CS 465 Homework 1b

out: Monday 8 September 2003
due: Monday 15 September 2003

In this first assignment, you will implement a simple painting program with a subset of the capabilities of a program like Adobe Photoshop. Your program will have two tools:

1. an airbrush-like tool, which allows you to paint on foreground and background layers with an adjustable brush size, brush softness, and color, and
2. the eraser, which removes the paint, leaving the underlying image (if any) showing.

The user will be able to choose the size and softness of the brush, the color of the paint, and the opacity of the paint—that is, how much each pass of the brush obscures the image.

This assignment is to be done individually.

Principle of operation

Your program will be based on compositing operations. You'll have two layers, a foreground layer and a background layer, to paint on. Behind these layers is a fixed checkered background. The framework described below sets these up for you. You should create another image that stores the brush as it appears under the current settings for size and shape, and when the user drags the mouse across the screen, you should composite that brush image into the current paint layer at the mouse position using the **over** operation. What you have after some time, then, is a composite of many images of the brush at different positions.

One purpose of keeping separate paint layers is to allow erasing. When the user selects the erase tool, you should decrease the opacity of the current paint layer by compositing the same brush image using the **out** operator (see Porter & Duff).

The brush itself is defined as an image of constant color with an α that varies as a function of distance from the center. Use the following function to define the brush's α channel:

$$\alpha(r) = \max\left(0, 1 - \frac{r}{R}\right)^p,$$

where r is the distance in pixels from the center of the brush. The exponent p is what the user adjusts to control brush softness, and R is the brush radius.

Framework code

To keep the focus of the assignment on image manipulation rather than on the details of interfacing with the windowing and graphics systems, we are providing you with the class `BrushFmwk` to get you started. You just need to define a class of your own that extends this class and implements the painting functions. Most of the framework code is concerned with building the user interface and responding to changes in all the controls; the parts that are relevant to this assignment are the method `BrushFmwk.display`, which does the compositing of the paint layers over the backdrop (you don't need to change this), and the methods `BrushFmwk.recomputeBrush`, `BrushFmwk.applyBrush`, and `BrushFmwk.applyEraser`, which you need to override in your extension. The framework provides a means for the user to adjust all the parameters that you need to define the brush. It also allows the user to load outside images into the currently selected layer. PNG files with alpha channels will be loaded with the appropriate transparency. Sample images are included with the framework code.

You do not need to change the `BrushFmwk` class to implement the required functionality; you only need to extend it with the `Brush` class. Unless you're doing extra credit, please don't edit the framework.

Requirements

To be specific, here is what your program needs to do:

1. When the "airbrush" mode is selected, composite the brush into the image at every mouse motion event, centered on the mouse cursor, as the user drags across the canvas. The color of the brush should be the color that the user chooses with the color picker included in the framework. The brush should affect a circular region with radius, in pixels, equal to the setting of the Radius slider.

2. When the "erase" mode is selected, erase (using the **out** operation) from a region of the image centered on the mouse cursor as the user drags across the canvas.The brush size and softness for erasing should have the same effects as for painting.

Some test cases for checking your implementation:

1. The center of the brush should completely obscure the background when the opacity is turned all the way up, and only partially obscure it when the opacity is turned part way down. When the opacity is zero painting should have no effect.

2. When the exponent p is set to a low value (near 0.1) the brush should look like a uniform circle. When it is set to a high value (near 3) the brush should be more like a dot surrounded by fuzz. The results of painting should always look smooth, except that the edge of the brush may be aliased when the softness is turned all the way down.

3. The center of the eraser should completely reveal the background when the opacity is turned all the way up, and only partially reveal it when the opacity is turned part way down. When the opacity is zero erasing should have no effect.

Discussion questions

Answer the following questions on paper, to be turned in in class the same day the program is due:

1. Sketch graphs of the brush's α channel along a line through the brush center for $p = 0.5$, 1.0, and 2.0. Pay attention to continuity and zero crossings.

2. The result of painting on the top layer should be the same as painting in the bottom layer. What property of Porter and Duff's **over** operation ensures this?

3. Suppose you start with the top layer completely transparent (this is its initial condition), put down exactly one brush image by clicking without dragging, then do exactly one erase operation by selecting the erase tool and clicking without dragging in exactly the same location.

   (a) Does this erase the paint completely? Explain.

   (b) Ignoring discretization effects due to finite resolution and precision, sketch a graph of the contents of the top layer's α channel after these operations, along a line through the brush center. Pay attention to continuity and zero crossings.

Handing in

You will hand in your source code using CMS (the Course Management System), linked on the CS465 web site. The file you hand in should be a Java archive file (`.jar`) that is like the one handed out containing the framework code, but with `Brush.java` filled in with your solution. Unless you are doing extra credit you should not modify the framework.

Extra credit

If you are enthusiastic about the project, we always encourage you to implement extra features for extra credit, subject to the ground rules described on the course web page. Here are some specific suggestions for extensions that would be worth about 5 extra credit points if implemented well:

- The simplest way to do this project results in brush strokes that turn into a bunch of separate images of the brush if you move the mouse fast. Implement a fix to this, such as always using a contiguous chain of brush centers no matter how far apart the mouse motion events are. However, turning on this feature should not cause the tool to behave very differently—for instance, by substantially increasing the opacity that results from the same mouse motion.
- Many photo editing programs provide a "rubber stamp" or "texture cloning" tool that is remarkably effective for removing scratches, blemishes, distracting background objects, and other unwanted features from photographs. The basic principle is the same as painting, except that the color of the brush is taken from the background image at a fixed offset from the painting location. Implement such a tool so that it can be used to cover up some paint that has been put into the background layer by copying texture from another area.

and, if you are familiar with rotation and scaling in 2D using matrices and vectors:

- Another common feature in paint programs is elliptical brushes, which can make nice calligraphic effects. Implement brush flatness and angle controls so that the brush can be made into an ellipse while still obeying the radius and softness controls. You can do this when you compute the brush image by rotating and scaling the vector from the brush center to the pixel where you are computing the opacity, before computing its length to determine the value of r in the brush profile equation given earlier in this handout.

If you want to implement some other extra feature we recommend talking to us first to make sure we agree that it would be worth extra credit.

Let us re-emphasize that extra credit is only for programs that correctly implement the basic requirements. Extra features must be in addition to the required features and not interfere with their operation. Your program must by default act like a solution to the standard assignment, for example by providing checkboxes to turn any extra features on.