Project #1
Seamcarve

Out: Tuesday, February 4
In: Thursday, February 20, 11:59 PM

1 Installing, Handing In, Demos

1. To install, run `cs0160_install seamcarve` in your terminal.

2. To hand in your project, go to the directory you wish to hand in, which should be
   `/home/<your login>/course/cs0160/seamcarve`, and run `cs0160_handin seamcarve`
   into a terminal.

3. To run a demo of this project, run `cs0160_runDemo seamcarve`.

2 Introduction

2.1 Silly Premise

Tokyo 2020 is coming QUICK, and as a lead reporter for the Brown Daily Herald, it is your job to get the whole world hyped about the upcoming Olympic trials! You have an article due in 3 hours with a profile on the 2016 gymnasts who will be trying for another 2020 run. Sadly, the photo has to be in a very specific format, so you need a new picture for the front page. You’ll have to figure out how to best calculate the least important seams in order to shrink the image and fit the size restrictions of the newspaper — and get it all done before printing starts!

2.2 What You’ll Do

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

3 Overview of Your Tasks

Your task is to fill in the `findLowestCostSeam()` method in the `MyPicturePane` class. The seam carving algorithm is not trivial so you’ll want to use your program design skills to simplify your code as much as possible with helper methods. How you implement the algorithm is up to you.
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’ll lose points.

- You must use dynamic programming to find the lowest cost seam. In other words, your code should not have a running time greater than $O(w \times h)$ where $w$ is the picture width and $h$ is the height in pixels. If you choose to find the lowest cost seam using brute force, your seam carver will be unreasonably slow and you’ll lose points.

- You’re only asked to implement vertical seam carving. We do not ask you to do both, as there is no algorithmic difference between vertical and horizontal seam carving.

4 README

You’re required to hand in a README text file (must be named README.txt) 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). Your README should be saved in the same directory as your Java files. Please refer to the README Guide in the Docs section of the CS16 Website (link).

5 Reading

As with any assignment in this course, it is very important that you fully understand the algorithms and/or data structures that you’ll have to write before you begin coding. The following resources will help you get a better grasp on seam carving or dynamic programming.

- Slides and docs on the website are your best resource.

- The original paper on seam carving is available in the course directory: /course/cs0160/lib/seamcarving_original_paper.pdf

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

6 Visualizer

The visualizer consists of 3 images and a slider. The top left image is the original image that was loaded into your MyPicturePane, 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. For the slider to work, you need to fill in the `findLowestCostSeam()` method to have it return a seam. See the next section for what a ‘seam’ is (i.e. how it is represented in code).

### 7 Your Code

For this project, you’ll only be required to implement the `findLowestCostSeam()` method. You’re allowed to write any number of helper methods or extra classes that you want. Below is an overview of what you need to do:

- Calculate “importance” values for each pixel in the image.
  - Take a look at the Support Code section of this handout to see how to get the color of a pixel.
  - 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.
  - You can store your pixel importances in a 2D array that corresponds to the pixels of the image.

- Compute the lowest cost of each vertical seam. This is the dynamic programming step of the algorithm.
  - This is the trickiest part of the project. Remember that when performing the seam carving algorithm we need to keep track of two things for every pixel. The first is the lowest cost of a seam from the bottom of the image to this pixel. The second is which direction that seam came from (the pixel directly below, below and to the left, or below and to the right).
  - Again, you should probably use an array (or two) that corresponds to the pixels of the image.

- Find the lowest cost seam and return it.
  - Using the costs and directions you stored, you need to determine and return the seam itself.
  - There are two steps to finding the lowest cost seam. First, find the top of the seam by looking at the costs associated with the top row of the image. Second, follow the seam down through the image using the directions that you stored.
  - The seam that you return is represented by an array of ints. The size of this array is the height of the image. Each index of the seam array corresponds to one row of the image. The data at each index should be the column index of the seam in this row.
For example, given the below “image” where ‘s’ is a seam pixel and ‘-’ is a non-seam pixel:

- s - -
- s - -
- s - -
- - s -

The following code will properly return a seam:

```java
int[] currSeam = new int[4];
currSeam[0] = 1;
currSeam[1] = 0;
currSeam[2] = 1;
currSeam[3] = 2;
return currSeam;
```

### 8 Support Code

The following are methods of MyPicturePane inherited from PicturePane that you’ll need to use:

- `int getPicHeight()` returns the current picture height in pixels.
- `int getPicWidth()` returns the current picture width in pixels.
- `javafx.scene.paint.Color getPixelColor(int row, int col)` returns the color of the pixel (row, col).

Note: We use zero-based indexing, so the pixel in the upper-left corner of the image is located at (0, 0) and the pixel in the lower-right corner of the image is located at (getPicHeight() - 1, getPicWidth() - 1).

Another note: Because the image is constantly changing width, you should be sure to use the `getPicWidth()` and `getPicHeight()` methods for setting your loop bounds and initializing your arrays.

### 9 On javafx.scene.paint.Color

In the support code we’ve provided you three methods: `getColorRed()`, `getColorGreen()`, and `getColorBlue()`. Each of these methods takes in a javafx Color and returns an integer between 0 and 255 representing either the Red, Green, or Blue value of the color taken in. 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.
10 Testing

The best way to test your code is to compare your results to the demo. Remember that your results do not need to be identical but should be similar (less important parts of the image are removed first). For help with debugging, we have put extra images into the course directory. You can also use any picture you want. To load a new image, just click File and pick your image from within the seamcarve application.

