LivingSlides
Extended Abstract

Joana Borges Pereira

October 2014
Abstract

The majority of classrooms now have computers and projectors, thanks to the popularisation of technology. Due to that, teachers use projected slides using tools, such as Powerpoint and Keynote, to help them deliver a lecture. Whenever a student poses a question or an additional explanation is needed, the teacher must switch the context or draw schematics on a whiteboard, all because presentations are static. Such explanations, much like presentations, are also static. It would be ideal if teachers did not make a context switching and the explanations were not static, like for instance, watch the values of a mathematical equation change or watch a graph react to input values. LivingSlides main goal is to help teacher make their presentations interactive, and by using Active Documents the presentation itself becomes interactive. When using LivingSlides, the teacher is able to add settings to elements so they become interactive with one another, so that later on, if a doubt comes up of if the teacher feels it is necessary, to have “life” injected onto the slides and the information is better presented. LivingSlides relies on web technologies such as HTML5, CSS and JavaScript to achieve the active part of the presentation. It also uses libraries, like Tangle, which help to add interactiveness to text or help to draw SVG shapes. User tests were made and they showed LivingSlides is an easy-to-use tool. Comments made by users encourage the development of LivingSlides furthermore, for they show people have an interest in different ways to deliver their presentations.

Keywords

Slideshow presentation program, Active Documents, Interactivity, Slides

1 Introduction

With the expansion of technology, most classrooms are equipped with computers and projectors. Due to it, many teachers are now using tools such as Powerpoint and Keynote to help their classes, and even though technology is now a common place in classrooms, presentations are still what they used to be when teachers used overhead projectors and transparencies back in the day: they are static. Whenever a student has a question, the teacher must switch context and draw something on the board, or show something online. This breaks the focusing and attention of students. It would be interesting to see a tool which would accomplish all of this without having to switch back and forth between media.

Active Documents are not new, in fact they appeared with the World Wide Web, when HTML documents became popular. An Active Document is a document which content changes to requests and needs of users. At first, they were designed to make a bridge between server-centred applications and client-centered applications.

LivingSlides aims at helping to change the way presentations are delivered, and how they are edited too. By using the concept of Active Documents, with help from several frameworks, LivingSlides brings interactivity to the presentations. Users when using LivingSlides can set up variables and then watch text, images or shapes respond accordingly to changes made on one variable. By creating presentations which are interactive, teachers can make them captivating and motivating for students; teachers can now answer to questions without having to stop their presentation.
Multimedia learning is learning done with the help of pictures and words. There are several studies on this matter, and they’ve come to the conclusion people don’t learn the same way: some people prefer words to pictures and others prefer pictures to words. A good multimedia presentation must satisfy both groups. Taking this to LivingSlides it means LivingSlides must support both ways of learning, by having an introduction slide where concepts are explained in words followed by another slide with has the same information but graphically explained, for example.

LivingSlides uses HTML5, CSS3 and Javascript to support itself. These technologies are the standard of web development and because of that you have a good support from the community. Furthermore, by using these technologies anyone can access it anywhere as longs as they have a browser and an internet connection. LivingSlides also supports SVG, for every shape drawn in LivingSlides is an SVG shape. Every element is interactive in LivingSlides, and that was achieved with the help of a library and some extensions performed on it.

2 Related Work

Research on related work showed there have been some developments on Active Documents and Multimedia Presentations. For example, on ActiveDocuments there is a paper written by Guy Boy in 1997 which talks about Active Design Documents. This approach introduces Active Documents as a way of reducing costs on manufacture of prototypes by helping users understand how products work simply by using an active document. He believes active documents should be a bridge between product manuals and the product itself. His idea for this approach was the fact that humans often use drawings on paper to expose their ideas.

On Multimedia Presentations, for example, there is NextSlidePlease. This application makes it easier for presenters to know their own presentations by helping them decide which path to take if they run out of time, or if they have to stop for some reason. It searches the document for similar words and builds a weighted graph with the paths a presenter can follow and how much time it will take to follow a path. This reduces cognitive load on the presenter, for he no longer needs to know every single slide in his presentation, NextSlidePlease does that for him by suggestion which can be the next slide.

