Intro to FlashPhoto (Iteration #2)

CSCI-3081: Program Design and Development

Iteration 2 Here We Come

FlashPhoto demo from the TAs.

Four Groups of Features

- Image Saving and Loading
- Image Filters
  - 5 Convolution-Based
  - 5 Other
- New Interactive Tools
  - Rubber Stamp
  - Blur
- Undo/Redo

Feature Group 1: Loading and Saving Images

Loading and Saving Images

- Support two different image formats (PNG and JPG)
- Without knowing any more… what can you say about some design goals for this portion of the software?

Group 1: Learning Goal

- You should learn during this process how to integrate an external C++ library into your program.
- This means not only calling the correct functions within the library, but also **adapting your Makefile** to include header files and library files for the library.

Side Note: Student Learning in a Group Work Setting

- Make sure everyone in your group learns this… you all need to know how to adjust a makefile to link with an external library.
- Consider adding a note to your "group expectations document" about making sure that all members of the group understand a solution before moving on.

Feature Group 2: Image Filters

Image Filters

- These are algorithms that you run on image data that change each pixel in the image in some way.
- They are not controlled by an interactive “brush”, instead they are applied to the canvas as a whole.
- This requires a big “for loop” that will loop through each pixel in your canvas and update its color in some interesting way based on the algorithm.

The Simplest Filter: Threshold

- Algorithm: Given a grayscale image, convert the image to black and white.
- Any pixel with a brightness value greater than the threshold 0.5 is turned white, otherwise turn the pixel black.
- Then… adapt this to work separately for the R, G, B channels.

Adjust Saturation

- Saturation is a measure of how vibrant the colors in the image are. A completely non-saturated image would be a grayscale version of the image.
- Algorithm:
    - convert pixel to a grayscale value (e.g., using ColorData::getLuminance())
    - linearly interpolate between the grayscale version of the color and the color
    - interpolate by 0% = grayscale
    - interpolate by 100% = the original color
    - interpolate by 200% = a really vibrant new color

Adjust R,G,B Levels

- Like a saturation filter, but operates on the Red, Green, and/or Blue channel separately.
- Given some adjustment factor from the user between 0.0 and 1.0, simply multiple the R, G, or B component for each pixel in the image by the adjustment factor.

Quantize Filter

- Reduces the number of unique colors in the image by binning similar colors.
- Takes as input a preset number of bins.

Filters So Far…

- So far, all of these image filters can operate “in place”.
- For other types of filters (e.g., blur), the new color for a pixel depends not only upon the original color of that pixel but also upon the original colors of its neighbors.
- In this situation, we need to save a copy of the original colors before modifying the image.
- One way to do this is to use the PixelBuffer class, you can only display one PixelBuffer at a time, but you can store more than one inside your program if you want.

Convolution-Based Filters

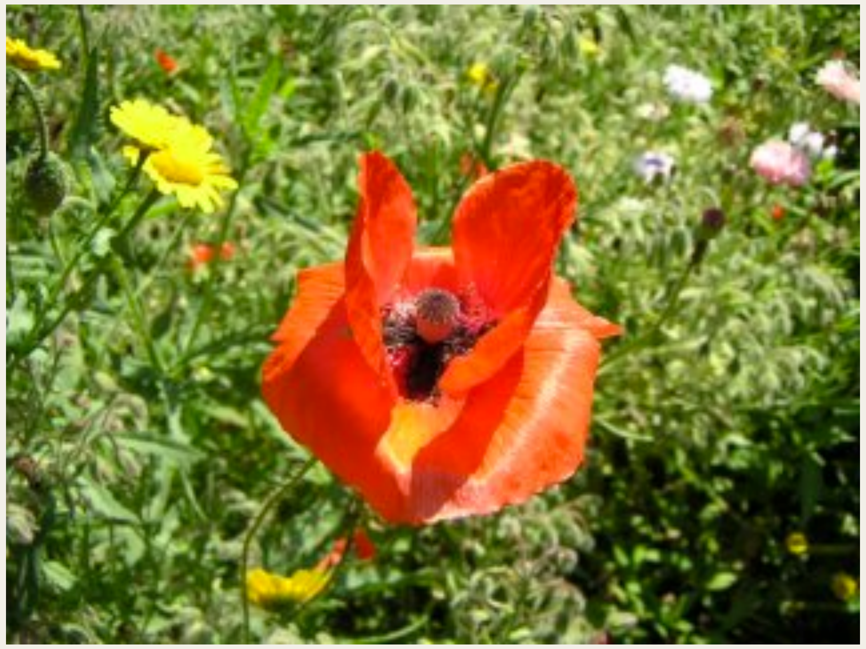
- Let’s learn a bit now about convolution.
- You can create a whole series of very cool filters based on this idea.
- In convolution-based filters, you have an image and you have a convolution kernel.
- The kernel is sort of like a tool mask from iteration #1, but it is not controlled by the mouse.
- Instead, your code moves it algorithmically across each pixel in your image and then *convolves* the kernel with that pixel and its neighbors in order to determine a new color for the pixel.

