Towards an Output-Sensitive Viewshed Algorithm

The viewshed of a point on a terrain is defined as the area visible to that point. Viewshed computation algorithms use a grid terrain (a matrix of elevation values) and an arbitrary viewpoint to determine the set of all points in the terrain that are visible from that point. Viewshed computation has applications in many disciplines such as path planning, navigation, landscaping, and placement of cellphone towers. This summer I worked on the development of an algorithm for the computation of viewsheds in linear, O(n), or sub-linear time.

The efficiency of viewshed algorithms becomes important when large data sets are considered and viewshed computations are run continuously. Traditional algorithms assume all the terrain data fits in the main memory of a computer, but in reality the computer’s hard disk will need to hold some of the data. Since the hard disk is approximately one million times slower than the main memory, this causes the computation time to bottleneck. Some of the best viewshed computation algorithms run in O(nlgn) time, where n is the size (number of data points) of the input terrain. Even if one viewshed computation takes one second, one hundred thousand computations will take about 25 hours.

We began the summer by working with code written by Phillip Koch, a previous student of Professor Toma who developed a single-pass algorithm for computing viewsheds. We thoroughly tested his code and qualitatively compared the viewsheds it developed with those created by other existing base algorithms. We found very few discrepancies, likely due to precision issues. We also found that Koch’s algorithm fails to run on the dataset Washington, which contains approximately one million points. Unable to debug this algorithm, we began work on a different viewshed computation algorithm.

During the second part of the summer we worked on a new approach for computing viewsheds on grids. The general idea is to compute a low-resolution grid and use it to speed up the computation while at the same time not sacrifice the quality of the results. We developed code to develop a low-resolution grid depending on a parameter specified by the user. We then developed a module for handling horizons. For a given viewpoint within a grid the horizon is the collection of maximum heights around the viewpoint that limit the visibility of more distant points. The two basic functions of this module are to insert a segment into a horizon, and to determine whether a segment is below a horizon. The elevation of an arbitrary point in the grid with respect to the viewpoint determines if the current segment needs to be inserted into the horizon.