CS61BL: July 11th, 2012 – Discussion of Blur Function's Run-time

**From:** Michael Yang, David Chang (CS61BL Sec 105, cs61bl-ed, cs61bl-ef)

**Memorandum for:** Courtney Wang

**Subject: Discussion of run-time for blur feature in new Picture manipulation software product**

**Summary:** The blur function in the Picture software takes significantly longer to run than other functions because the nature of its algorithm requires that, for each pixel, an area of surrounding pixels is analyzed. The user decides the size of this area, and every increase in this size causes a significantly longer run-time.

**Introduction:** Blur is a function in the Picture package that allows the client to apply a blur effect to a bitmap picture. Essentially, it averages the red, green, blue, and alpha values of an area of pixels surrounding the selected pixel, and sets that pixel to those average RGBA values. A method called averagePatch does this. We use averagePatch for all pixels in the picture. And by inputting a parameter called threshold to blur, the user determines the size of this area, which is a square with sides (2*threshold+1).

The run-time of blur tends to be relatively long because it must check each pixel of the area, which is (2n+1)x(2n+1) pixels large, where n is the threshold or range chosen by the user. This long iteration is done to each pixel in the picture, therefore, the number of pixels to analyze would be **$((2n+1)^2$ x total pixels)**. This means that as the threshold n increases, the run-time would increase quadratically, which is very slow. As the number of total pixels increases, the run-time would increase linearly.

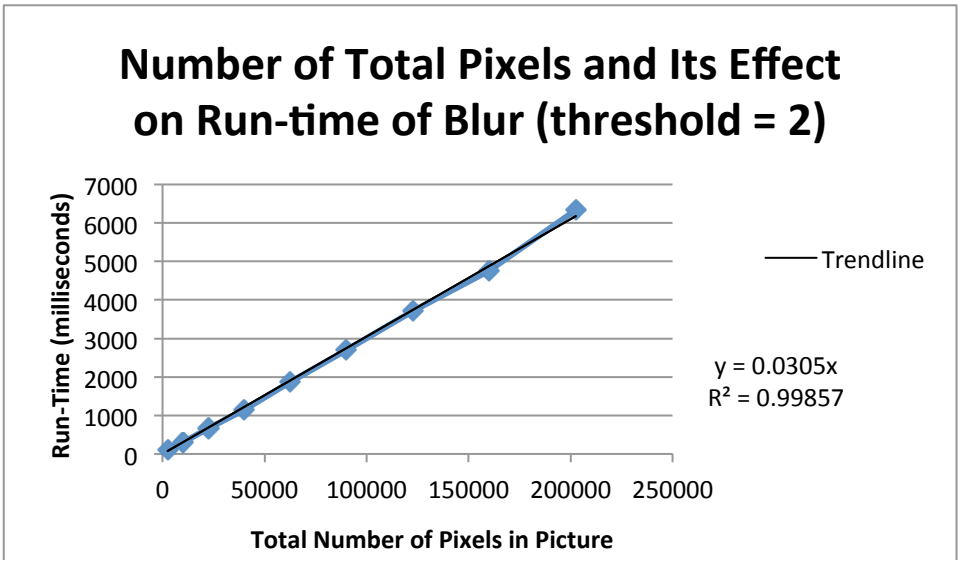
**Methods:** Blur's run-time was analyzed by varying the two factors that affect its run-time: total pixels and threshold.

Our first test varied the number of total pixels: we tested the run-time of differently sized "Creek" pictures, from 50x50 pixels to 450x450 pixels, and incremented by 50x50 pixels each time. The area parameter was held constant at n = 2 (so each pixel checked the surrounding 5x5 square of pixels). We analyzed the average run-time of each these differently sized pictures and checked if it was in accordance to the run-time growth model we discussed in the Introduction.

The second test varied the area parameter n, and used the "Colleen.bmp" picture. The average run-time for blur for each threshold was recorded. We analyzed these run-times and checked if it held to our run-time growth model.