Examples

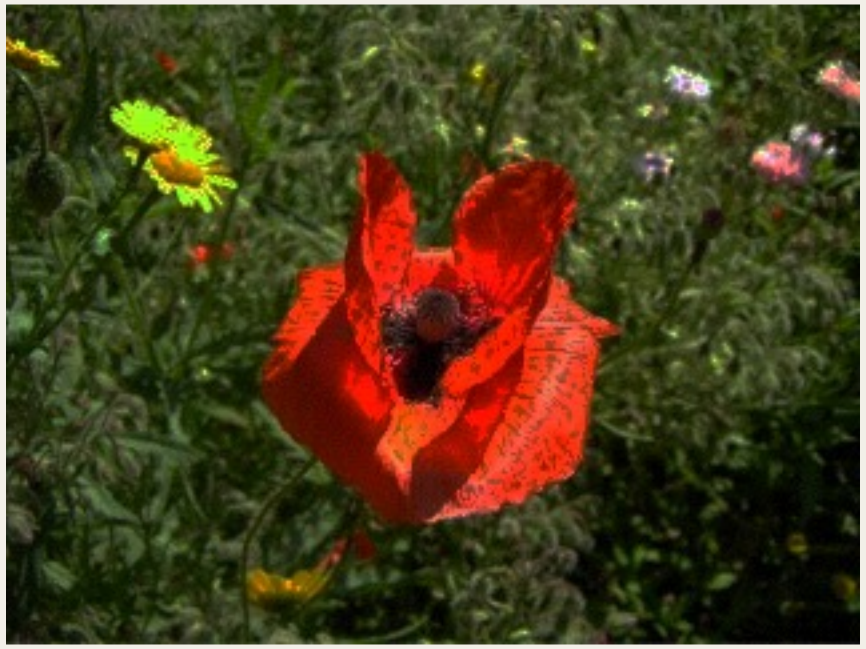
From: http://lodev.org/cgtutor/filtering.html

1. **Blur**
2. **Motion Blur**
3. **Sharpen**
4. **Edge Detection**
5. **Emboss**

What would this kernel do?



