The last decade has witnessed an increase in the amount of terrain data collected in the geosciences. This comes from remote sensing devices and from newer technology such as LIDAR (light emission and detection). The massive amounts of high-resolution terrain data bring potential for high accuracy modeling processes on terrains, but also present the challenge to develop algorithms that are efficient and scalable. This summer I worked on the problem of computing viewsheds on terrains. The viewshed of a point on a terrain is the set of points on the terrain that are visible from the given viewpoint. Computing viewsheds is one of the fundamental problems in computational geosciences, with a multitude of applications that range from planning and landscaping to placement of cell phone towers. The best known viewshed algorithms run in $O(n \log n)$ time, where $n$ is the size (number of points) of the input terrain. For repeated viewshed computations on large terrains this is too slow and not feasible in practice.

In this work we are investigating a new approach to speed up the viewshed computation, while still computing an exact viewshed (that is, not using approximation). The observation is that viewsheds are relatively small compared to the size of the terrain. Our basic idea is to divide the grid into blocks, and to identify and eliminate blocks of points that are guaranteed to be invisible, and take them out of the grid; on grids where a lot of blocks are eliminated, this will effectively reduce the size of the input.

More precisely, the approach is as follows: First, in a pre-processing step, we split the grid in blocks of size $k$ by $k$ and record some minimal information about each block (the min and max elevation); this step runs in $O(n)$ time. Second, we read the blocked grid (now a grid of $n/k^2$ points, each corresponding to a block), and determine which blocks are guaranteed to be invisible. This step runs in $O(n/k^2 \log n/k^2)$ time. Finally, in the third step, we perform a viewshed computation on the grid from which we eliminated the invisible blocks; let $n'$ denote the number of points in the grid after the points in the invisible blocks are filtered out. This step runs in $O(n' \log n')$. In the worst case, no block is marked invisible, $n'=n$, and the first two steps become an overhead. However, when a large number of blocks are determined to be invisible, we have that $n' \ll n$ and the total viewshed running time (step 2 + step 3) is sub linear.

This summer I have implemented the first two steps and run preliminary experiments. Given a terrain, I have a script that picks a large number of viewpoints randomly, runs the block elimination algorithm for each one in part, and records the area that is eliminated as a percentage of the total size of the terrain. The results are extremely promising and show that, on the average, a large part of the terrain is filtered out by our pre-processing steps. For example, on the Appalachians grid ($n=66$ million), with $k=10$, and $N=400$ sample viewpoints, on the average $98\%$ of the terrain is marked invisible and eliminated. On dataset srtm1.region03 ($n=2$ billion), with $k=100$ and $N=100$ viewpoints, on the average $87\%$ of the terrain is eliminated and the process takes approx. 5 minutes (for a comparison, computing the viewshed on this dataset takes a few hours). I have experimented with various values of the block size $k$. The smaller the block size, a larger part of the terrain is eliminated, but the running time is also larger.

Developing the third part of this process, computing the viewshed on the resulting grid with holes, is topic for a further project.
Faculty Mentor: Laura Toma

Funded by the NSF Grant #0728780 for Scalable Algorithms for Realistic Terrain Processing in GIS