Report from course ISGC03 April 2011

Anders Brundin

Possibilities on incorporating Adobe Photoshop as a shell builder for Ozlab

Table of content

Anders Brundin
Framtida Webstandarder ISGC03
2011-03-30

1 Introduction

1.1 Purpose

This paper will focus on and show how it will be possible to use graphical programs and incorporate them into interactive programs. If it were possible in prototype situations to import a graphical file directly into test-programs without having to go through converters it would make work significantly easier for testers in their field. The test program this paper will focus on most is Ozlab, a program used and developed by Karlstad University. The graphical program I will focus on is Adobe Photoshop.

1.2 Background

1.2.1Background for Ozlab

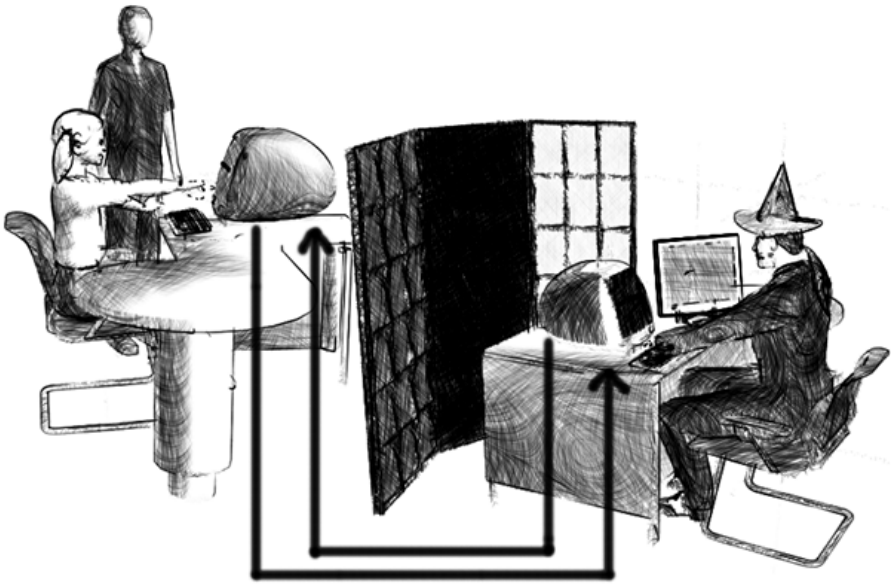
Ozlab encompass two aspects of testing.

Firstly, the facilities are using the Wizard-of-Oz technique. The Wizard-of-Oz technique basically presumes a test environment that disguises the test moderator (TM) from the test person (TP). This is usually done by having 2 adjoining rooms for the tests connected through a one way mirror. Usually the test is conducted with video and audio recording to save the test sessions. The sessions are controlled by TM's behind the one way mirror, They have access to all audio and video recording as the test session progresses helping the TP through the test scenarios.

Secondly, the other aspect of the program is a matter which is vital to Ozlab-based testing. Ozlab is a program that enables the TP to see what the TM shows. All the prototypes that is viewed and "controlled" by the TP is in fact all controlled by the TM, every click and every change in screen image are operated by the test moderator (Fig.1.1). This is so that the prototypes won't have to be programmed, allowing more focus to be on the test and the test results.

> *"The benefit for designers, whether they are professional or inexperienced, of avoiding programming is the ease with which alternative proposals can be produced for discussions with peers and for testing with prospective end-users."* (Pettersson 2002)

Currently the system is based upon Macromedia Director MX and its programming language LINGO. This is not an ideal program to use since it was changed quite drastically since Adobe acquired Macromedia in 2005. The Current Director version from adobe is 11.5 and it can't be used by the Ozlab converter.



[Fig 1.1] Showing the Ozlab concept. Illustration by Anders Karlsson

1.2.2 The Background of Photoshop

Photoshop is one of the most famous programs used to make digital art and photography editing. Since the start of Adobe Photoshop in 1990 the program has been used by millions of users and has become the most efficient tool to use, by everyone from photographers to web designers. Photoshop was originally created by two brothers, Thomas and John Knoll, to be used as a way to convert analog images to digital (Story 1990). It was sold with scanners made by Barneyscan, but then it was called Image Pro. After getting a licensed partnership with Adobe and releasing Adobe Photoshop 1.0 in 1990 the product began to increase in popularity. It was by then still mainly programmed by Thomas Knoll.

Photoshop has evolved drastically since its first release by Adobe. Especially the Layers functionality that was added in Adobe Photoshop 3.0 or the unification with other adobe programs made in 4.0 (West 2010).

