During rehearsing, presenters can clearly see the dynamic arrangement of slides on each screen, which would not only facilitates practicing the speech, but also inspires some content updating.

Separation of content preparation and external restriction

When authoring their presentation, the presenters should not restrict the content based on how many external screens do they have. They should focus more on how to present the content better and how to make the slides more comprehensive and attractive, than how to fit their content into one or two external screens. Multipresenter++ should allow users to show their presentation on any number of external screens without changing their premade content.

Support content-rich authoring

Multipresenter++ should allow presenters to add different kinds of rich resources including images, audios, videos. Users can also specify the exact start time, end time of video/audio play.

Dynamic presentation on three or more large displays

With the prevalence of multiple displays in lecture rooms, the ability to support three or more screens gradually becomes a key requirement. In Multipresenter, the default setting is only on two displays, which would limit the flexibility and portability in future presentation. While Multipresenter++ should extend this and work seamlessly for three or more external displays.

High usability in practice

Usability is one of the most important review criterions, especially in systems with complicate functionalities. To simply put, the system must be easy to learn and use at three levels.

First, we should only need simple infrastructure support and lightweight deployment. The "how-to-use" for Multipresenter++ should follow the convention in professional software world, like PowerPoint, which supports one-click installation and one-click workspace loading.

Second, the system must be compatible with the current presentation practice. For example, people usually use PowerPoint to design their single-flow slide deck then load it into Multipresenter. In Multipresenter++, all versions of MS PowerPoint are supported. Users can minimize their effort of transitions from PowerPoint to Multipresenter++ by a single load-and-run click. Also, a multi-stream presentation can be created by simply loading multiple PowerPoint files, which is not included in previous Multipresenter.

Lastly, the system self must be intuitive to use. Users should not feel any confusion between different options. The effort must be minimized on the interaction with interface, so that most energy can be focused on designing and delivering the presentation itself.

Promotion on learning and understanding.

A presentation system is different from regular application, since presentations are usually carried in public, where audience are also critical participants and also users of the system. So any user-centric design must include the audience and consider their requirements. Mostly, presentations are given in lecture halls or conference (meeting) rooms. Accordingly, audience aim at learning or understanding the materials presented. So, any presentation system that cannot ease material understanding would be useless.

Task Examples

Kevin Tyler is a professor in Computer Science apartment. He has worked in the fields of computer engineering for 30 years. He began to teach CS 246, Object-Oriented Software Development, since 20 years ago. At the early days of his teaching, he used blackboards to organize and deliver the lecture. He always rehearsed several times before the lecture to guarantee every concept would be explained clearly and all materials were easy to understand. There were 8 blackboards in the classroom, four by two. At the beginning of each lecture, he would write down today's lecture outline on the top left blackboard, which would be kept through the whole lecture. The rest content are organized left to right, up to down. Sometimes, he need to refer to an earlier concept and make some comparisons (like the difference between object and class), so he would underline the key points if they were still on the blackboard. Sometimes, students were asked to finish a question on the blackboard, like drawing a UML diagram of a specified class.

With technique advances, Kevin switched to PowerPoint to prepare and deliver the lecture presentation. At first, he could not adapt well with this tool since he can no longer easily switch from one piece of content to another. Even with the help of hyperlinks inside the slides, it was easy to get lost in the slides stream. So he doubled the preparation time to make up for this problem. Also Comparisons are harder to see clearly because the screen was not large enough to fill all points, he can only choose to show one point at a time. But gradually, he adjusted his presenting style and became adapted to the regular one-stream presentation provided by PowerPoint.

Jessica Kacy is a undergraduate student, and majors in Economics. Currently she is taking the course Labour Economics. Usually, she would print out the handout and do some preparation for today's topic. The course is carried out in a standard lecture room with two big screens. Jessica always wondered the purpose of two screens in the room, because they always displayed the same content. "One larger screen with more content space would be far better than two screens with duplicate materials." She thought. The course covers many economical concepts which confused Jessica a lot, so sometimes she wishes the instructor could give some terminologies comparisons. On the other hand, since the course is on a quick pace, many students including Jessica cannot catch up well with the material flow on the screen. Sometimes, a totally new page would flash into before she can digest the concepts on the older one; Sometimes, a question would refer to the idea covered in an earlier page, however, she cannot remember clearly. Since the learning efficiency is quite low in class, Jessica has to spend much time back home to catch up.

Features Design and System Description

We intuitively divide the presentation practice into three parts: authoring, rehearsal, and the final delivery. Each phase contains different design goal and features as described below:

Presentation authoring This stage involves the creation of slides streams to be presented over multiple screens. Our system focuses on increasing the dynamic and multidimensional nature of presentation, and this should be distinguished from traditional slidewares, like PowerPoint, which focuses on creating more fancy content on slides using hyperlinks and animations. Basically, we want to build a bridge to fill the gap between slides authoring to final presenting, which we call it presentation authoring.