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

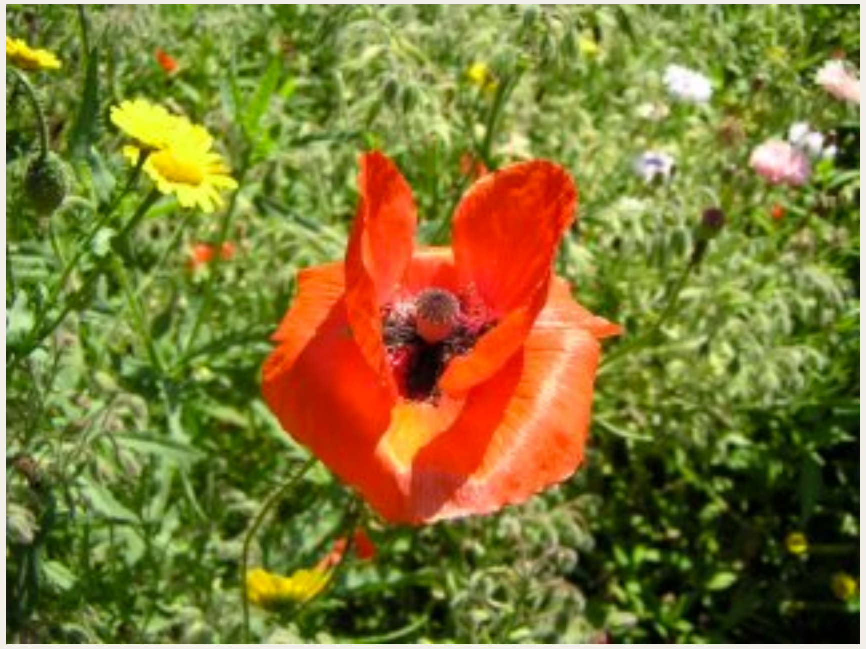


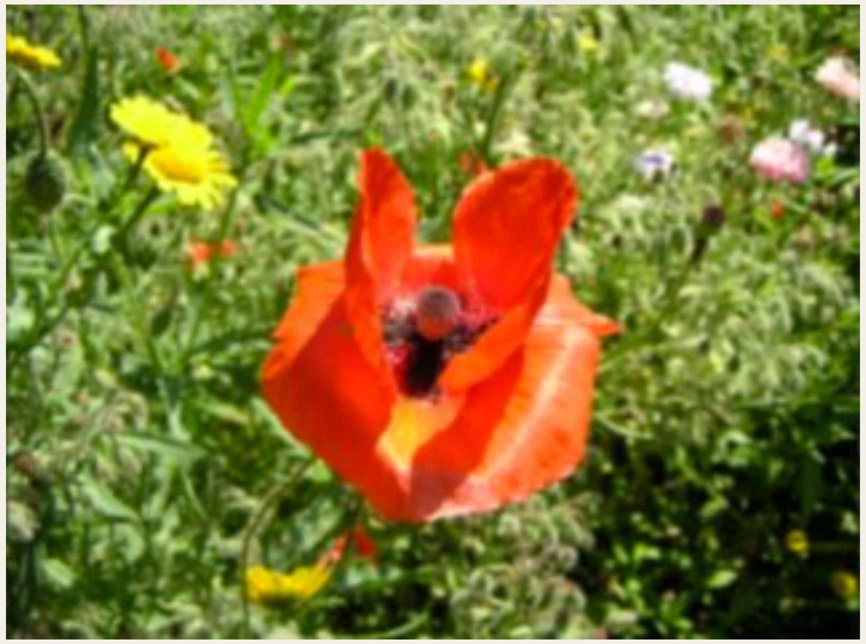
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Properties of the Kernel

- Like a brush mask, make the kernel size an odd number so you can center it around a specific pixel.
- If the values in the kernel add to < 1.0, then the image gets darker.
- If the values add to > 1.0, then the image gets brighter.
- If the values add to 1.0, then the brightness doesn't change — this is what you want.

