Project #1
Seam Carving
Out: Fri, Jan 19
In: Fri, Feb 12, 7:00 PM

1 Installing, Handing In, Demos, and Location of Documentation

1. To install, type `cs016_install seamcarve` into a shell in the directory in which you want the project to be installed.

2. To hand in your project, go to the directory you wish to hand in, and type `cs016_handin seamcarve` into a shell. Before using this script, make sure that you are in the correct directory! It will hand in the entire current directory, and we don’t really want all of your files—just the ones pertinent to this project. (So be sure to save your README and any tests or other files you need to hand in in the same directory as your code.)

3. To run a demo of this project, type `cs016_runDemo seamcarve` into a shell.

4. See the Seam Carving documentation link on the assignments page for TA support code docs.

2 Introduction

2.1 What You’ll Do

In this project, you will implement the seam carving algorithm for image resizing that you learned on the first day of class. You will write the code for pixel importance calculation as well as lowest cost seam finding.

2.2 Purpose

This project serves as a (sort of) gentle warm-up to get you living and breathing Java again after the long break. Also, you will have the chance to practice writing a dynamic programming algorithm. Dynamic programming is a non-trivial and important technique which you will be very familiar with after completing your seam carver.

3 Overview of Your Tasks

Your task is to fill in three methods in your `MyPicturePanel` class:
• calcImportances()
• calcSeamVals()
• findNextSeam()

Stencil code for MyPicturePanel is provided.

3.1 Further Specifications

• The seams that your program generates do not need to exactly match the demo but they should be reasonable lowest cost seams. If your seam carver destroys obviously “important” parts of the image that the demo does not, you will lose points.

• You must use dynamic programming to find the lowest cost seam. In other words, your code should not have running time > O(w * h) where w is the picture width and h is the height. If you choose to find the lowest cost seam using brute force your seam carver will be unreasonably slow.

• You are only asked to implement vertical seam carving. We don’t ask you to do both, as there is no algorithmic difference between vertical and horizontal seam carving.

3.2 README

You’re required to hand in a README that documents any notable design choices or known bugs in your program. Remember that clear, detailed, and concise READMEs make your TAs happier when it counts (right before grading your project.)

4 Reading

With any assignment in this course, it’s very important that you fully understand the algorithms and/or data structures that you will have to write before you begin coding. The following resources will help if you feel uncomfortable with seam carving or dynamic programming.

• The slides from the first lecture (available on the website) are your best resource.

• If you have no idea what seam carving is, read the wikipedia article.

• Check out the video: http://www.youtube.com/watch?v=c-SSu3tJ3ns
5 Visualizer

The visualizer consists of 3 images and a slider. The top left image is the original image that was loaded into your MyPicturePanel, the top right image shows the seams that have been carved from the image. The color of the seams changes from white to red to black as more and more seams are carved. The bottom image is the result of removing the seams from the image and “squishing” the remaining pixels into a thinner image.

The slider simply picks how many seams the user would like to carve from the image. It ranges from 0 to the image width - 1. The visualizer takes care of both forwards and backwards seam carving for you, so all you need to do is tell it what seams to carve. To do this you must call setSeam(int[]). If you do not call this method, the visualizer will not know what seam to carve so it will carve no seams.

6 Your Code

For this project, your code consists of the following three methods of MyPicturePanel:

- **void calcImportances()**
  - Calculates importance values for every pixel in the image based on the contrast between a pixel and its neighbors.
  - Pixels that are very different from their neighbors should have high importances and pixels that are similar in color to their neighbors should have low importances.
  - It might be a good idea to use setImportance(int x, int y, int val) in this method.

- **void calcSeamVals()**
  - Assigns lowest costs and “from where”s to each SeamElement in the 2D array of SeamElements that corresponds to the current image.
  - This is the dynamic programming method. It should leave you with a filled in SeamElement array that will be used to find the lowest cost seam.
  - You should probably call setSeamElementCost(int x, int y, int cost) and setSeamElementFromWhere(int x, int y, int cost) in this method.

- **void findNextSeam()**
  - Determines the absolute lowest cost seam by referring to the filled in SeamElement array.
  - There are two steps to finding the lowest cost seam. First, find the pixel where it ends by looking at the costs associated with SeamElements in the bottom row of the image. Second, follow the seam up through the image using the “from where” field in each SeamElement.
This method **must** call `setSeam(int[] seam)`. See documentation for how seams are represented.

Your specific task is to take the current image and tell the visualizer the seam which should be removed. Because the image is constantly changing width, you should be sure to use the `getPicWidth()` method for setting your loop bounds. If you just use the original image width every time, you will end up accessing pixels and finding seams that may not be on the image anymore.

7 On `java.awt.Color`

To calculate pixel importances, you'll need to quantify how “different” pixels are from their neighboring pixels. More different pixels should have higher importance values than more similar pixels. It's up to you to choose how to do this, but we'd like to point out that the `java.awt.Color` class contains three methods for `getRed()`, `getGreen()`, and `getBlue()`. Each of these methods returns an integer. Your importance values should probably take into account all three of these values in some way. Hint: RGB color differences can be positive or negative, but for calculating importance, it is the magnitude that matters.

8 Compiling and Running

To compile your program, type `make` into a shell. To compile and run your code, simply type `make run` instead.

9 Testing

Testing your seam carver is very simple. If it works and produces results like the demo, then it works. We'll provide you with several pictures to test on but you can use any picture you want.

10 Using Eclipse

If you would like to use eclipse, you may certainly do so. In order to set up your project and make eclipse work with the Seam Carving support code, you'll need to do the following:

- When launching eclipse, set your workspace to `/home/<your-login>/course/cs016`
- Next, select `File->New->Java Project`
  - Enter “seamcarve” for the project name.
  - Choose “Create project from existing source.” In the “Directory” box, enter `/home/<your-login>/course/cs016/seamcarve`
– Click “next”
– Under the “libraries” tab choose “Add External JARs…”
– Select /course/cs016/lib/seamcarve.jar
– Click “Finish.”

• Now you can run your program by pressing the green “play” button at the top of your screen and selecting “Java application” when prompted. You’ll of course need to fill in your code before it will do anything useful.

11 What to Hand In

1. A filled in MyPicturePanel class

2. README

12 Support Code

The following are methods of MyPicturePanel that are inherited from PicturePanel which you will need to use:

• int getImportance(int x, int y) Returns the importance value of pixel (x, y)
• int getOrigWidth() Returns the original picture width in pixels.
• int getPicHeight() Returns the current picture height in pixels.
• int getPicWidth() Returns the current picture width in pixels.
• java.awt.Color getPixelColor(int x, int y) Returns the color of the pixel (x, y)
• int getSeamElementCost(int x, int y) Returns the cost of the seam element corresponding to pixel (x, y)
• int getSeamElementFromWhere(int x, int y) Returns the “from where” of the seam element corresponding to pixel (x, y)
• void setImportance(int x, int y, int imp) Sets the importance value of the pixel (x, y)
• void setSeam(int[] seam) This very important method tells the GUI which pixels to remove from the image.
• void setSeamElementCost(int x, int y, int cost) Sets the cost of the seam element corresponding to pixel (x, y)
• void setSeamElementFromWhere(int x, int y, int from) Sets the “from where” of the seam element corresponding to pixel (x, y)