To integrate better with PowerPoint, we have some legacy issues to fix. Firstly, as being developed many years ago, and lack of maintenance, old Multipresenter only supports PowerPoint 97-03 version, and will crash on any later versions. So, this needs to be fully extended in Multipresenter++. Secondly, since designed to only work for two external screens, Multipresenter assumes users would only load one single PowerPoint file to the workspace, which does not work for real multi-screen scenario, since users will quite possibly load more PowerPoint files. In Multipresenter++, users can load as many PowerPoint files or multimedia files as they want, they can also specify the exact position they want to put the resources. These two improvements make sure the integration with PowerPoint is seamless.

After loading the PowerPoint slide deck into workspace, it occupies one slide column inside the window panel, and we call it one stream. Usually, one stream maps directly to one external screen. Then users can build multiple streams by manipulating the current slides or adding more resources. For example, some intuitive and conventional operations like copy/paste, drag/drop, are supported to move slides inside or between streams. As designed, Multipresenter++ supports multimedia resources like videos, audios and images or even another PowerPoint file. To achieve this, simple click on the "Add resource" menu icon and specify the resource directory, together with the exact coordinance where it sits.

Multipresenter++ also allows authoring one stream content using another stream's previous slides. One epic use scenario is that the instructor wants to compare one term with the other term introduced earlier by putting the old slide from stream A to stream B, then he can easily achieve this by adding "dependent resource"[2] , and specify how "previous" it will be: Here we use "N-slide-back" mode to in-

[2]Dependent resource means this resource is dependent on some resource on the current streams. Independent resource is simple external files like images, videos...

dicate that we want to stream B to show previous N slides of stream A. Another common scenario is that the instructor puts the questions on one slide followed by analysis and answers on the next several slides. But it may not be that efficient since students may not be able to grasp the relations between questions and the corresponding analysis. To do this, instructors can use Multipresenter++ to put the question on the secondary screen while display the analysis/solution on the primary screen.

Finally, to improve the robust of the system, we must guarantee after leaving the system, users can still get what they left when they come back. The implementation is to provide the save and load mechanism into the system. This is achieved by using the ".mprx" file. When users decide to stop their presentation authoring, they can save the whole workspace in the ".mprx" file and load it later by one single click, which is what PowerPoint currently does.

Presentation rehearsal Rehearsal plays a crucial role during presentation process which however, didn't earn enough attention in original Multipresenter. The lack of efficient rehearsing tools increase the complexity and thus time consumption during preparation. Especially in multiple screen domain, the extended dimension leads more flow control issues. We do want to improve the dynamic and versatile presentation styles, but not on the cost of highly increased preparation time. In Multipresenter++, we provided the option to go through the whole presentation in pretty much the same way we rehearse single stream presentation. This will greatly reduce the unknown issues during real presentation.

To rehearse in Multipresenter++, turn on the "rehearsal mode" and follow the operation just like you will do in real presentation, e.g. progressing through slides, sketching on screen, resizing the slide, etc. You will see exactly what a multi-screen presentation looks like just on your laptop.

Presentation delivery The greatest feature in this stage is to provide some powerful tools to take full control of the presentation content during actual presentation. This is where in-presentation dynamic get implemented.

First, users can copy and paste content between. In this scenario, screens are used as clipboards. For example, instructors may think some portion of current slide is quite important that could be referred for a while, so he can select that portion and drag it to the other screens which would be kept until users' manual deletion, also users can put away key points to other screens and sum them up in the end; users can also drag several slides together, and each slide would be resized proportional to the available area. Further, any selected, pasted items can be moved, resized and erased according to users' preference. This feature is also

WYSIWYG (what you see is what you get), any modification made on the laptop's presentation window is immediately reflected on external screens.

On the other hand, ink panel increase the dimension of presentation delivery, and collects the sparkles during presentation. Electronic ink can be used on different screens, from annotation to sketching, from different colors to different sizes. Also, users can choose to erase the ink or retain it for the next speech. It's especially useful when instructors need to elaborate ideas or give keys to previous answers. This feature is also WYSIWYG, which makes the interaction both efficient and effective.

Lastly, large canvas window can be used to do some careful drawing since it will display the current slide as full screen on your laptop so any tiny ink strokes can be made correctly.

Implementation

Platform choosing

The initial implement of Multipresenter and Multipresenter++ is on Windows platform. To be more specific, the system is running over .NET framework. There are several reason for this choice. First, Windows is still the most used Operating System, and office suite has gained lots of prevalence over decades. So, we decide to target MS PowerPoint as the slideware tool and make easy integration between PowerPoint and Multipresenter++. Second, .NET framework is an integrated framework which provides libraries and runtime for easy interaction with MS products like PowerPoint. We use C# for all the development, which is "Java" in Windows. The features provided in this development environment make the version update and code maintenance easier and more efficient.

One of the future improvement is to make Multipresenter++ cross-platform, since for the moment, it's not available to be used on Linux or MacBook. Considering the success of Java and its great crossing platform support, we plan to migrate the system into Java in the future.

Implementation procedures

Upgrade from Visual Studio 2005 to Visual Studio 2010 Multipresenter is originally hosted as a vs2005 project which use .NET 2.0 framework. While vs2010 has some useful modeling tools such as Architecture Explorer, which graphically displays projects and classes and the relationships between them. Also, it provides easier code organization, so we decided to use VS2010 environment to rebuild the whole project. Visual Studio provides an automatic project upgrade

tool, but the conversion will lose some key connection between classes. So, we build the project from bottom to top. Rewrite every class until they follow the standard rules.

