Jumana Schmidt (jumanas2), David Medina (davidrm3), Roy Lu (roylu2), Will Marshall (wjm5)

Project Proposal: Simulating Hydraulic Erosion

1. **Leading Question:** *How and Why Simulate Hydraulic Erosion?*

For this project, we intend to simulate erosion from water on a given terrain. There are many reasons as to why this project is useful. Many different types of erosion are used in countless graphics softwares and games. Many people even use specifically hydraulic erosion to make their generated terrains more realistic and natural.

The program should take in a black and white image of a heightmap of some terrain, and this image will essentially be our dataset. We will expand on the algorithm, but the foundation of our algorithm will include traversing and processing the image. Some abstract amount of water will be placed on a pixel, and the algorithm will look at the surrounding nine pixels for the pixel with the lowest elevation or darkest color/brightness value. As the water follows this path, it darkens and changes the height value of the pixels it passes through (ie. eroding the path).

There are many ways to expand this algorithm, but we plan on incorporating sediment into the erosion or darkening factor. So, as the water follows down the terrain until it reaches flatground or pixels with all the same color/lightness value, it picks up material that makes the erosion exponential. Likely we will run into other issues or necessary changes for exceptions or ways to make it look more realistic, such as probably editing surrounding pixels that the water passes through. This change will be harder to implement because it will require accounting for edge cases, but it is probably necessary because the erosion will only darken in a line otherwise. Additionally we could also account for different types of material and how they erode versus others. Later, likely after the last mp, we can 3-D render it and color process the final image using OpenGL or even some parallel programming.

For the structure of our code, we will make our own heightmap class that operates on heights, sediment, color value, and possibly material. We also will likely use the PNG class to process the image and HSLAPixel to process color, as well as take concepts from the MP traversals.

2. **Dataset Acquisition and Processing**:
    a. *Data Format:*

The input will be a PNG image of a heightmap, so part of our algorithm will need to be converting that image into our dataset. So, we will get height and sediment values and generate HSLAPixel luminance values for each pixel. These amounts will be stored in an *array format*. Likely, the image will be of lower quality, so this storage option will be viable and easier to traverse. For this part, we will use the entire dataset, but depending on how many pixels water is placed on, the algorithm could only access a part of it (ie. the part of the image the water traverses).

    b. *Data Correction:*

For the algorithm, its method of parsing the data is described more thoroughly above, including its edge cases. For actually implementing the erosion, we will definitely have an edge case of not allowing the algorithm to darken below the viable range of luminance. We also cannot traverse or edit pixels outside of the image/array, so we need a bounds check throughout.

    c. *Data Storage:*

The sediment, height, and luminance values will be stored in a *2-D array format*. Likely, the image will be of lower quality, so this storage option will be viable and easier to traverse. For this part, we will use the entire dataset, but depending on how many pixels water is placed on, the algorithm could only access a part of it (ie. the part of the image the water traverses).

3. **Graph Algorithms**:
    a. *Function Inputs:*
        i. The input for the erosion algorithm will be a black and white image height map. This image will have a max size of 600 x 600 pixels. This will allow us to easily create a 2D array that will hold the pixel data for each pixel. The pixel data will be

read using the HSLA pixel class that is within the PNG class. We decided to go with an array formatting since it allows to work with images that visually easy to test and prototype code. For fun, colored images and actual height maps can be used and then converted with the same erosion algorithm. This would be a more realistic approach to the algorithm.

    ii. For the height map for the OpenGL 3D plane imagem we'd need to take the original greyscale image and run it through another algorithm that will define the vertices of the 3D image. We are basically creating triangles by plotting this mesh and then adding the height depending on the greyscale factor. This while require a 2D array input of the pixel and then another function to place into new 3D array for plotting in OpenGL

b. *Function Outputs:*
    i. The expected output of the erosion algorithm is an image with the drawn in water flow of where the erosion would be going on. It should use the input image and modify and output an image with the drawn in erosion map of water flow.
    ii. For fun, a 3D rendering program called OpenGL could be used to render the image in 3D. We'd read the file into an array and use the erosion algorithm to decide what to color in. From here apply the z-axis conversion depending on the the greyscale value. This would then create a height map of the image that we can render in 3D. This will then allow us to create a 3D plane map using the "OpenGL terrain rendering demo".

c. *Function Efficiency:*
    i. The time complexity to traverse a 2-D array is found to be O(n*m), where n is the number of arrays, which correlates to the first dimension of our 2-D array, and m is the max size of the arrays, our second dimension. Knowing this the actual time complexity can be described as O(N), meaning it is linearly related with the size of the input.
    ii. The time complexity for the OpenGL heightmap output is heavy. Essentially we are generating two triangles per meshmap tile and the uploading them to get processed by the OpenGL software. Since we are also working with arrays here and a new 3d array most likely, the time complexity will jump to O(n^3) worst since we'd have to iterate through three times.

4. **Timeline**:

Week 1 (March 28th - April1st): Research/have Mentor explain OpenGL, find heightmap with linked height values, and write README in github

Week 2 (April 4th - April 8th): Write class structure with private variables that store sediment, height, & luminance, and write code to process the input PNG image to storing values in an array

Week 3 (April 11th - April 15th): Research how erosion actually works (how much terrain gets eroded), and write pseudocode for basic erosion algorithm: traversing array, decreasing height & luminance values, and account for edge cases

Week 4 (April 18th - April 22nd): Finish & test functional code (test output image), make algorithm more realistic (soften hard erosion line by eroding pixels around water's path), and make algorithm more efficient (Big O)

Week 5 (April 25th - April 29th): Write pseudocode & basic real code/class for OpenGL, learn about parallel programming, and write written report & start presentation

Week 6 (May 2nd - May 6th): Finish complete functional code with OpenGL (test output 3D model and find Big O), and finish presentation