

Force-Directed Graph Layout using the GPU

Takeshi Hakamata*

Thomas Caudell†

Edward Angel*

Background

Graph layout has had important applications in many areas of computer science. When dealing with machine generated data, we often want to see the data to have better understanding of the structure or organization. Many such data are represented by graphs. By laying out a graph, we can untangle information and intuitively show relations of objects.

We present a force-directed graph layout of a 3D graph using the GPU and show the performance advantages of using the GPU over the CPU. The main challenge to implement the algorithm is how to represent a graph using texture maps to make the computation possible and efficient on the GPU.

Methods

The force-directed layout consists of two components: a *model* consisting of physical objects representing the elements of the graph, and an *algorithm* to compute an equilibrium configuration of the system. The classical force directed layout starts with a random embedding of the graph and apply an algorithm and refine the configuration of the graph until it reaches equilibrium. Our approach uses Hooke's law to compute both per-edge attractive forces and repulsive forces. Repulsive forces are computed using springs of a long natural length to all the other nodes.

We use a texture to represent per-edge forces, and two more textures to store node positions. The latter two textures are ping-pong buffered. We use two fragment shaders to implement the algorithm: one for updating per-edge forces and another for integrating node positions using the computed forces.

In the simulation loop, the program updates the texture containing forces, then updates positions of nodes in a fragment shader by rendering positions to the destination texture. A readback of the node positions from the GPU memory to the CPU memory only occurs when rendering of the graph is required.

Results

We compared performance of our algorithms and the CPU implementation of the same algorithm by measuring the time to run 100 iterations of the simulation. The speedup of the GPU implementation of laying out an 8×8 grid with 64 nodes and 224 edges is 4.04, and 15×15 grid with 225 nodes and 840 edges is 2.55.

Conclusion

We showed the performance improvement of the graph using the GPU. However, due to the nature of the algorithm, large graphs cannot be laid out since they would require too much GPU memory. Running time is also a problem. The node integration takes $O(n^2)$ time since it has to look up all other nodes in the graph to compute the net repulsive force. We are working on solving the latter problem.

*Department of Computer Science, University of New Mexico

†Department of Electrical and Computer Engineering, University of New Mexico