The images we’re providing are located in /course/cs0160/lib/seamcarve-images/.

11 Compiling and Running

To compile your program from the terminal, type `make` from the directory that your code is in. To compile and run your code, type `make run` instead. Make sure that your `Makefile` is in the same directory as your code. If you’re using Eclipse, see section “Using Eclipse”.

12 Using Eclipse

If you’d like to use eclipse, you may certainly do so. However, please make sure that you’re using Eclipse Photon. You can launch Eclipse Photon by running `eclipse &`. In order to set up your project and make eclipse work with the seamcarve support code, do the following after you have run the install script:

- **Open Eclipse and select File->New->Java Project**
  - Enter “seamcarve” for the project name.
  - Uncheck the “Use default location” checkbox
  - Hit the “Browse” button, navigate to your seamcarve folder, and click OK.
  - Click “next”
  - Under the “libraries” tab choose “Add External JARs...”
  - Select /course/cs0160/lib/cs0160.jar
  - Select /course/cs0160/lib/nds4/nds4.jar
  - Select /course/cs0160/lib/junit-4.12.jar
  - Select /course/cs0160/lib/hamcrest-core-1.3.jar
  - Click “Finish.”
  - If it isn’t already made for you, use File->New->Source Folder to create a new source folder in your new project named “src”.
  - Use File->New->Package to create a new package in your new source folder named “seamcarve” and move all the stencil java files into this package. Ignore any errors.
• Right-click on App.java and select Run As → Java Application. Now you can run your program by pressing the green “play” button at the top of your screen and selecting “Java application” if prompted.

• To configure your Eclipse projects to run over FastX or SSH, follow these setup steps
  – Right click on the package icon next to the project name. Go to properities.
  – Go to Run/Debug Settings, select the main window App and click Edit.
  – Go to the arguments tab and, and enter -Dprism.order=sw in the VM arguments block
  – Hit Apply and OK
  – You should be all set to work on this project remotely with Eclipse. Make sure to do this for each new project.

13 Working Locally

In order to do seamcarve on your own computer, you’ll have to copy the .jar files listed above from the department file system into your local directory. If you would like to set up a system for working locally, we highly recommend you use SSHFS. Follow instructions in this document to set up SSHFS or the standard department file transfer systems. Then, follow the same Eclipse set-up instructions as above.

You’ll also have to adjust the path to the image files. First, copy the above images (or your own images!) into a local directory. Then, on line 37 in App.java, replace

```
String defaultFilename = "/course/cs0160/lib/seamcarve-images/teamUSA.jpg";
```

with

```
String defaultFilename = "<YourDirectory>/seamcarve-images/teamUSA.jpg";
```

Note that you’ll need to use the absolute path to the file on your local image, and that before you hand in, you should change the path name back to the /course/cs0160 file path.

14 What to Hand In

1. A filled in and commented MyPicturePane class and an unchanged App class. NOTE: Change the image path name back to the /course/cs0160 file path in App.java if working locally.

2. Any other classes you wrote to help along the way.

3. A README named README.txt (see the README Guide for help).
15 Reflection

15.1 Silly Premise

Sadly, you just got word that Madison Kocian has decided not to compete in this year’s games, so you need to get a new picture for the front page without Madison Kocian in it! But the picture you already chose is sooo adorable, so instead of finding a new picture, you decide to try to crop Madison out.

In order to remove Madison Kocian from the picture, you have decided that the best way to do so is by creating a mask of the original picture, where all of the pixels representing Madison Kocian are colored black, and the rest of the picture is white. You can view the mask at /course/cs0160/lib/seamcarve-images/objectRemoval.jpg. Where the mask is black, the pixel values are 0 and where the mask is white, the pixel values are 1. We have already successfully removed Madison from the picture by multiplying these mask values by their corresponding importance values from the original picture. This ensures that the importance values are lowest at the pixels corresponding to where Madison is in the picture.

15.2 Your Task, Handing In, Demos

1. Note that there are no new code files to hand in or install. Your task for this part of the project is to answer the following questions below in a PDF.

2. To hand in, go to the directory you wish to hand in, which should be /home/<your login>/course/cs0160/seamcarveReflection, and run cs0160_handin seamcarveReflection into a terminal. Please make sure that your PDF is located within the correct directory, or we will not be able to grade it.

3. To run a demo of this part of the project and see how object removal works together with seamcarve, run cs0160_runDemo seamcarveReflection.

15.3 Reading

You may find a brief elaboration on the object removal process here helpful.

15.4 Written Questions

In a separate document, please answer the following questions thoroughly. Each answer should be 3-4 sentences and will be graded on thoughtfulness, clarity, and specificity. There are no specific right or wrong answers! Submit your file as a PDF separate from your code using the hand-in script cs0160_handin seamcarveReflection.

1. What are some real world consequences of photo manipulation? Please be specific and respond with a thoughtful answer.
2. How could you as the programmer of this algorithm mitigate malicious use of seam carving? If you believe that you cannot prevent misuse of this algorithm, please explain why.

3. Are there situations in which removing someone from a photo would be justified? If so, give an example and explain why.

15.5 What to Hand In

Submit a PDF containing your answers to the written questions above. Note that this is separate than the handin for your seamcarve code.