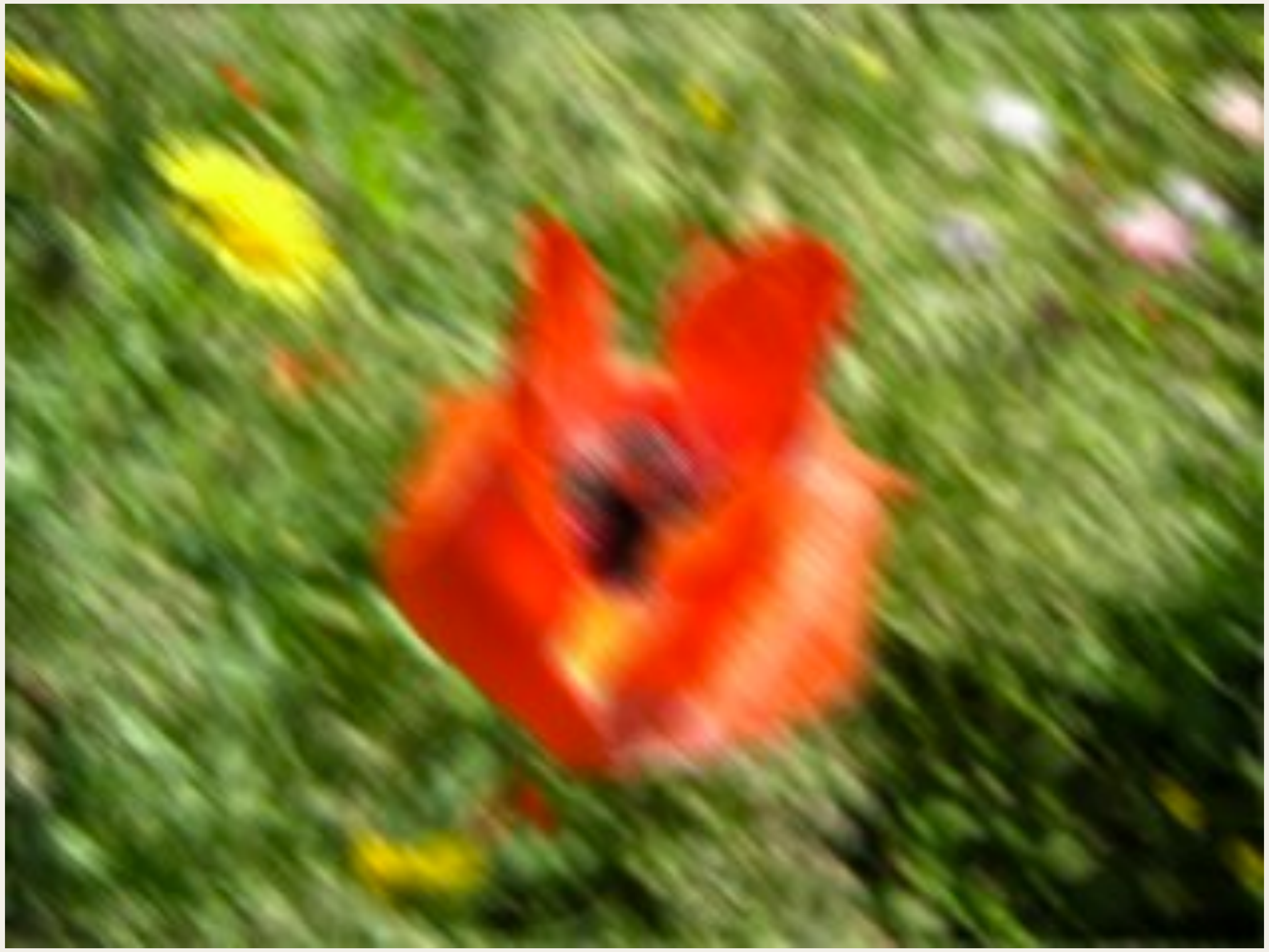
What would this kernel do?