There are also frameworks which emulate Active Documents, with Tangle being the most important and the most popular. Tangle was created by Bret Victor, an Apple ex-employee, and it is designed for people to explore what they are reading, something he calls a Reactive Document; users analyse and change what the user had in mind and observe their consequences.

Reveal.js is a framework which served as inspiration for LivingSlides. It is a framework which allows users to create highly customizable presentations. This framework can be used as a library, if a user has advanced knowledge of HTML and Javascript, or as a web application by accessing the website slid.es, if one is not too comfortable with HTML or Javascript. This framework was refurbished recently, with the ability to add shapes and the possibility to drag and resize every element of the presentation.
LivingSlides was implemented using web technologies such as HTML5, CSS3 and Javascript. This has the advantage of being independent of any Operating System and the user does not need to install anything to use it. The disadvantage of using such technologies comes in the form of old or incompatible browsers, such as Internet Explorer, which do not support some elements used on LivingSlides. LivingSlides was tested on Google Chrome, Mozilla Firefox and Apple Safari.

LivingSlides has two components: a server (the back-end), which supports LivingSlides by saving the presentations and a slideshow presentation program (the front-end). They communicate with each other by means of an HTTP connection. The server also communicates with the FileSystem to store the files of the HTML code of the presentation. This happens because it is easier for the user, if he wants to use the presentation on another computer he just has to take those files with him, just like he does currently with Powerpoint or Keynote. The front-end also communicates with the FileSystem when it uploads and converts local images and communicates with the web due to online images that may be linked on the presentation.

The layout of LivingSlides was made with the help of Twitter Bootstrap and it is quite simple: it was four sections: the navigation bar, the sidebar, the slide editor and the speaker’s note editor. It has five sub-sections: a dialogue to change the theme, a dialogue to upload images, a dialogue to save a presentation, a dialogue to change the aspect of the slide’s background and the dialogue of interactivity.

Each slide is a `<div>`, this means that it is a block or section of a document. In the beginning every element of the slide was a `<div>` with CSS applied to each of them, according to their function in the slide. But, later on, it was changed to an `<svg>`, but still a slide was delimited by a `<div>`, because when the SVG shapes were added problems were found with the click events. So, to solve this problem, the previous `<div>` elements were transferred into an `<svg>` with the aid of the `<foreignObject>` tag. Every text element in the slides is draggable and resizable.

Figure 3.1: LivingSlides’ initial dialogue

3 LivingSlides
The SVG shapes were implemented with the help of the library Raphael.js, which offers a wide range of SVG properties and extensions, such as the Free Transform which was used to apply the draggable and resizable properties to SVG, which was otherwise impossible with just jQuery. Drawing was made possible by having an SVG element with mouse events bound to it; when it detects a click followed by a drag of the mouse, it is shown to the user the outline of the shape he selected to help him understand the drawing area.

As mentioned previously, users can upload images they have stored in their computers. When an image is upload, it is converted into a base 64 string. This means the image will be part of the file and will always be accessible to anyone who possesses the file, and there is no need for a server and requests to servers.

When a user wants to add interactivity between two elements, he does so by clicking on the green circles placed next to every element of the presentation and the drags the arrow to the green circle of the target element. When the connection is made, a dialogue appears with the text of the source element on the left and the text from the target element on the right. This means that the source element is the independent variable (the variable which will have power over the other variable) and the target element is the dependent variable. Users can select sections of the text to be interactive, by selecting it and then clicking on the button “Set Region”. He then has to specify how the values will change; it can be a linear, a discrete, a logarithmic, exponential or a formula variation. The user must also specify which is the initial value, the minimum and maximum value of the independent variable. The user only needs to specify how the dependent variable will vary, because the limits of it are imposed by the independent variable.

This was all Tangle supported: text and numeric values. To support SVG and images, this library had to be extended. The extensions were made inside the scope of the Tangle function, because it was the only place the name of the variables were accessible. A user can specify the size, position and source of an image and the size, position and the colour of fill and stroke of an SVG shape.