Naming conversion The first step in the new project environment is to rename classes and method to make them more intuitive and meaningful. For example, one of the Windows form is named Form1, which can hardly reflect the behind intention. Actually, it's the form which manipulate the display content on each screen, so we change Form1 into FormDisplay. The whole revision is included in Appendix.

Bug fixing Since Multipresenter is only a prototype, it contains many bugs which effects the user experience. So, bug fixing becomes important part of the early implementation stage. The general description of main bugs and corresponding fixing are listed below, while details code can be found in Appendix.

Bug description	Bug fixing
When loading PowerPoint files into Multipresenter, it will launch Microsoft PowerPoint without approval, user need to close the PowerPoint manually.	Multipresenter uses COM interface of PowerPoint to dump all slides into .png files. However, the default visibility of the dump procedure is true. We fix this bug by move dumping procedure background.
Since Multipresenter is developed early, it only support PowerPoint 2003. PowerPoint 2007 and 2010 are not supported.	Change the file filter of open file dialogue, to add file extensions like .pptx, such that all PowerPoint versions can be loaded.
It crashed with no sign, if you select multiple ppt files to load into the workspace.	Add try catch block for multiple files loading, show friendly reminder that PowerPoint files should be loaded separately.
Multipresenter would dump all the slides in PowerPoint files as .png pictures and save them into a temporary directory, but does not clean it up after exit.	Add clearTemp method to guarantee clean up temporary files each time we load a new workplace or exit the program.
Scrolling using mouse wheel is not working properly in the main form, you cannot scroll up or down.	Add appropriate code to handle the mouse scrolling event.

Bug description	Bug fixing
Does not allow dragging a slide to a new column if the new position has no slides currently, which in other words, users can only switch slides but not move them freely.	Enable all kinds of the drag and drop.
Fail to detect all display screens in order. Intuitively, we want to manage the screens in the exact order we see them. Like the leftmost screen should be the first external screen, and rightmost be the last external screen.	Detect the screen sequence and automatically rearrange them by their nature location order. Make sure what we see is what we get.
Multipresenter assumes all display screens have the same size, which would shut out some area for bigger sized screens or leave some blank for smaller ones.	Make default to stretch the picture according to the screen size, such that slides would occupy the full screen.
When using large canvas form for interactive drawing or annotating, it use the screen on the laptop but specify the canvas size as extra screens.	Change the canvas size and resize the slide on that screen to the laptop.
Scrolling is not working in Presentation form.	Add mouse wheel scrolling handler in FormPresentation.
Multipresenter only works when Office default language is English. For example, if the default language of your PowerPoint is Chinese, you cannot use Multipresenter.	The loading is hard coded that, it use the “slide” keyword to do the sorting of all dumped slides files. In Multipresenter++, we don’t need that assumption, and works for all languages.
Multipresenter would crash when navigating to the last slide in Presentation form.	The bug is caused by missing check on hitting boundary. Fixed by adding some check message.
After loading a totally new PowerPoint deck, the selected slide in last PowerPoint deck still displayed on the top right corner.	Clear all control, including the picturebox on the top right corner, after each new PowerPoint dump.

Bug description	Bug fixing
When using electronic ink, the mapping from presentation window or large canvas to external screen is not maintained.	Choosing the correct resizing property when constructing the mapping at the first place.

Architecture Analysis As a really large and self-contained project, Multipresenter is built on a complicated, hierarchical architecture, in which not only exist lots of classes, also the relationship could bring huge pain in any modification or extension. So, the step right after bugs fixing is to reconstruct the blueprint of the system design and glue our own bricks onto the building.

Below is a diagram[3] built by Visual Studio 2010, to describe the whole project architecture, including all classes and their relationship. As we can see, all kinds of interactions flooded through the relationship network. To better explain this blueprint, I will briefly describe each class and the relation (call or reference) between them.

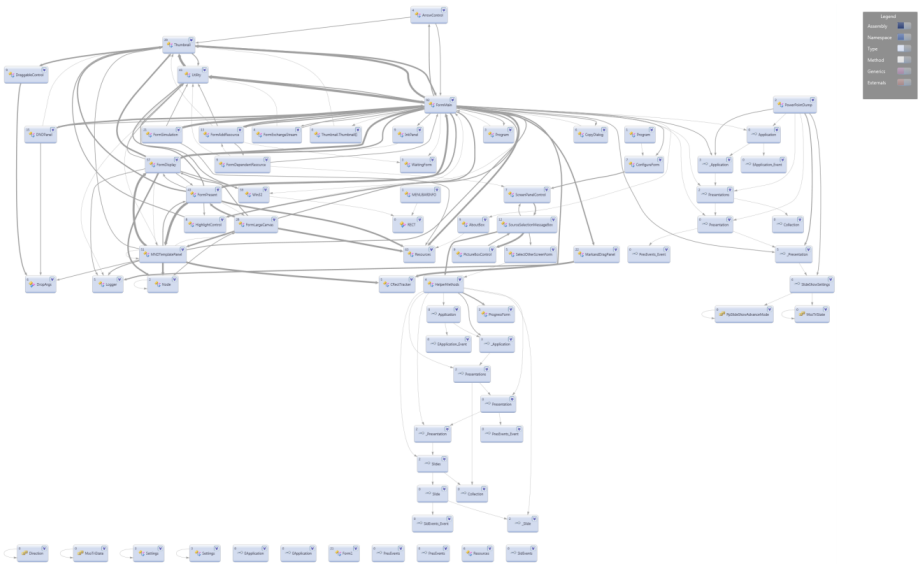


Figure 5: Class diagram by Visual Studio

Logger:

