Fotoscript: A Simple Language For Image Creation and Editing

Matthew Raibert
mjr2101@columbia.edu

Norman Yung
ny2009@columbia.edu

James Kenneth Mooney
jkm2017@columbia.edu

Randall Q Li
rql1@columbia.edu

September 28, 2004

1 Introduction

There are many tools for creating and editing images. Almost none of those tools allows the user to write scripts. Most of them rely on a graphical interface that presents to the user a virtual canvas; she is then expected to pull down some menus and make some gestures until the result seems to look like what she expected. Whereas this is the familiar way to interact with images it is error prone, inexact and not particularly powerful. In this paper we propose to create a simple, robust, relatively familiar language for scripting image manipulations on a computer.

2 Why Not GUI?

It may appear that the logical way to interact with an image is to use a graphical interface, perhaps with a mouse approximating the way a painter uses her brush. This comes from the paradigm of computing which dictates that the way we interact with computers ought to simulate traditional tasks—indeed, what could be more traditional than a painter with a brush. This kind of interface is certainly familiar and intuitive but it puts a number of limitations on what the user can reasonably do. We get the most significant advantage from using a computer by using it for those tasks that we humans are not particularly good at. So by limiting the user to using gestures and estimation—which we can do just fine without a computer—GUI oriented programs necessarily limit the users ability to achieve clear, unambiguous and consistent results. As an alternative, we propose to create a language for the user to specify what she has in mind in a clear and simple way that allows for precision and repetition.

3 What Software Already Exists?

One example of a language for scripting image manipulations is the old LOGO family of languages. These languages allow the user to direct a "turtle" around a canvas and it leaves a trail as it goes. It can detect the color at its current position and use user programmed logic to decide how to react. We will probably take some ideas from it but LOGO was designed primarily to teach children how to program. It focuses on visualizing common programming tasks and is not very powerful as an image editor.

Most image manipulation programs such as MS Paint and Adobe Photoshop focus on the graphical interface. We will be getting ideas for functionality for our language primarily from these programs. Because these programs are familiar to many users, we will try to base our function names on naming conventions used by such programs.

4 What Functionality Will Fotoscript Include?

There should be a set of basic operations such as resize, stretch, rotate, brightness, contrast etc. These are good cantidates for operators, for example division '/' and multiplication '*' might signify adjusting the image size.

Then a second set of operations could be like LOGO: the program maintains a 'sprite' which consists of a point of focus and a heading. The user directs the sprite to move and to draw or perform other operations on the small portion of the image at the point of focus. The user could have a movements in mind and program them specifically, or he could program a slightly smarter sprite that would find places with some set of characteristics and perform its operations at those positions.

A third set of operations could include what are called "filters" in Photoshop. Some examples of these filters might include the ability to overlay one image on another, the ability to change the color of all pixels in a certain color range, or the ability to cause the image to conform to a user defined histogram. The user should also be able to design his own filters by combining the simpler filters and perhaps even using the sprite.

5 What Will Fotoscript Look Like?

Fotoscript will be a concise scripting language with an image object as a basic type. The user will be able to designate any section of an image as itself an image object and perform any operation that can be performed on the whole.

There will be about half a dozen operators for the most basic operations; some possible examples (a and b are both variables of the image type):

- a / 2; //resize the image by a half
- a @ 90; //rotate the image counterclockwise by 90 degrees
- a + b; //concatenate two images vertically
- a — b; //concatenate two images horizontally

6 How Will Fotoscript Be Implemented?

Our focus will be on designing a regular language that simplifies common image manipulation tasks. To that end we have decided to use existing libraries to perform many of those tasks rather than implement the algorithms ourselves. We will write our compiler in C and use the GIMP library for implementations of image manipulation tasks. As an alternate implementation

style we might write our compiler in Java and output C code that has calls to the GIMP API. It could then be translated into machine language by the C compiler.

7 Summary

Fotoscript will increase the productivity of graphic designers by allowing them to better leverage the power of computers. It will certainly not seek to replace the canvas and brush approach to image manipulation, but to complement it. Fotoscript's concise syntax will simplify image manipulation, even when working with a hundred images at once.