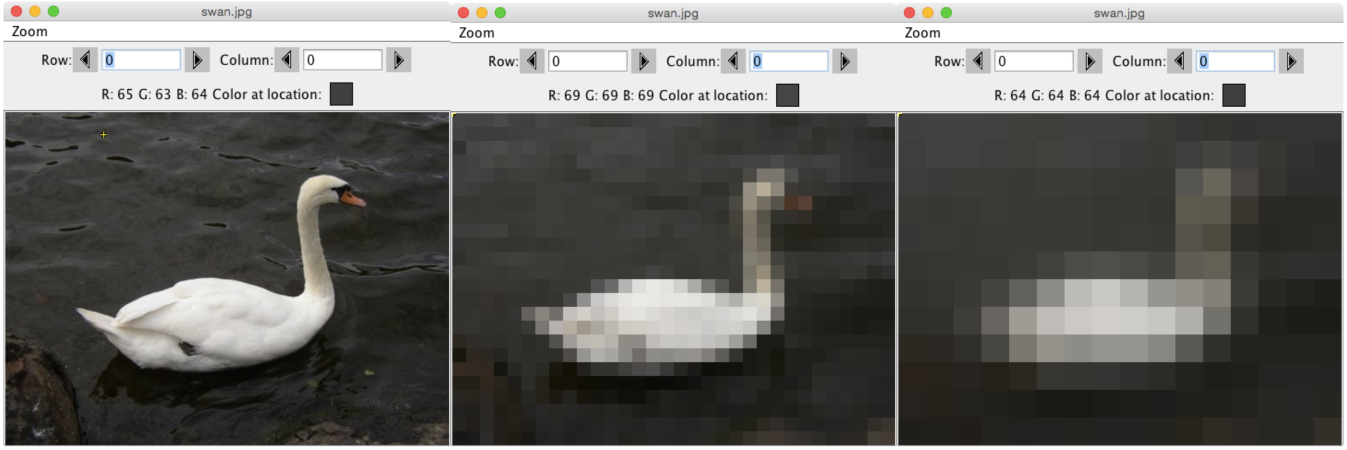
AP Computer Science A
Picture Lab 4B: Working with Pixels

Name ____
Date ______________________________ Period __________

Objective: To amplify all the pixels in a picture using different algorithms.

Background:
Today's cameras take pictures with very high resolution, often 300-pixels per inch (ppi) or more. This density makes these pictures appear very detailed and you cannot distinguish the pixels. What if we used less information? The picture would look grainy and you could see the individual pixels. Use PictureExplorer and magnify a picture to 500%. Notice how grainy it has become and how you can see individual pixels.

What if we could make the whole picture grainy? We could see each pixel and the picture would appear blurry, like below.



swan.jpg
Zoom
Row: 0 Column: 0
R: 65 G: 63 B: 64 Color at location:

swan.jpg
Zoom
Row: 0 Column: 0
R: 69 G: 69 B: 69 Color at location:

swan.jpg
Zoom
Row: 0 Column: 0
R: 64 G: 64 B: 64 Color at location:

Important!!! Your project directory and program naming conventions are very important when Mr Greenstein auto-grades your project. You must use specific names and method signatures to run your project. If these names do not match, then your grade will be delayed and possibly lowered.

Directory name: **pixLab** - Inside your MyDocuments directory (default directory).

Class name: **Picture.java** - All your methods should be in this class.

Activity:
A1) Create a new method `pixelate(int size)` inside **Picture.java** that pixelates a picture. Break the picture into a grid of **size** by **size** pixel squares, get the average red, green, and blue of each square and set the square to that color. Perform this to the picture several times increasing **size** each time.

Use the following method comments and signature to begin your method:

```
/** To pixelate by dividing area into size x size.
 *  @param size     Side length of square area to pixelate.
 */
public void pixelate(int size) {
```

Blurring a Picture

Pixelating is a form of blurring a picture. Another way to blur an image is to process each pixel and average the pixel colors around it to make the new pixel.

For example, the following is a 4x4 array of pixels representing a picture. We will use a 3x3 area of blur. The algorithm takes the highlighted pixel in the original picture, calculates the average of the shaded 3x3 pixels around it

(50+65+70+30+20+40+100+25+150) / 9 = 61

and puts that number into the new picture (blurred picture below). It does this for each pixel and you get the blurred picture.