Regular logger class, creates log directory and log files. Update log files by append method.

[3]Details can be found at http://www.cs.ubc.ca/~bruceli/files/class-diagram.png

PowerPointDump:

A utility class which provides a static method to dump PowerPoint into pictures, with size 800*600.

WaitingForm:

A form with a progress bar. It would show up during dumping PowerPoint, and automatically get closed when finished.

DraggableControl:

The basic control class for a thumbnail, which would decide whether the drag distance is large enough for an effect and record the drop point after dragging, which would be passed to the drag-and-drop handler in descendant class.

Thumbnail:

The display unit in authoring panel, this is a subclass of DraggableControl, which means it can be dragged to other screens in the authoring panel. Basically, we create a bitmap image from the image file created by PowerPointDump. It would be hidden when an arrow occupy its place.

ArrowControl:

When we want to keep one slide for a sequential positions on one screen, we want to show that persistence in authoring panel. This is done by put ArrowControls over the sequence slides.

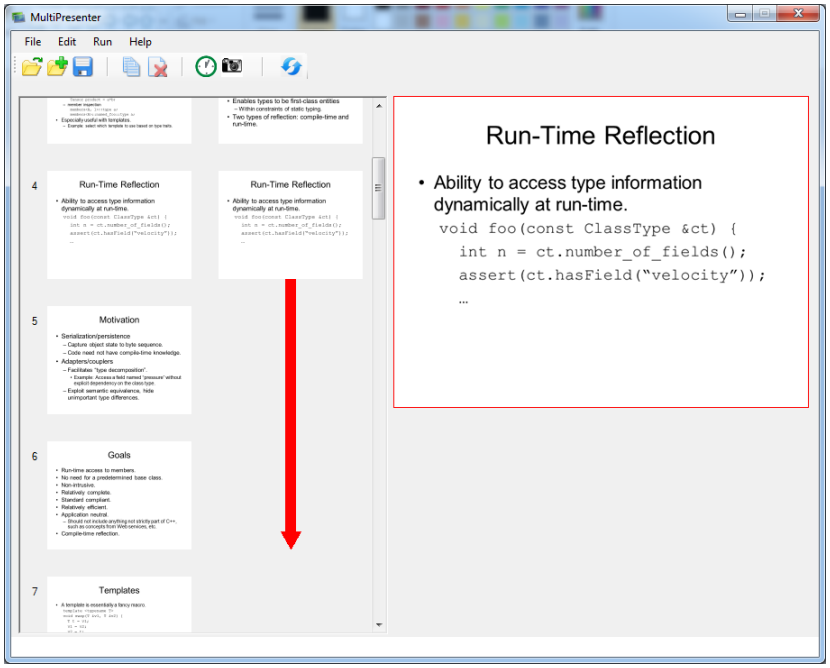


Figure 6: Arrow control inside workspace view

HighlightControl:

This control is used to highlight the selected thumbnail. Basically, each time a thumbnail is selected by click, the highlight control would be assigned to that thumbnail. We can see an aqua-colored rectangle surrounds the target thumbnail to distinguish it from the rest [figure]. Also, when a thumbnail is selected and highlighted, it would become the default object of the copy and paste operation.

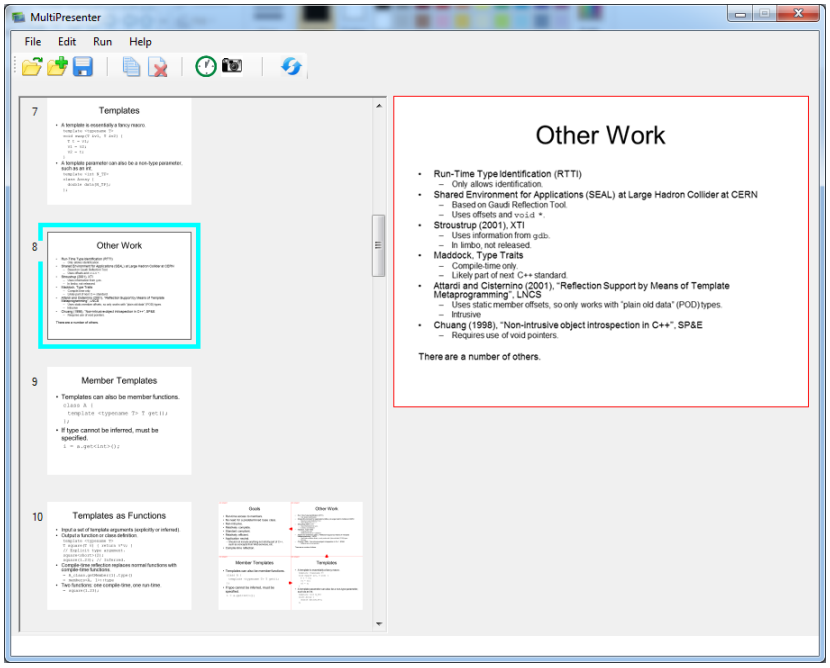


Figure 7: Highlight control inside workspace view

DNDPanel:

