Pamela Cheema

June 9, 2019

Seam Carver Project Reflection

I first initialize the height, weight, and the colors values of the picture.

In my energy method, I check to see that the x and y coordinates passed are valid for the current pic, and if not then I throw an exception. Then check to return BOARDER\_ENERGY of 1000 if the pixel coordinates is on the boarder of the pic. Otherwise I calculate the energy of the current pixel by using the equation found in the spec. I broke up the energy calculations into two methods: one to calculate the energies of the X coordinates, and the other to calculate the energies of the Y coordinates. After I calculate the pixel energy, I round it to two decimals using a rounding method I included.

For find vertical and horizontal seams, I visited each pixel in each row and column (depending on the method) and relaxed the edges to any pixels that were not on the boarder. Then compared the distances from each pixel to find the shortest path and add the next pixel to the seam.

For remove vertical and horizontal seam methods, I copied the color values into another, smaller, array, but skipped over the colors at the indexes in the seam given to modify the pic size.

My code finished with exit Code 130.

Originally, I planned to just compare the values in each row for finding the vertical seam, and compare the values in each column for finding the horizontal seam. However, I ended up going pixel by pixel to calculate the edges and distances to find the shortest distance.

The main challenge I had when writing my code was navigating through the rows and columns. I continuously mixed up the pointers for both these positions, which gave me either incorrect seam calculations, or IndexOutOfBounds exceptions for trying to access energies that were greater than the height or width.