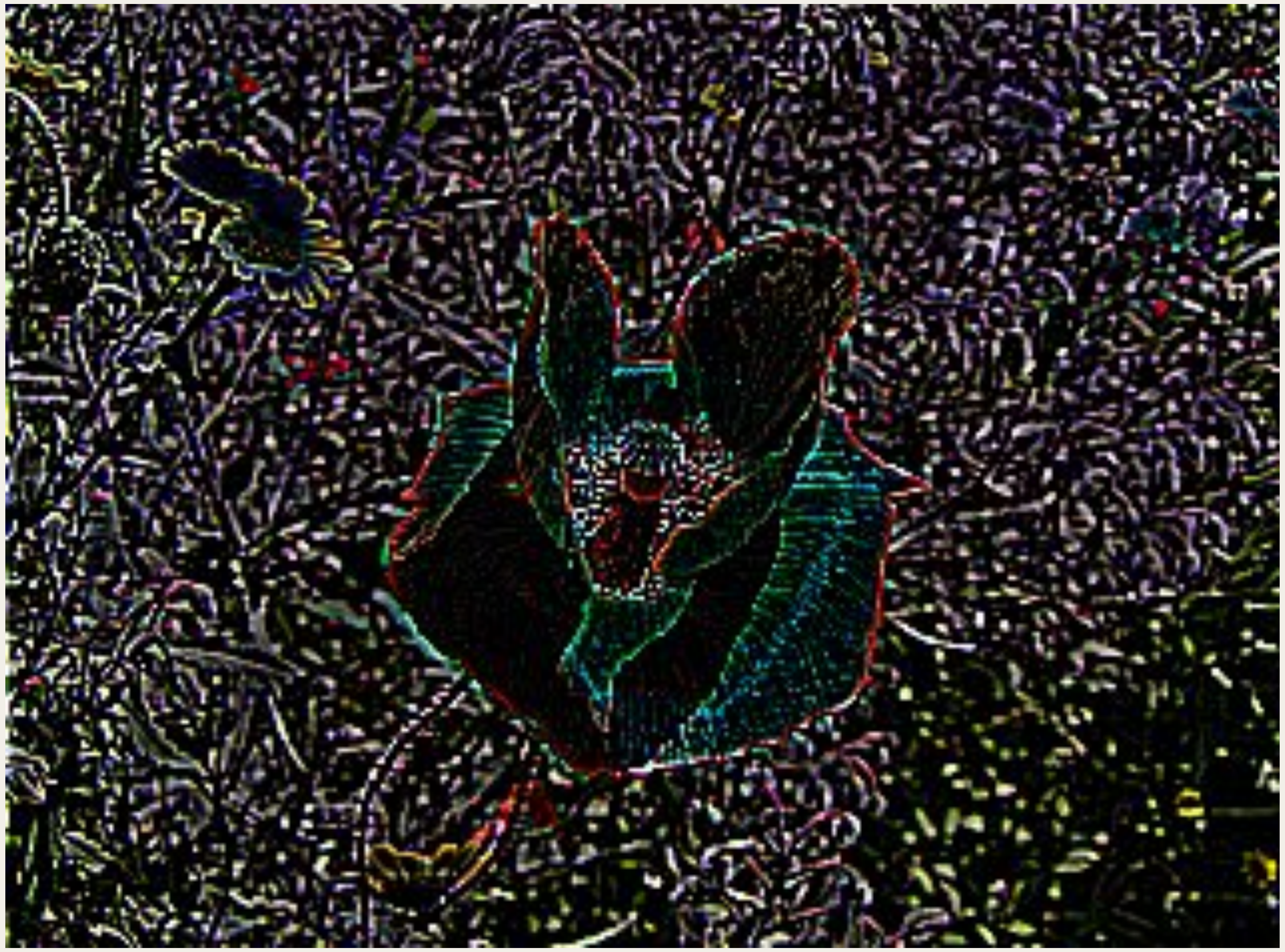
$$\begin{bmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{bmatrix}$$