The panel that contains all thumbnails. This panel would handle events like drag-and-drop by catching the parameter (DropArgs) threw by thumbnails. And then do the corresponding position change for intended thumbnails.

Node:

This class is used to represent the boxes created by dragging on the panels, like MarkandDragPanel. The attributes include the rectangle it holds, the image on that rectangle, and also the pointer to previous and next Node. It can be used as linked list, in which the order corresponds to their Z depth on the screen.

CRectTracker:

This class provides some public method, like draw focus rectangle [footnote] on the Node. Also, we draw small tracker rectangles for resizing the Node.

InkPanel:

This panel acts more like a form. It's transparent and cover the real formDisplay, which is used to display slides on real screens. So, it has the exactly same size and location as the parent formDisplay. Also, it's the main form to display the inks onto the screen every time we paint the form.

FormLargeCanvas:

This is the form which we used as a canvas when we want some interactive drawing. Usually, we create a canvas form and associate it with content on one of the screens. We can sketch on the formPresentation, but the panel for drawing is too small so we provide the largeCanvasForm to use the whole screen on user's laptop for sketching. And any update on the canvas would be reflected on the real screen.

MarkandDragPanel:

This panel is used to preview the selected thumbnail. Uses can create, resize, delete the Nodes on this panel by doing some basic drag-and-drop operation. This is part of the authoring process since the Nodes would be reserved on real presentation.

Utility:

Just like all utility classes, our Utility class provides many common and reused functions, which can ease other classes handling the workflow. Some of the most useful functionalities include maintaining (adding, indexing, sorting, moving, deleting) a list for all thumbnails. Also, it provides methods for project loading (read the .mpr file for thumbnail source and their position in the panel) and saving (save the ink, image files for thumbnails and their position in the panel).

FromMain:

The main form and also the entrance of Multipresenter. This form works as the environment for initial basic presentation authoring. The functionalities provided in this part include loading PowerPoint or project files, adding PowerPoint decks, adding external multimedia resources, or thumbnail maintenance for example, rearrange positions like exchange slides, move between slides or delete slides.

Extended features Support loading multiple PowerPoint files into workplace

The original Multipresenter did not take the need of loading multiple PowerPoint files into consideration, which however, is quite useful in parallel presentation and comparison. We implement this feature by rewriting the click listener of adding files button, such that it supports loading multiple PowerPoint files and would increase the column position in the workplace for each new slides deck.

Add multimedia resources

In Multipresenter, users can add multimedia resources like videos, audios into

workspace as their presentation content. This is achieved by adding AxWindowsMediaPlayer object which is the root object for the Windows Media Player control. When advancing to the next slide, we first detect the type of the slide, if it's audio/video, we launch player widget with the file path, otherwise, we stop and hide it.

Get the thumbnail of the video/audio resource

We use ShellFile to retrieve the thumbnail as the slide for the multimedia resources. Then we draw the type of the resource, i.e. video, sound to the top left of the thumbnail to indicate the slide is not only a picture.

"About" menu

An "about" menu is created to show some basic information about Multipresenter++, including the author, brief introduction, user manual download link.

Menu bar

Old Multipresenter doesn't have a menu bar which means users will not be able to see the menu's explanations until hover mouse on it. In Multipresenter++, menu bar is created with explanations and categorization such that all means are categorized which makes it easier for beginner users to understand and use.

Rehearse presentation

In Multipresenter++, users have the ability to rehearse their multi-screen presentation on their laptop/desktop. By switching to rehearsal mode, the presenter can see exactly what the real presentation look like on multiple external screen, by mapping the contents on external screens onto a minified screen on their laptop/desktop, which will ease the preparation of the multi-screen presentation.

Software installation

Multipresenter++ can be installed with one single .exe file which is the convention on Windows platform. It is implemented by creating a separated "setup" project inside Visual Studio 2010. Then we specified the install target as our Multipresenter++ exe file and then configure all the ".mprx" file extension to be associated with Multipresenter++.

Save/load workspace

Before quit Multipresenter++, you will now be warned whether you want to save your whole workspace. To implement the saving feature, we created a temporary file, then loop through all thumbnails and grab their file source, put them inside the temporary file, then created a configuration file which specifies the exact position and source, type of the thumbnail which we call it "mpr" file, also we save all inks inside ".ink" file. Finally we tar-ball the temporary directory into our ".mprx" file.

Loading is exactly the same way but in reverse order: first un-tar-ball the ".mprx" file to get the temporary directory, then load the ".mpr" file so we can restore all thumbnails with correct information.

