ComS 455: Homework 1

Gaussian Pox

1 Requirements

In this homework, you will revisit class design in C++ and hone your understanding of rasters, color, and normalization. You will do these these in the context of creating a *heightmap* of a landscape pocked with Gaussian-shaped hills.

This homework does not directly involve OpenGL.

1.1 Image

Write a class `Image` which abstracts a 2-D raster of pixels, with three color channels per pixel. Assume color intensities are in $[0, 1]$. `Image` must have the following interface:

- A constructor taking parameters for width and height.
- `operator()` taking two `int` parameters for row and column coordinates and returning a `float` pointer to the red channel of the pixel. (Thus, `image(5, 3)[1]` would reference the green channel of the pixel at row 5, column 3.) This method allows easy pixel access to clients of your class. Provide both `const` and non-`const` versions.
- Getters for the image width and height.
- A method `WritePPM` which accepts a `const std::string` reference parameter to a path on disk. The image data in $[0, 1]$ is range-mapped to $[0, 255]$ as it is written to the file in ASCII PPM format. Use 255 as the maximum intensity.

1.2 Main

Write also a `main` function that randomly generates a sequence of Gaussian bumps and *splat*s them onto an image. Each bump can be represented with three values: an xy location and a *variance*—often called σ^2. Generate the xy location to be within the image dimensions. Then, for each pixel, sum up how much energy it receives from each of the Gaussian bumps.

The energy received at a pixel (r, c) from a bump at (μ_y, μ_x) can be calculated using the 2-D Gaussian function as follows:

$$\text{energy} = e^{-(\frac{(c-\mu_x)^2}{2\sigma^2}+\frac{(r-\mu_y)^2}{2\sigma^2})}$$

Once the total energies for all pixels have been determined, map a pixel's energy to its color. The particular mapping you choose is unspecified. The very simplest mapping is to divide a pixel's intensity by the maximum intensity found across the entire image. This operation *normalizes* all energies into the range $[0, 1]$. Normalized values can be thought of

as generalized proportions—which are easy to apply to other domains. In this case, since color is also represented by numbers in $[0, 1]$, we can use the normalized energies directly to set color. If we use the normalized energy to assign all three color channels equally, we'll produce a pleasantly organic grayscale image.

This simple mapping is rather boring. I encourage creativity and color.

2 Memory in C++

Dynamically allocating 2-dimensional arrays using `new` or `malloc` is complicated in C and C++. One must allocate the outer array and then use a loop to allocate each inner array. These *arrays of arrays* don't exhibit the same properties as real two-dimensional arrays. In particular, since the inner array are each allocated separately, they may appear in different parts of the heap. When we start shipping image data to the graphics card, we'll need the data to be stored contiguously.

Stack or static allocation (e.g., `int pixels[5][3]`) does allocate contiguous memory for a multidimensional array, but to statically allocate we must know the dimensions of the array at compile time. Your `Image` class constructor does not know the resolution at compile time and must dynamically allocate its raster.

To dynamically allocate contiguous storage, we allocate a flattened 1-D array and map the row-column indices to the correct 1-D index. Assuming row major flattening, to set the blue intensity of the pixel at row 4, column 10 in a 512 by 256 image, we might have:

```
float *pixels = new float[512 * 256 * 3];
pixels[(512 * 4 + 10) * 3 + 2] = 1.0f;
```

This funny indexing should be hidden away inside the `operator()` method of your `Image` class. If clients of `Image` use `operator()`, they need know nothing about how the pixel data is actually stored.

3 Submission

Push your code to Bitbucket. Post an example image or two on the course blog. Describe your thought processes and any hurdles you had to overcome. Use categories `spring 2015`, `cs455`, and `postmortems`.