When the user clicks OK on the interaction dialogue, a string is built and then injected into the parent slide of the source and target elements. This has the advantage of when the presentation is saved, the
Tangle code is interwoven into the presentation, and there is no need to send anymore requests to the user or to add anything else to the presentation when it is opened again. An example of injected code can be:

```javascript
var tangle1 = new Tangle(document.getElementById('slideDiv1'), {
    initialize: function () {
        this.Var0 = 1; /* valor inserido pelo utilizador */
    },
    update: function () {
        this.Var0 = (this.Var0 * 1) + 1; /* equação definida pelo utilizador */
        var imgSizeVal = this.Var0 / 15;
        var minWidth = 200; /* valor inserido pelo utilizador */
        var maxWidth = 500; /* valor inserido pelo utilizador */
        var widthResult = interp(minWidth, maxWidth, imgSizeVal);
        $('#imageDiv0').attr('width', widthResult);
        var minHeight = 200; /* valor inserido pelo utilizador */
        var maxHeight = 500; /* valor inserido pelo utilizador */
        var heightResult = interp(minHeight, maxHeight, imgSizeVal);
        $('#imageDiv0').attr('height', heightResult);
    }
});
```

Once the editing is over the user has two options: to save the presentation or to begin the presentation. If he chooses to present it, he clicks on the play button and the editing section expands to make the presentation occupy the whole of the window. Another window appears, which contains the speaker’s notes the user might have added during the editing of the presentation. There is also the previews of the current and the next slide and a timer, which counts the time since the presentation was started.

![Figure 3.3: Presentation Mode](image)

If the user wants to save the presentation, he can do so by clicking on the save button. A dialogue will appear asking the user to choose a name for the presentation. When the user clicks OK, the HTML code of the presentation is then sent to the server to be saved in the FileSystem.
4 Evaluation

LivingSlides was tested with 15 potential users. In this group there were people with ages ranging from 20 to 35 years; 6 people were female and 9 were male. All of them claimed to have worked with either Powerpoint or Keynote. 2 of them said they do presentations "very frequently", 6 people said "frequently" and 7 said "regularly".

Before asking users to perform the tasks, they were asked to play around with LivingSlides for 5 minutes. This was to make them feel comfortable with the tool and also for them to learn on their own.

After that, participants were given 25 tasks. All but 5 were timed, because those 5 tasks asked for user input, which was not the aim of this evaluation.

The conclusions drawn from the evaluation tests are satisfying. Users made none to little errors, and the times of tasks which were interactivity related dropped as the tasks progressed. That means the users learnt how to use the tool, especially the interactivity.

When the tasks were completed, users were asked to answer a System Usability Scale questionnaire.

![Figure 4.1: SUS Questionaire, Questions 1 to 5](image)

As it can be observed 47% of users would like to use this tool frequently. 53% of users think the system is not complex, and 60% think this is an easy to use system. 67% of users who tested LivingSlides say they do not think they would need technical support to use this tool and 47% against 7% who believe the different features of LivingSlides are well integrated.

![Figure 4.2: SUS Questionaire, Questions 6 to 10](image)

When it comes to inconsistency, opinions are divided: 40% disagree with the affirmation "I thought there was too much inconsistency in this system.", but 40 of users have a neutral position. 53% of the users who tested LivingSlides do not have an opinion about how fast the people would learn how to use LivingSlides, but 33% thinks they would do it fast. 64% of users did not think LivingSlides was cumbersome to use and 47% say they felt confident using the tool. Finally, 60% of users disagree they...
had to learn a lot of things in order to use the system.

5 Conclusion

LivingSlides is indeed a tool which brings benefits to people who do presentations. The great challenge of implementing interactivity with every element was concluded with success. Potential users have at their disposition a lot of options to create their "living" presentations.

With this new feature, they can now motivate and help who is watching the presentation to retain and comprehend the information better.

By the research done on Related Work, it is concludable that there is space for a tool like LivingSlides. New ways of delivering and editing presentations are slowly emerging.

These kind of editing tools, which use web technologies, are highly customizable and allow them to be extended, all that is needed from the user is knowledge of the technology and some imagination. LivingSlides also wants to offer users who are not familiar with HTML or Javascript the opportunity to use these tools, by offering them an interface and withdrawing the programming of variables by asking of them only the values for the variables.

There is still some work to be done on LivingSlides. It would be interesting to have more than one-to-one interactive relationship between elements, one-to-many would be far more interesting and even the whole document react to changes of every single independent variable.

LivingSlides hopes to be one of many tool which help to change the way presentations are created.