Exchange streams

Sometimes, you find the stream order is not what you want and want to change them, then you can click on the "exchange" menu icon, and specify the two streams for exchange. The implementation behind is loop through all thumbnails, change their stream information as requested.

More than two external screens support

Previously Multipresenter hard-coded the external screens to be primary screen and secondary screen, so all operations need to be applied on either of it. To extend and make it real "multi-screen", we created an list to held all external screens and associate them correctly with their corresponding panels on the presentation window, which is also stored inside a list. The panel size on the presentation window will not be hard-coded but taking number of panels(external screens) into consideration.

Right click to add resource

If you know exactly where to add resource, you can simple right click on the spot and choose add resource. From where you will see some position information fields have been pre-filled for you. This is implemented by detect where the user clicks and fill the position information with that.

Toggle stream number

Inside presentation window, sometimes you feel confused on the mapping between streams and the real external screens. This can be checked by the "stream toggle" menu icon. The implementation is fetch stream number from the external screen object and then show the text label with the stream number.

Copy and paste across different screens

Multipresenter by default, only support copy the selected slide to second screen, which needs to be extended, and be able to paste on any screen. In Multipresenter++, users get the opportunity to specify the target screen and start, end position of the slide on that screen. Accordingly, we update the arrow control to denote the persistence of that specific slide on different screens instead of only on the second screen.

Update the Z order of all Nodes in the panel

In MarkandDragPanel, users can create, resize nodes by dragging on the image and save them for presentation. These nodes are stored in the time order they are created, however, when we select one node, we intuitively want them to be brought to the front, which means an update on their order should be executed on the nodes list. This is done in Multipresenter++.

Other features and rationale

Industrialized Multipresenter++ In order to increase the usability, we brought more industry software features into Multipresenter++. We implemented the setup program to install the Multipresenter++ as a software onto machines. After that, open a Multipresenter++ file is just simply one click.

The benefits of using a setup program to deploy Multipresenter **Consistent**. The most common way of using a program is to install the program and then use the shortcut to open the executable software. Multipresenter is definitely a program, so this rule should also apply. Every time someone wants to try and use Multipresenter, all they need is a installation file and follow the usual route as they use other software.

Clean. Now, Multipresenter is not only an single executable file. To improve the user interaction experience, we used some extra Windows runtime library to make Multipresenter more usable. For example, we support add video as a source during presentation authoring. We will display the video thumbnail as a demonstration of the video source for distinguishment. In order to use that functionality, we adopt WindowsAPICodePack runtime library as other libraries are used. This means if we want someone to use Multipresenter, they need to copy those libraries as well, which is neither intuitive or extendible when more and more libraries would be added. However, use a separate setup program for Multipresenter, all these dependencies are automatically compressed into the installation file and distributed to the file system afterward. This will save pain for users.

Easy. Previously, when someone wants to bring his premade presentation to another computer, he will need to take the Multipresenter executable, the .mpr file, the .ink file, and the thumbnail images all together. The files look like this:

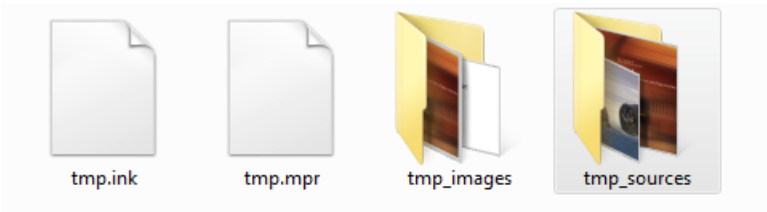


Figure 8: Workspace files with old Multipresenter

Which is kind of pain, any missing from those files would not make Multipresenter work. And it can easily be forgotten what those files are used for after some time which bring more work for clean-up. Also, to open this presentation file, they need to open Multipresenter, click on open menu and select the .mpr file in order to use, which is really not a straightforward practice compared to PowerPoint

or Word. To fix that, a separate setup program provides the ability to associate a file extension to a specific software. In our case, we use .mprx to achieve that. x can be interpreted as “extended” or “executable”. So to replace the previous tedious steps, users would only need to focus on a file with mprx as extension, like:

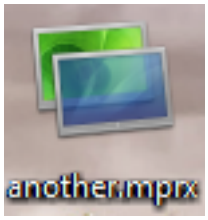


Figure 9: Workspace files with Multipresenter++

This file is actually a zipped file which contains all four files mentioned above, with a thumbnail demonstrate the intention of this file. What left is double click the file and users would see the presentation authoring panels like they never left.

Version control the source code Since Multipresenter will be an ongoing project, and need some collaboration in the future. Tarball the source and copy it over is a good way for source code changing any more. The process can be very error prone and lack the log tracking on the changes, which also makes the code harder to understand.

Let’s imagine, currently all source code sit on my computer, if it crashed, all source code might be lost; In the future, people will hand their current work over to someone else, while using email is not a practical way if we consider about security risks; When someone wants to refactoring the previous versions, it would be hard for them to trace all the changes of the previous version and the rationale behind.