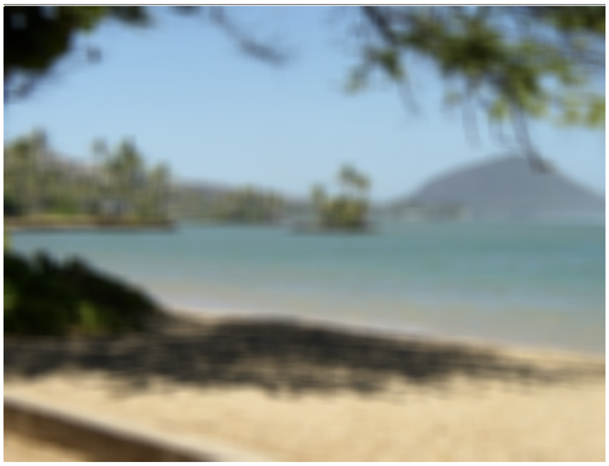
50	65	70	80
30	20	40	35
100	25	150	120
200	180	160	140

Original picture

41	45	51	56
48	61	67	82
92	100	96	107
126	135	129	142

Blurred picture

The key is not to change the original picture, but to create a new picture and pass that picture back to the calling method. Below right is an example of a blur using a 11x11 blur area.



Activity

A2) Write a new method `blur(int size)` that blurs a picture using the `size` of the blur area.

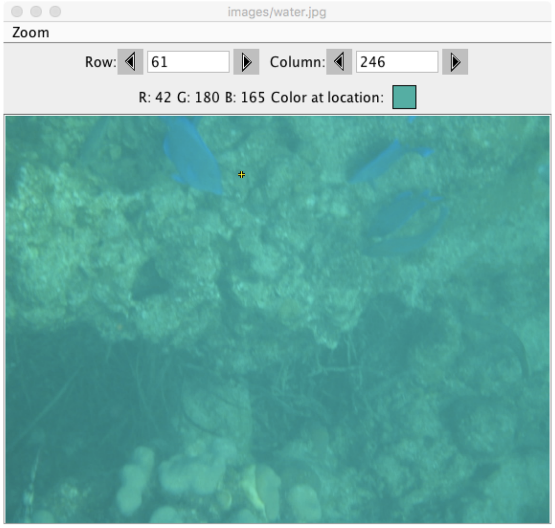
Use the following method comments, signature, and code to begin your method:

```
/** Method that blurs the picture
 *  @param size    Blur size, greater is more blur
 *  @return        Blurred picture
 */
public Picture blur(int size)
{
  Pixel[][] pixels = this.getPixels2D();
  Picture result = new Picture(pixels.length, pixels[0].length);
  Pixel[][] resultPixels = result.getPixels2D();
```

Enhancing a Picture

Pictures are always more vivid in real life. When we commit them to a photograph, they often loose their crisp details and brightness. To bring some of that clarity, we can deploy photo enhancing tools to sharpen and brighten the picture.

You will use a technique similar to the blur method, except the information will be used to enhance the photo. Below, the photo on the left is the original water picture with fish. On the right is an enhanced version of the same photo.

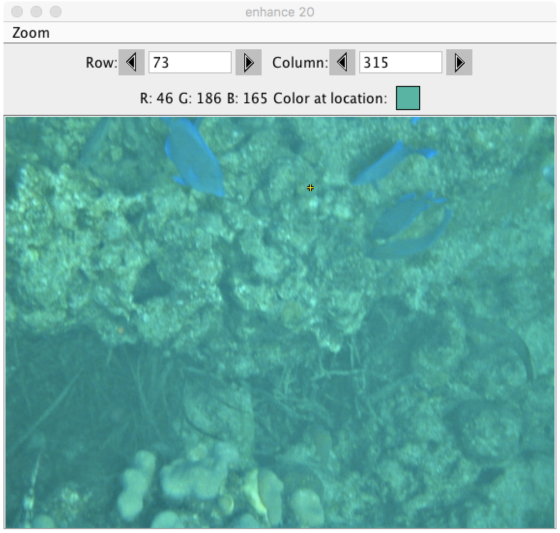


images/water.jpg

Zoom

Row: 61 Column: 246

R: 42 G: 180 B: 165 Color at location:



enhance 20

Zoom

Row: 73 Column: 315

R: 46 G: 186 B: 165 Color at location:

Like the blur feature, the algorithm gets the average color of the pixels around itself then uses the following formula:

```
new pixel color  <-  2 * current pixel color - average area color
```

Perform this operation on each pixel's red, green, and blue of the photo.

Just like the blur feature, do not to change the original picture. Create a new picture and pass that picture back to the calling method.

Activity

A3) Write a new method `enhance(int size)` that enhances a picture using the `size` of the enhanced area.

Use the following method comments, signature, and code to begin your method:

```
/** Method that enhances a picture by getting average Color around
 *  a pixel then applies the following formula:
 *
 *    pixelColor <- 2 * currentValue - averageValue
 *
 *  size is the area to sample for blur.
 *
 *  @param size    Larger means more area to average around pixel
 *                         and longer compute time.
 *  @return                enhanced picture
 */
public Picture enhance(int size)
{
  Pixel[][] pixels = this.getPixels2D();
  Picture result = new Picture(pixels.length, pixels[0].length);
  Pixel[][] resultPixels = result.getPixels2D();
```