Since 2003, Photoshop is a part of the Adobe Creative Suite, and has gone from version Adobe Photoshop CS to CS 5. Creative suite 5 was released in 2010. From Version 1.0 up to CS 4 Thomas Knoll has been the lead designer of Photoshop, but now he works with the Photoshop function camera raw which helps smooth the transition between different camera formats. Figures 1.2 and 1.3 show two different versions of Photoshop.

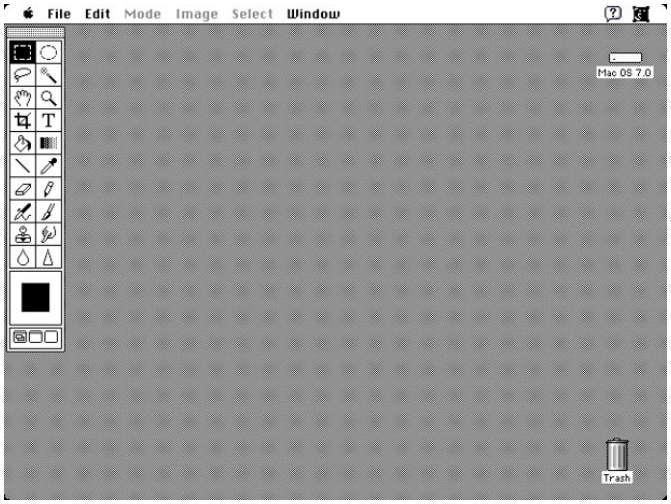


Fig [1.2] Workspace in Adobe Photoshop 1.0.7

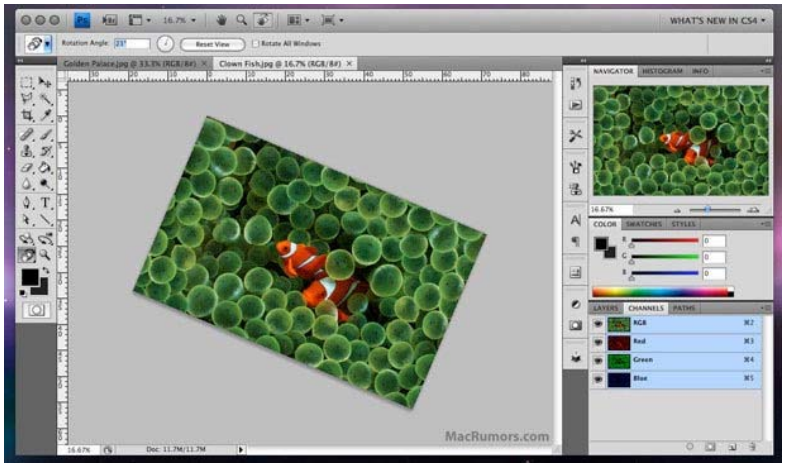


Fig [1.3] Workspace in Adobe Photoshop CS 4

1.3 Delimitations and Definitions

1.3.1 Delimitations

This essay will stay within several boundaries. As concerns Photoshop, I will not go deeply into how Photoshop works when it comes to creating graphics or how to modify the graphics. Even though this is the main idea of Photoshop, I feel that it's irrelevant to the outcome of the paper.

Furthermore, the discussions on Ozlab are limited to how the test-runner and shell-builder works, even though a wider description of Ozlab was given in the background. This wider scope was taken in the background in order to explain why this paper is being written.

1.3.2 Definitions

Test moderator (TM) - A moderator during test sessions.
Test leader (TL) - Controls the responses during Ozlab test sessions.
Test person (TP) - An individual a test is done to.
Low-fidelity prototypes (lo-fi) - Primitive prototype to cover the basics to be tested.
High-fidelity prototypes (hi-fi) - Advanced prototype to test a near-finished product.
Ozlab - Test facilities for conducting Wizard-of-Oz tests.

- **Test runner -** The program that the prototypes are tested in**.**
- **Interaction shell -** The prototype that is used in Ozlab is called interaction shell.
- **Shell-builder -** The part of Ozlab where the shells for the test runner gets built.

Wizard-of-Oz (WOZ) - Test technique where one person (the Wizard, often hidden) executes the responses seen or heard at the test person's computer.
Photoshop Document (PSD) - Specific file format used by Adobe Photoshop
Extended markup language (XML) - File format usable by most interactive programs and databases

2 Method

To gather the information I needed to assess the problems and opportunities of Adobe Photoshop I had to gather information from several different sources. Because the problem revolves around adobe Photoshop and not Ozlab it will be quite difficult to actually gather information through interviews and other related methods. Methods that focuses on gathering information from people that have in some way specialized in the area will be harder, because the specific area of Photoshop this paper dwells deeper into is an area that, as far as I have seen, is yet unexplored. This means that the knowledge that I needed to gather will have to be a foundation so that my analysis can theoretically cover this possible utilization of Photoshop as a main tool for Ozlab.