Motion Blur



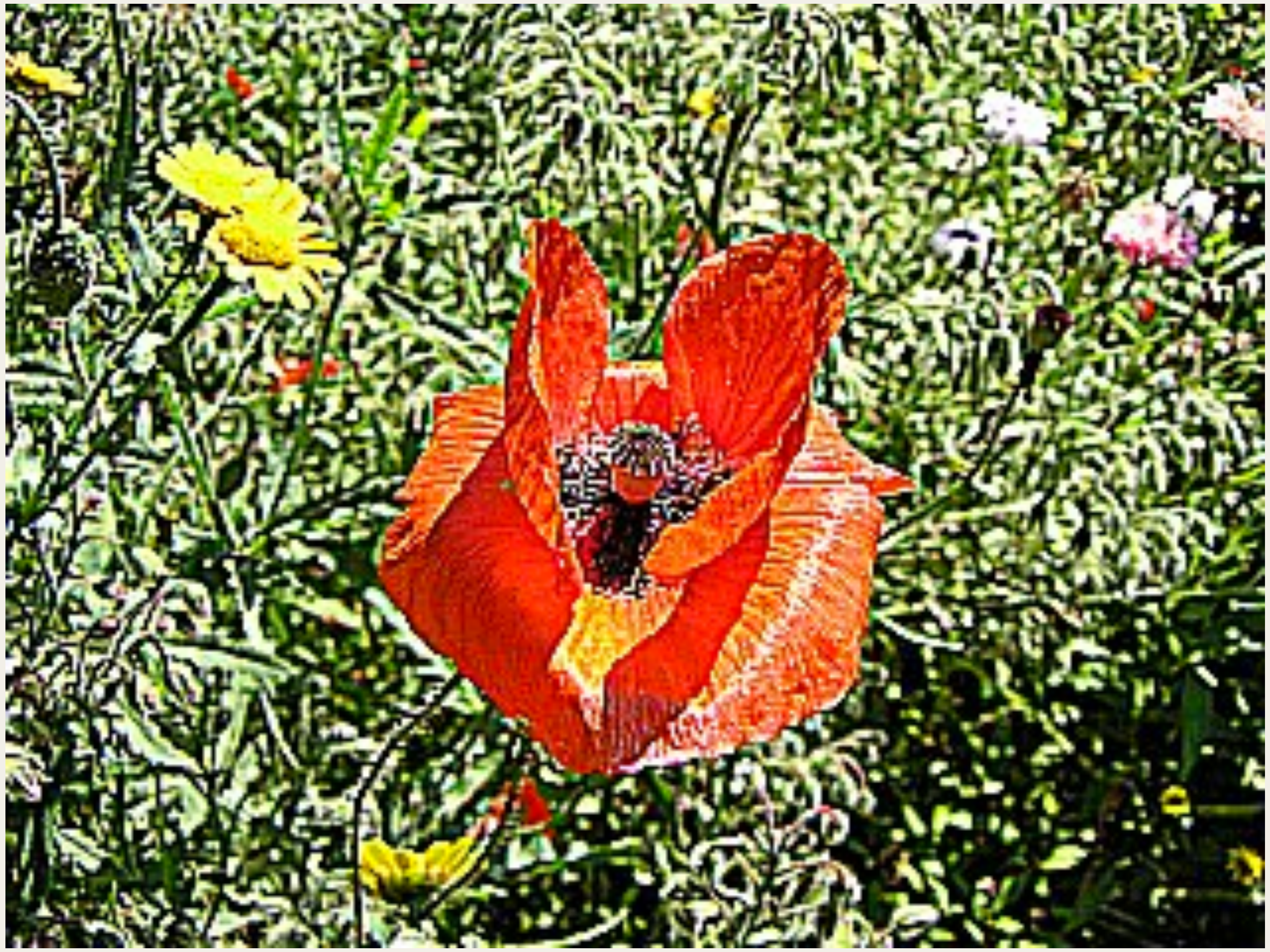
$$\begin{matrix}
1/9, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 1/9, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 1/9, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 1/9, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 1/9, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 1/9, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 1/9, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 1/9, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 1/9
\end{matrix}$$