All these concern can be handled by version control. But sadly, previous Multipresenter does not have one, which makes the refactoring quite painful.

New Multipresenter uses Git as the version control software, with bitbucket as the hosting website. Developers need to register an account and follow the common routine of using Git. Then we can enjoy the big benefits for that.

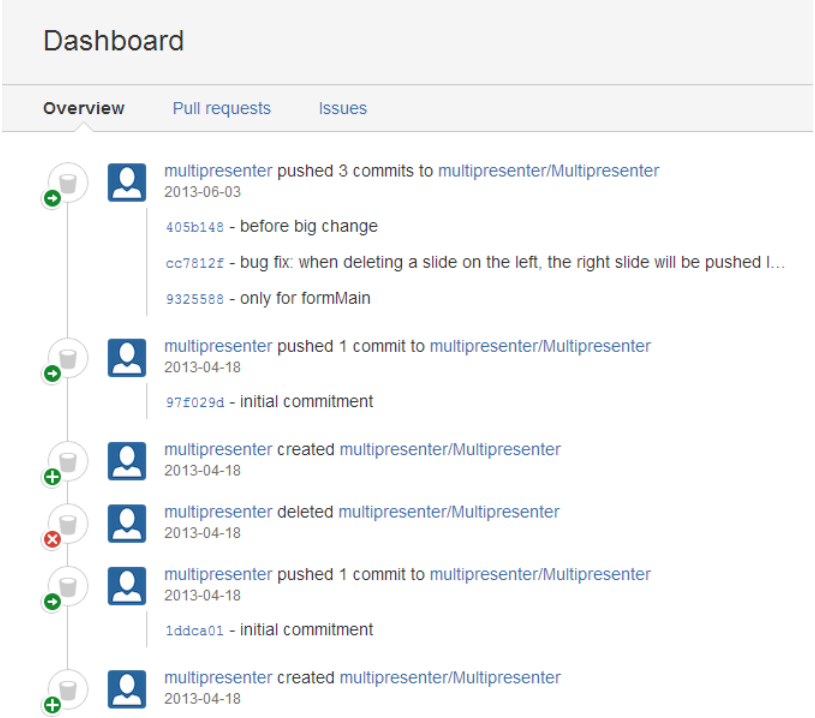


Figure 10: Version control for Multipresenter++

As we can see, all changes are tracked like trees, each commitment is documented with a change message, this reduces the confusion in understanding what the changes do.

Bibliography

Jannette Collins. Education techniques for lifelong learning: Giving a powerpoint presentation: The art of communicating effectively 1. *Radiographics*, 24(4): 1185–1192, 2004.

Barbara A Frey and David J Birnbaum. Learners' perceptions on the value of powerpoint in lectures. 2002.

J Holzl. Twelve tips for effective powerpoint presentations for the technologically challenged. *Medical Teacher*, 19(3):175–179, 1997.

Joel Lanir, Kellogg S Booth, and Kirstie Hawkey. The benefits of more electronic screen space on students' retention of material in classroom lectures. *Computers & Education*, 55(2):892–903, 2010.

Russell V Lenth. The cognitive style of powerpoint. *Journal of the American Statistical Association*, 99(466):569–569, 2004.

David G Levasseur and J Kanan Sawyer. Pedagogy meets powerpoint: A research review of the effects of computer-generated slides in the classroom. *The Review of Communication*, 6(1-2):101–123, 2006.

Andrés Lucero, Dzmitry Aliakseyeu, Kees Overbeeke, and Jean-Bernard Martens. An interactive support tool to convey the intended message in asynchronous presentations. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, pages 11–18. ACM, 2009.

Richard E Mayer. Multimedia learning. *Psychology of Learning and Motivation*, 41:85–139, 2002.

Dave Paradi. Are we wasting $250 million per day due to bad powerpoint? 2012. http://www.thinkoutsidetheslide.com/are-we-wasting-250-million-per-day-due-to-bad-powerpoint/.

Navkar S. Edward C. Kentaro T. Collage Saurabh P., Aakar G. A presentation tool for school teachers. *In Proc. Info. & Comm. Tech. and Development*, 2010.

Ryan P Spicer and Aisling Kelliher. Nextslideplease: navigation and time management for hyperpresentations. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 3883–3888. ACM, 2009.

Attila Szabo and Nigel Hastings. Using it in the undergraduate classroom: should we replace the blackboard with powerpoint? *Computers & education*, 35(3): 175–187, 2000.

Appendix: Bug fix log

1. In method

public int Dump(string path, string outpath, int width, int height, int screenwidth, int screenheight){

if (Path.GetExtension(path) != ".ppt")

throw new ArgumentException("Input file must be a PowerPoint (.ppt) file", "path");

...

Presentation pres = app.Presentations.Open(path, Microsoft.Office.Core.MsoTriState.msoFalse, Microsoft.Office.Core.MsoTriState.msoTrue,

Microsoft.Office.Core.MsoTriState.msoTrue);

=>

if (Path.GetExtension(path) != ".ppt" \&\& Path.GetExtension(path) != ".pptx")

throw new ArgumentException("Input file must be a PowerPoint file", "path");

...