My theory is based upon a loose literature study of a few books. But mainly this paper will get its information from web pages that associate with Photoshop. As the internet community has been very supportive of Photoshop and its evolution with both add-ons and solutions for unusual problem, it is the optimal forum for the information I need.

3 Theory

In this chapter theory will be presented that shows the importance of usability testing and prototyping in Ozlab. Also it will give a base for describing pros and cons with Adobe Photoshop in the Ozlab environment, which will be presented in Chapter 4.

3.1 User Testing

User testing is essential when assessing how well a product or program will work when it released on the market. Or at least minimizing the flaws of the product. Ruben and Chisnell

(2008) discuss usability tests as a research tool with a lot of uses. It can be used with a large group of participants with a complex design, or it can be small tests with individuals. Both can be used to gather either quantative or qualitative data, both formal and informal. It all depends on what objectives the user tests are determined to meet.

The main reasons for user tests are to establish the design, remove design flaws, eliminate user frustration, and increas profit. There are several benefits of fulfilling each of these reasons; eliminating user frustration for instance will give the consumers a motive to continue to buy the product released by the company. It can also be truly valid since it will give the customers a reason to think that the developer company truly thinks of the customers priorities (Rubin & Chisnell 2008).

3.2 Prototypes

When user tests are conducted it is common that prototypes are involved. There are several kinds of prototypes that differ both in the way they are constructed and in content. Firstly we have low-fidelity (lo-fi) prototypes which are easy and fast to make. An example of lo-fi prototypes is the paper prototype which is a very basic sketch that shows a fast overview of what a program or layout can do. It becomes easy to test this on test persons by just having several sketches and switch them depending on what the TP chooses in menus. It is also usable for just showing a rough draft of a design.

> *"The value of the paper prototype or paper-and-pencil evaluation is that critical information can be collected quickly and inexpensively"*
> (Rubin & Chisnell 2008)

The opposite of the lo-fi prototype is the high-fidelity(hi-fi) prototype. The hi-fi prototypes often show a lot more of the product than the lo-fi prototypes. A hi-fi prototype can be made by having depth in the content of the prototype or by having an almost complete design. It is often used to test whether the graphical design or navigational structure of the product is satisfactory (Cagan 2008).

Ozlab is a mixture of both hi- and lo-fi prototyping. It's possible with the Ozlab setup to create very lo-fi prototypes and test them in a hi-fi test environment as well as creating hi-fi prototypes and test these in a hi-fi environment (Pettersson 2001).

3.4 Ozlab Behaviors

Ozlab is filled with behaviors; they can also be referred to as functionality. This functionality is what makes the different scenes and images truly intractable for both the TP and the TL. There are several different behaviors available listed below in fig 3.1

objektFlyttbartAvTL Makes an object moveable by the TL
objektFlyttbartAvTP Makes an object moveable by the TP
objektFlyttbartAvTLTP Makes an object moveable by both TL and TP
objektGömbartAvTL Makes it possible for the TL to hide an object from the TP
objektKnapp Creates a clickable button out of any object
objektKomIhågPosition Makes is possible for the TL to bring an object back to its original position by clicking the restore button
objektOsynligtFörTP Makes an object permanently invisibly to the TP
objektOsynligtFörTL Makes an object permanently invisibly to the TL
sidmarkör Is used as pause-function in the interaction-shell. This function is made by adding code in the frame script channel in the Score-window. This needs to be placed in the same column as a marker in order to work properly.
spelaUppLjudFörTP Allows the TL to control the playback of an audio-file by clicking on director objects of the type button.
textfältEditerbartAvTL allows the TL to edit an object of the type text-field, can be combined with the function below
textfältEditerbartAvTP allows the TP to edit an object of the type text-field, can be combined with the function above.

[Fig 3.1] Ozlab behaviors. Siponen et al. (2002)

4 Analysis

In this chapter the possible solutions are described. It is worth mentioning that if any of these options are implemented it would undermine several aspects of User Testing and Prototyping described in chapter 3.1 and 3.2.

