CS 418 Program 1: Paint

(revised February 4, 2003)

out: Wednesday, January 29, 2003
due: Monday, February 10, 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 an image with an adjustable brush size, brush softness, and color, and
2. the eraser, which removes the paint, leaving the original image showing again.

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 background.

Principle of operation

Your program will be based on compositing operations. You'll have two layers stored in separate images, one for the image in the background and one for the paint in the foreground (this part the framework described below does 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 paint layer at the mouse position using the **over** operation. What you have after some time, then, is a layer that is a composite of many images of the brush at different positions, which is then in turn composited over the background.

The purpose of keeping the paint in a separate layer is to allow erasing. When the user selects the erase tool, you should decrease the opacity of the 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. We suggest you use the following function to define the brush:

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

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 `PaintFmwk` 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 `PaintFmwk.display`, which does the compositing of the paint layer over the background (you don't need to change this), and the methods `PaintFmwk.recomputeBrush`, `PaintFmwk.applyBrush`, and `PaintFmwk.applyEraser`, which you need to override in your extension. The framework provide a means for the user to adjust all the parameters that you need to define the brush.

Requirements

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

1. When the "paint" mode is selected, draw the brush into the image, centered on the mouse cursor, as the user drags across the canvas.
   (a) 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.
   (b) The color of the brush should be the color that the user chooses with the color picker included in the framework.
   (c) The brush should affect a circular region with radius, in pixels, equal to the setting of the Radius slider.
   (d) 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.
2. When the "erase" mode is selected, erase some paint from a region of the image centered on the mouse cursor as the user drags across the canvas.

(a) 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.

(b) The brush size and softness for erasing should have the same properties as for painting.

Groups

This assignment is to be done individually. However, because I implied earlier that it would be in pairs and did not clarify this point, you may also turn it in as a pair. If you do work as a pair, you are required to implement one of the two extra credit features below in order to receive full credit (that is, we will treat the extra feature as a required feature for grading).

Extra credit

If you are enthusiastic about the project, we always encourage you to implement extra features for extra credit, subject to the constraint that no extra credit will be given for programs that do not fulfill the basic requirements correctly. Here are some specific suggestions for extensions that would be worth about 5% extra credit 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. Bresenham's algorithm may be helpful for this.
- 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.

  Implementation hints:

  – You need to provide a way for the user to indicate the location where the texture should be copied from (perhaps using a shift-click).
  – When the user begins painting, compute the offset from the source location to the painting location.
  – When you are putting down paint at a particular pixel, use the offset to compute a location in the background image, and take the color from there instead of using the fixed color from the color picker.

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

Let us emphasize again 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.