Presentation pres = app.Presentations.Open(path, Microsoft.Office.Core.MsoTriState.msoFalse, Microsoft.Office.Core.MsoTriState.msoTrue,

Microsoft.Office.Core.MsoTriState.msoFalse); // don't show the window

2. buttonOpen_Click

this.DisplayThumbnails(openFileDialog.FileNames);

=>

Try {

this.DisplayThumbnails(openFileDialog.FileNames);

}catch (Exception) {

MessageBox.Show("please only select multiple image files");

}

3. clearTemp

=>

try{

if (PPoutputpath != "") {

if (Directory.Exists(PPoutputpath)) {

Directory.Delete(PPoutputpath, true);

}

```
}
}catch (System.Exception e) {
MessageBox.Show(e.ToString());
}
```

4. Mouse Wheel

```
FormMain\_MouseWheel
this.HorizontalScroll.Value = this.HorizontalScroll.Value + 1;
=>
int val = panelThumbnails.VerticalScroll.Value - 50 * Math.Sign(e.Delta);
if (val > panelThumbnails.VerticalScroll.Maximum)
panelThumbnails.VerticalScroll.Value = panelThumbnails.VerticalScroll.Maximum;
else if (val < panelThumbnails.VerticalScroll.Minimum)
panelThumbnails.VerticalScroll.Value = panelThumbnails.VerticalScroll.Minimum;
else
panelThumbnails.VerticalScroll.Value = val;
```

5. Drag and drop instead of switch

```
if (toThumb == null)
{
thumb.xPos = xPos;
UT.MoveUpSlides(yPos + 1, END);
return true;
}
```

6. Rearrange the screen by location

```
Screen[] s = new Screen[numDisplays - 1];
Array.Copy(System.Windows.Forms.Screen.AllScreens, 1, s, 0, numDisplays
- 1);
Array.Sort(s, delegate(Screen s1, Screen s2)
{ // put the leftmost screen first in the array
return s1.Bounds.X.CompareTo(s2.Bounds.X);
});
```

7. Change canvas size

```
Size screenSize = (Size)fp.frmMain.GetDisplaySize((int)(panel.bigDisplay.sType+1));
this.AutoScaleBaseSize = new System.Drawing.Size(0, 0);
this.ClientSize = new System.Drawing.Size(screenSize.Width, screenSize.Height);
=>
Size screenSize = (Size)fp.frmMain.GetDisplaySize(0);//(int)(panel.bigDisplay.sType+1));
this.Location = new Point(0, 0);
this.ClientSize = new System.Drawing.Size(screenSize.Width, screenSize.Height);
```

8. Mouse Scrolling in Form Presentation

```
private void thumbPanel\_MouseEnter(object sender, EventArgs e)
```

```
{
thumbPanel.Focus();
}
private void thumbPanel\_MouseWheel(object sender, System.Windows.Forms.MouseEventArgs
e)
{
int val = thumbPanel.VerticalScroll.Value - 50 * Math.Sign(e.Delta);
if (val \&gt; thumbPanel.VerticalScroll.Maximum)
thumbPanel.VerticalScroll.Value = thumbPanel.VerticalScroll.Maximum;
else if (val \&lt; thumbPanel.VerticalScroll.Minimum)
thumbPanel.VerticalScroll.Value = thumbPanel.VerticalScroll.Minimum;
else
thumbPanel.VerticalScroll.Value = val;
}
9. language version
private class CustomComparer : System.Collections.IComparer
{
public int Compare(object x, object y)
{
string s1 = (string)x;
string s2 = (string)y;
if (s1.Length \&gt; s2.Length) return 1;
if (s1.Length \&lt; s2.Length) return -1;
for (int i = 0; i \&lt; s1.Length; i++)
{
if (s1[i] \&gt; s2[i]) return 1;
if (s1[i] \&lt; s2[i]) return -1;
}
return 0;
}
}
...
Array.Sort(str, new CustomComparer());
3.4.1
if (this.openFileDialog.ShowDialog().Equals(DialogResult.OK))
{
if (openFileDialog.FileName.EndsWith(".ppt") || openFileDialog.FileName.EndsWith(".pptx"))
{
xPosGlobal = UT.GetNumberOfCols();
try
```

```
    {
    PPoutputpath = openFileDialog.FileName;
    PPoutputpath = PPoutputpath.Substring(0, openFileDialog.FileName.EndsWith(".pptx")
? PPoutputpath.Length - 5 : PPoutputpath.Length - 4);
    PPoutputpath += "- MultiPresenter files\\";
    if (!Directory.Exists(PPoutputpath))
    Directory.CreateDirectory(PPoutputpath);
    wForm = new WaitingForm();
    wForm.Show();
    // do the loading work, including dump ppt, save it in imageList
    backgroundWorker1.RunWorkerAsync(PPoutputpath);
    }
    catch (System.Exception)
    {
    MessageBox.Show("Unable to write to disc and therefore unable to open a
PowerPoint presentation.");
    }
    }
    else
    {
    this.DisplayThumbnails(openFileDialog.FileNames);
    DrawThumbnails();
    }
    }
```