Project #1
Seamcarve

Out: Thursday, January 26
In: Monday, February 13, 11:59 PM
"I shrunk the kids."
-Wayne Szalinski

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, and run `cs0160_handin seamcarve` into a shell. This will probably be `/home/<your login>/course/cs0160/seamcarve`.
3. To run a demo of this project, run `cs0160_runDemo seamcarve`.

2 Introduction

2.1 Silly Premise

After the success of their father's shrinking machine, Amy and Nicky want to try to invent something too. They've decided to create a computer program that can shrink images, however they can't agree on how to decide which seams of the photo to get rid of first or how to calculate these seams. They've decided to shrink you down to pixel size to help them decide which pixels are important and they won't unshrink you until you've completed the algorithm.

2.2 What You'll Do

In this project, you will implement the interesting parts of the seam carving algorithm for image resizing that you learned on the first day of class. We've done a lot for you - like give you a GUI and a lot of stencil. You will 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 isn't trivial so you'll want to use your program design skills to simplify your code as much as possible with helper methods and maybe even helper classes. 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 will lose points.
- You must use dynamic programming to find the lowest cost seam. In other words, your code should not have running time greater than O($w * h$) 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 will lose points.
- 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 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).

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 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

5 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. To do this 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).

6 Your Code

For this project you will only be required to implement the `findLowestCostSeam()` method. You are 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 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 image pixels.
- 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. This 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 x coordinate (column) 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:

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

7 Support Code

The following are methods of `MyPicturePane` that are inherited from `PicturePane` which you will 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 y, int x)` Returns the color of the pixel (y, x). 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`).

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.

8 On `javafx.scene.paint.Color`

In the support code we've provided you with 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.

9 Compiling and Running

To compile your program from the terminal, type `make` into a shell. To compile *and* run your code, simply type `make run` instead. If you're using Eclipse, see section 11.

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've put a few tiny images into the course directory. These images will allow you to use `System.out.println`'s without flooding your terminal with tens of thousands of numbers. These images will be helpful if you're having trouble calculating importances correctly or finding the right seams. 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 Using 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.

11.1 Working Locally

In order to do seamcarve on your own computer, you will have to copy the .jar files listed above from the department filesystem into your local directory (to set up file transfer for the first time, see the relevant sections in the following references for PC and OSX). Then, follow the same Eclipse set-up instructions as above.

You will 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/honey.jpg";
```

with

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

Note that you will need to use the absolute path to the file on your local image, and that **before you handin, you should change the path name back to the `/course/cs0160` file path**.

12 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).