CS 4411 **Computer Graphics**
Spring Semester 2022

Project 1 : Impressionist

Assigned : Feb 5
Due : Feb 26

Quick Links

1. Sample Solution [executable]
2. VS19 Skeleton
3. B&W Checklist
4. Possible bitmap fixes (updated: 2/8/2014)
5. FLTK Impressionist Tutorial Document
6. HelpSession: Fltk
7. HelpSession: OpenGL
8. HelpSession: Image processing
9. Impressionist Architecture Diagram
10. Roadmap for Project
11. Impressionist FAQ
12. Tool Kit Resources (FLTK, OpenGL)
13. Use Google's image search capability to find more images.
14. When downloading images from the sources above, remember to save them in BMP format!

Project Description

Impressionist is an interactive program that creates pictures that look like impressionistic paintings. It is based on a paper and a program by Paul Haeberli. Here is the web "Paint by Numbers".

To create an impressionistic picture, the user loads an existing image and paints a seqence of "brush strokes" onto a blank pixel canvas. These brush strokes pick up color from the original image, giving the look of a painting.

Project Objective

You will add the functionality to a skeleton version of the Impressionist program, which we will provide. The purpose of this project is to give you experience working with image manipulation, OpenGL primitives, user-interface design, and image processing.

Getting Started

To get going, you need to download the Impressionist skeleton code. Click here to retrieve a zipped copy of the VS19 skeleton program source code. Open the VS project to build and run the program

from within Developer Studio. Fltk is included with the project, so you will not need to download and setup fltk yourselves.

Explanation of the Skeleton Program

The skeleton program we provide does very little. It allows you to load the original image (which must be a 24-bit uncompressed BMP file), and save the painted version. Brush selection is done via a drop down list on a separate window called up via the "File" menu. There is one brush implemented (points) and a slider for controlling the brush size.

You can find some sample input images in the BMP format under the "images" directory that comes with the skeleton program.

Required Extensions

You must add the following features to the Impressionist program:

1. Implement 5 different brush types: single line, scattered lines, scattered points, (filled) circles, and scattered (filled) circles. See the sample solution for an example of each brush's appearance. Note that scattered brushes should sample from each location they color individually, not just use a single color for each splotch.
2. Add sliders to control various brush attributes. You need to include sliders for the line thickness and brush angle, in addition to the existing brush size slider.
3. Add the ability to control the brush direction. The stroke direction should be controlled four different ways: using a slider value, using the right mouse button to drag out a direction line, using the direction of the cursor movement, and using directions that are perpendicular to the gradient of the image. You can use a radio box to allow the user to select which method to use.
4. Allow the user to change the opacity (alpha value) of the brush stroke. An alpha value slider should be added to the controls window.

To see what these features should look like when they're done, you can look at the sample solution (with some of the extra credit) here. Your implementations of brush strokes, brush direction controls, etc. do not have to behave *exactly* the same as the sample solution, but they should be fairly close. Also, note that future updates to the sample solution will be available right here that display some of the bell and whistle implementations.

Project Artifact

When you are done with this project, you will create a project "artifact" to show off the features of your program. For the Impressionist artifact, you will create an impressionistic painting from an image of your choice. We will then create a gallery of all the paintings on the course web page. You will then vote on your favorite artifacts!!

Bells and Whistles

Here is a list of suggestions for extending the program. You are encouraged to come up with your own extensions. We're always interested in seeing new, unanticipated ways to use this program!


To give your paintings more variety, add some additional brush types to the program. These brush strokes should be *substantially different* from those you are required to implement. You will get one whistle for each new brush (within reason).


The skeleton program allows the user to paint outside the boundary of the paint rectangle, then erases this region when the stroke is completed. Change this to clip brush strokes to the region as they're being painted.


When using your program, you currently can't see what part of the original image you're painting. Extend the program so that when the cursor is in the painting window, a marker appears on the original image showing where you're painting.


Sometimes it is useful to use the contents of the painting window as the original image. Add a control to swap the contents of the painting window and the contents of the original image window.


Add controls that allow you to manipulate the color of the image. For example, you could implement independent scaling of the red, green, and blue channels.


Design a brush that selectively applies one or more filters such as blurring and sharpening.


Add an undo feature with at least one level of undo so that you can try a brush and decide to undo its effect on the canvas. This comes in very handy for experimenting with brush and filtering effects.


Add the ability to dissolve one image into another.


A different solution to the problem of not being able to see where you're painting is to show a dimmed version of the painting on the canvas. Add a slider that allows the user to fade in or fade out the original image beneath the user's brush strokes on the canvas. (Beware, this bell and whistle is more difficult than it looks).


Add a "mural" effect to your Impressionist by implementing the ability to load in different images while preserving what has been drawn on the canvas. Add a "New Mural Image" or "Change Mural Image" to the controls window that allows the user to change images. The user may then load an image, draw in what he / she prefers on the canvas, and then load a different image and continue drawing on the canvas; thus, a "mural" effect.


To make your painting more interesting, add "alpha-mapped" brush strokes. In other words, allow the user to load a bitmap representing a brush stroke. This bitmap would contain an alpha value at each position. Then when this brush is used to draw, a single color would be selected from the image, all pixels in the brush bitmap would be set to this RGB color (without changing the alpha value), and this partially transparent bitmap would be painted on the canvas. A new color would be used each time the brush is drawn.


It can be time-consuming to paint an image manually. Add a feature so that a whole painting can be created *automatically*. The user should only have to specify a brush type, size, and angle to use. Then the program should automatically paint brush strokes over the entire image, using a randomized brush order and varying the brush attributes slightly as it goes (to increase realism).


At times, you may want the brush strokes to follow the gradient of a *different* image than the base image. Add a button(s) that will cause the direction of brush strokes to be automatically determined from a user specified image.


The "accuracy" of the painting can be also be improved by clipping long brush strokes to edges in the image. Allow the user to load a black-and-white image that represents the edges in the picture. Then add a checkbox so that the user can turn on edge-clipping, which will automatically clip brush strokes at edges in the image.


Construct a filter kernel design interface so that you can enter the weights of filters of arbitrary sizes. Provide an "apply" button that will cause the convolution to happen. Include a "normalize" checkbox that will automatically divide by the sum of the weights when the user wishes it.


Use the image processing techniques described in class to *automatically* find the edges in the base image. Once you have found the edges, add a button to the user interface that will allow the user to select whether or not the brush strokes should be clipped to the edges in the picture.


Implement a multiresolution automatic painting technique. See Painterly Styles for Expressive Rendering.


Design a brush that can be used to stretch and pull the image as if it were rubber. See Warp George Bush.


Implement a curved brush that follows the image gradient. See Painterly Styles for Expressive Rendering.


Given a source image, construct a new image that is really a mosaic of small (thumbnail) images. To do this, you need to partition the original into tiles and find new thumbnails that are reasonable matches to the tiles. Then draw the new image by substituting the thumbnails for the tiles. See, for example, Adam Finkelsteins Web Gothic. Credit will vary depending on the success of your method.


Extend the Impressionist program to work with video. The user should be able to load a series of images from a video and set up some initial parameters,

and the program should *automatically* generate an impressionistic version of the video. The video should exhibit temporal coherency.

 Implement Bayesian matting.

---

design by Ian Li, maintenance by Chi Keung Tang.

CS 4411 Computer Graphics
Spring Semester 2022

Last modified: Thu Feb 2 17:31:48 HKT 2022