4.1 The pure Adobe Photoshop solution

Photoshop saves everything about your work, all the layers, folders, channels and file info in one single smart bundle that is the PSD file. If it is possible to generate the information directly from the PSD to Ozlab, the need for programs to meddle between them would disappear. This would eliminate the need for extra programs and users could go directly from Adobe Photoshop to Ozlab. Eliminating all programs except for Photoshop and Ozlab would require less of the users, since the required knowledge of users would consist of user testing, usability and a base knowledge of drawing programs, in particular Adobe Photoshop. Since this would enable more users from various backgrounds to use Ozlab it can be used for several other purposes. Generating fast and cheap prototypes would no longer need an extended experience in Ozlab.

You will still be able to generate both hi- and lo-fi prototypes through Photoshop. The lo-fi prototypes could still just be paper prototypes that are scanned in and perhaps edited. It would allow for flexible and fast tests.

For example, let's say we have a Photoshop file we want to import into Ozlab and use as a prototype. We will at first need to structure the groups and layers in Photoshop so that Ozlab can recognize how we want to present the images. So when we import the PSD into Ozlab we want to create structure based on scenes. All the groups in the PSD will become our scenes by naming them #Scene [scenename]. If we have a group filled with layered images. These layers can have behaviors if they are added in Photoshop. You add behaviors by naming the layers accordingly in Photoshop. If an image with different behaviors and images needs to be added to a scene you can create a group within a group starting with #Image instead of #Scene. This is shown in fig 4.1.

To incorporate Adobe Photoshop in the Ozlab set-up would be difficult. The Photoshop document is viewed as difficult to work with and alter. According to Fuller and Fuller (2007) only two drawing programs have yet to successfully import PSD files in a fulfilling way.

> *"The downside of the Photoshop format is that relatively few applications other than Photoshop support it, and those that do don't always do a great job"*
> (Fuller & Fuller 2007)

The programming needed to make this work would probably be quite intense, which makes it a less viable option. This solution would unfortunately also require a lot of proper naming inside Photoshop making it a less flexible solution.

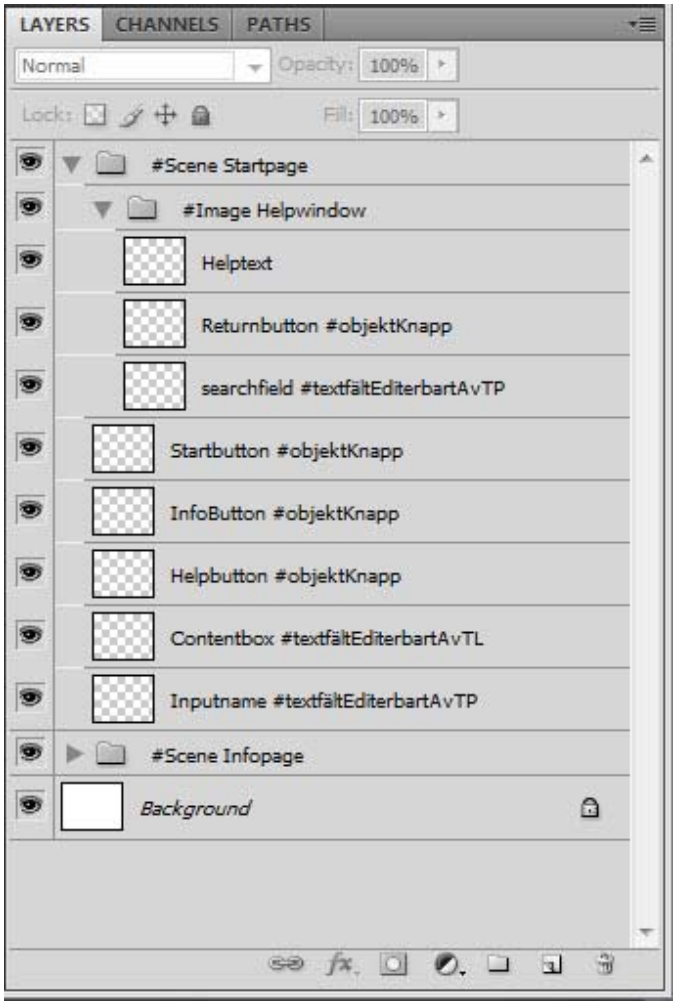


Fig [4.1] Structure of how layers and groups can be structured in Adobe Photoshop

4.2 The XML/CSV Solution

This solution requires some converting between formats, but will make Ozlab more flexible for future development. Export a PSD file from Photoshop. The PSD file will work as a folder that the XML file will fetch layers and information from. Import the PSD file into a shell builder for Ozlab to just get a basic overview of the layers. In the overview you categorize the layers into scenes and add some behaviors to them. From the shell builder you save an Oz-prototype as an XML file that is ready to be imported into Ozlab to be tested.