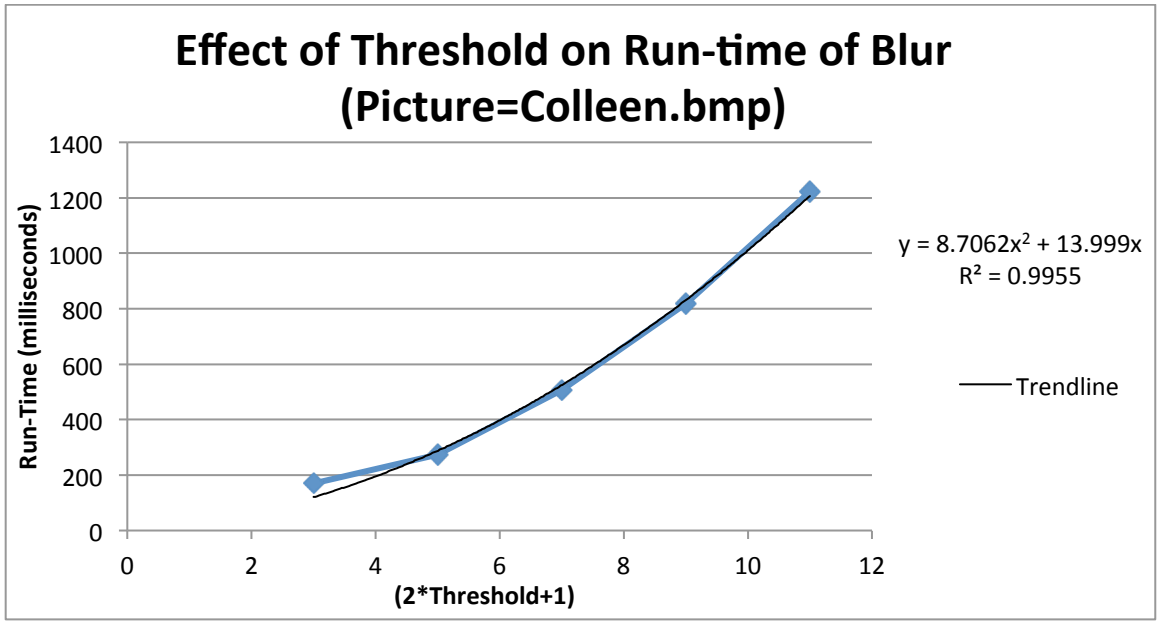
**Evidence:**

As one can see from the graph on the right, the data fits the trend line very well (high $R^2$ value), which means that run-time does indeed



increase linearly as the total number of pixels increases.

In the following graph, one can see that the quadratic trendline fits the data very well. This means that the run-time does indeed increase quadratically as the threshold increases.




Note: Supportive code can be found in Appendix. The code uses nine images that are attached to the pdf file.

**Conclusion:**

Again, the total number of pixels "blur" must check is modeled by this equation: $(2*\text{threshold} + 1)^2 *$ (total pixels in picture). Increasing the total number of pixels or threshold would increase the number of pixels "blur" must check, and hence, the run-time would increase as well. However, increasing threshold would cause a much greater (in fact, quadratic) increase in run-time compared to increasing the total number of pixels (which would simply be linear).

Therefore, it is advised that the user is not allowed to set the threshold to a very large number, because this would cause the program to run for too long.

**Appendix:**

```
public void testRunTime() {
        LinkedList<Long> times = new LinkedList<Long>();
        for (int n = 5 ; n < 50  ; n += 5) {
                long total = 0;
                for (int r = 0 ; r < 6 ; r++) {
                        Picture pic1 = Picture.loadPicture("Creek" + n + "0.bmp");
                        long start = System.currentTimeMillis();
                        pic1.blur(5);
```

```
                    long end = System.currentTimeMillis();
                    total += (end - start);
                }
                times.add(new Long(total/5));
        }
                System.out.println(times);
}

public void testRunTimeThreshold(){
        LinkedList<Long> times = new LinkedList<Long>();
        for (int n = 1 ; n < 6 ; n++) {
                long total = 0;
                for (int r = 0 ; r < 6 ; r++) {
                        Picture pic1 = Picture.loadPicture("Colleen.bmp");
                        long start = System.currentTimeMillis();
                        pic1.blur(n);
                        long end = System.currentTimeMillis();
                        total += (end - start);
                }
                times.add(new Long(total/5));
        }
        System.out.println(times);
        assertTrue(true);
}
```