Detect Edges



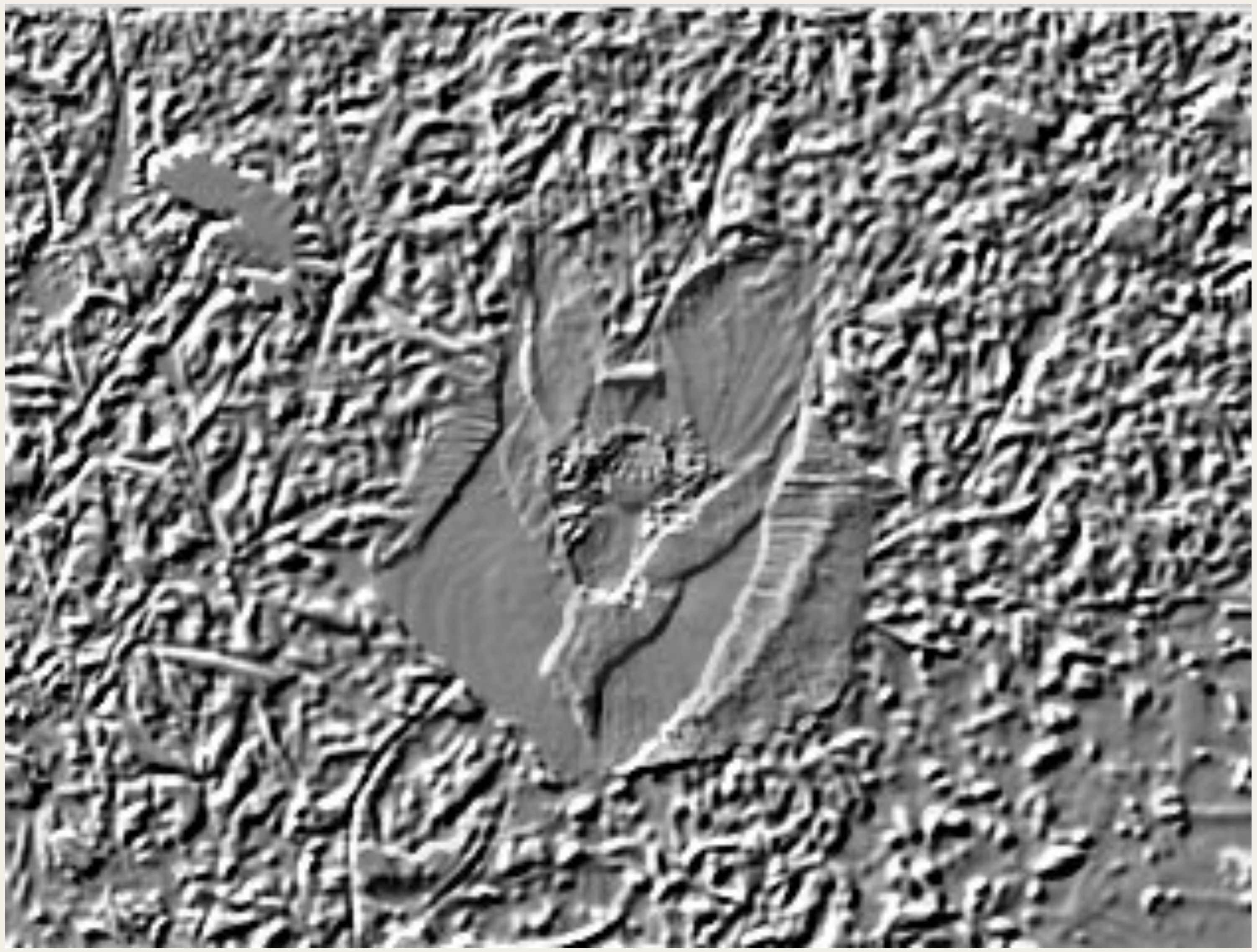
$$\begin{bmatrix} -1, & -1, & -1 \\ -1, & 8, & -1 \\ -1, & -1, & -1 \end{bmatrix}$$

Sharpen (Enhance the Edges)



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Emboss (Create shadows and highlights based on a light direction coming from the top-left)



$$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Feature Group 3: New Interactive Tools

Rubber Stamp Tool

- Takes an image loaded from a file.
- Stamps it anywhere on the canvas.

Blur Tool

- Uses the blur image filter functionality.
- But, applies it only in the local area of the mouse.
- The “amount of blur” should “fade out” in intensity from the most blurry in the center to least blurry at the edges.

Feature Group 4: Undo/Redo

What should we consider to be a “command”?

- When implementing undo and redo the place to start is defining the set of commands that can be undone.
- What belongs in that set for us?

Be careful with “Redo”

- When is a “redo” operation valid?

Group Handins

Schedule of Handins

Mon 3/29, 11:55pm	Revision of Team Policies and Expectations, most importantly a schedule and plan for Iteration #2.
Mon 4/11, 11:55pm	Your FlashPhoto Program, which includes: 1. The source code for your program, 2. Your Group Design Document.