There is several plug-ins made by users for Adobe Photoshop that can help you save a PSD file in different ways. None that allows you to save as an XML file, although this can probably be programmed. Adobe Photoshop is not really usable with XML, though the solution can be made manually through much labor. This said, XML is a great option. It is easy to learn the XML format and you can easily edit directly in the XML file. Also there are signs that XML as a file format will continue to be used in all sorts of interactive media today. So a solution with XML can be used for an extended period of time, and it can always be updated to include other digital graphics programs if for some reason Adobe Photoshop should be discontinued.

The structure for the XML files that will be imported into Ozlab are shown in fig 4.2. It all expands from the psd tag that includes a path to the PSD file that will be used. In the base psd tag there will be scene tags taken from the psd wich will include image/layer tags with information on how and where the layers are positioned and their size. This will include a behaviors tag that is the added behaviors for each layer.

```
<psd path=”Ozprot.psd”>
  <scene groupname=”startpage”>
    <image layername=”helpwindow” posX=”230” posY=”120” width=”300” height=”200”>
      <behaviours #objektFlyttbartAvTP ></behaviors>
    </image>
    <image layername=”soundbutton” posX=”330” posY=”520” width=”40” height=”40”>
      <behaviours #spelaUppLjudFörTP  #objektKnapp  #objektOsynligtFörTL></behaviors>
    </image>
  </scene>
</psd>
```

Fig [4.2] An example of how the XML structure will be.

5 Conclusion

Most of the current and possible future users of Ozlab have, and most likely would have an extensive knowledge of how to use Adobe Photoshop. So to make the most of this theory, using Adobe Photoshop would be the optimal choice.

Almost all of the prototypes that are used in Ozlab today are first drawn and designed in Photoshop, and later added behaviors in Director. If all the middlemen could be removed and the graphics created in Photoshop could be directly implemented into Ozlab, then the prototypes could be created at a much more rapid pace.

For me as an Ozlab user it would be great to be able to create graphics and directly import these in a flexible way into Ozlab. I have always found Adobe Photoshop to be the program that the majority of the time is spent in, and then the rest trying to figure out how to make the behaviors work just to be able to test my graphical creations.

The upside of using Photoshop in Ozlab is that you eliminate a lot of unnecessary work between the created graphics and the prototype testing, and as previously mentioned it is a widely used program by developers, testers and designers.

There are several downsides to use Photoshop. The huge amount of programming needed to be able to customize Ozlab for Photoshop will be a big obstacle to overcome, no matter what solution that might be implemented. The fact that Ozlab will be bound to another program is bad. A more portable and free solution would be a better solution.

The pure Adobe Photoshop Solution would be better than an XML solution, since the XML solution would go back to having programs in-between Photoshop and Ozlab. But in both cases the users could be quite frustrated by the naming of layers and groups, and the need to always keep tab of the structuring in the psd/xml files. The Prototypes that are designed and tested by the Photoshop/Ozlab combination will be used to find design flaws and user frustration areas. If the prototyping tools themselves were bad in terms of usability and had several design flaws, it would undermine the entire process of creating the prototypes.

6 Referenses

Cagan, M. (2008). *High-Fidelity Prototypes.* [Electronic]. Available: http://www.svproduct.com/high-fidelity-prototypes/ [2011-02-28]

Creativity, 13 (3), 144 - 156

Fuller, L.U. & Fuller, R.C. (2007). *Photoshop CS3 Bible.* Indianapolis: Wiley Publishing, Inc.

Pettersson, J.S. (2001) *Presentation av idéerna bakom Ozlab.* [Electronic]. Available: http://www.is.kau.se/~jsp/ozlab/n.php?goto=Swe/Presentation_av_ideerna_bakom_Ozlab_010701.html [2011-03-24]

Pettersson, J.S. (2002) *Visualising interactive graphics design for testing with users*. Digital
Rubin, J. & Chisnell, D. (2008). *Handbook of Usability Testing - how to plan, design, and conduct effective tests.* Indianapolis: Wiley Publishing, Inc.

Siponen, J. Pettersson, J.S. Alsbjer, C. (2002). *Ozlab Handhavandemanual*. Karlstad: Institutionen för informationsteknologi informatik/Centrum för HumanIT, Karlstads Universitet.

Story, D. (2000). *History of Photoshop.* [Electronic]. Available: http://www.storyphoto.com/multimedia/multimedia_photoshop.html [2011-03-18]

West, A. (2010). *20 years of Adobe Photoshop*. [Electronic]. Available: http://www.webdesignerdepot.com/2010/02/20-years-of-adobe-photoshop/ [2011-03-23]