A Novel Single Projection Method to Calculate View Factors for Radiosity, and a GPU-Based Implementation in a Large Multiphysics Code

Sriram Swaminarayan, John Turner, CCS-2

We describe a new method for calculating view factors for radiosity using a single Cartesian projection, and its implementation on graphics hardware to accelerate heat transfer in Truchas, a large multiphysics flow and solidification code developed at Los Alamos National Laboratory. By utilizing the hardware accelerated z-buffer of the Graphical Processing Unit (GPU) we are able to achieve speedups of the order of 30 times in the view factor calculation.

Algorithm

Our single projection method is similar to the Hemicube method [1] in that we do Cartesian projections, but is more efficient since instead of five projections, we only project onto a single plane. This requires modification of the original hemicube equations to explicitly include the distance to the imaging plane as follows:

$$dF = \frac{z_o^2 \, \Delta^2}{\pi \left(r^2 + z_o^2 \right)^2} \qquad (1)$$

where dF is the contribution to the view factor matrix by any pixel on the imaging plane, z_0 is the height of the imaging plane from the center of the face of interest, Δ is the size of the pixel, and r is the radial distance to the pixel from the z-axis.

As is evident from Eq. 1, the pixels closest to the z-axis will contribute the most to the view factor matrix (inverse dependence on r^4). Consequently, we use a finer mesh closer to the center of the imaging plane and a coarser mesh further away. On the GPU this is achieved by simply changing the field of view while keeping the window size fixed. This allows us to achieve high accuracy in the view factor calculation where it is needed most without sacrificing speed. Figure 1 demonstrates this for a geometry of concentric spheres where three levels of refinement have been used to calculate view factors. The faces are colored by their IDs and the contents of this buffer are read back onto the CPU to combine contributions into the overall view factor matrix. A significant advantage of our method is that we can selectively refine the view factor matrix along directions that need the high resolution due to increased detail in that direction, while at the same time keeping the mesh coarse in other areas.

Implementation

To interface the GPU code into Truchas, which is written in modern Fortran (F90), required a platform-independent library for generating windows and drawing to them using OpenGL. We decided against using OpenGL Utility Toolkit (GLUT) for this since it does not give the application control over the event loop, and it was not easy to recast the F90 code in such a way that it could be called from the display routine of GLUT. We instead wrote a simple library that opens a window on the client machine, and draws to it with the geometry given. This way, the physics code itself has no graphics code embedded in it and we can easily eliminate the GPU-based extensions

when building on machines without powerful GPUs.

Results

Figure 2 shows the times for calculating the view factor using the Hemicube method on a 3.4 GHz 64-bit Xeon and using the plane projection method on an NVIDIA Quadro FX1400 GPU for a problem containing two concentric spheres. For any given problem size, the GPU implementation is at least 30 times faster than the CPU for a comparable resolution.

For more information contact Sriram Swaminarayan at sriram@lanl.gov.

[1] M.F. Cohen, D.P. Greenberg, "Hemi-Cube: A Radiosity Solution For Complex Environments," *Comp. Graphics* **19**, 3 (1985).

Funding Acknowledgements

NNSA's Advanced Simulation and Computing (ASC), Telluride Project; and Laboratory Directed Research and Development.

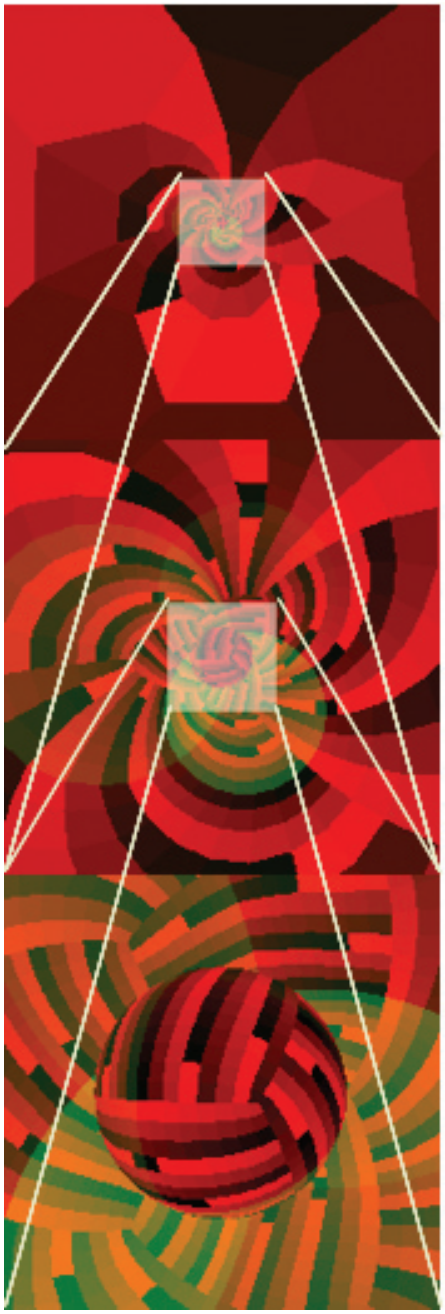

Fig. 1.
Snapshot of one face of the view factor calculation. The faces are colored by their IDs. The frame buffer is read back and the sum of the contributions of each face is used to calculate the view factors for that face. The three panes in the image represent three fields of view used in obtaining this image.

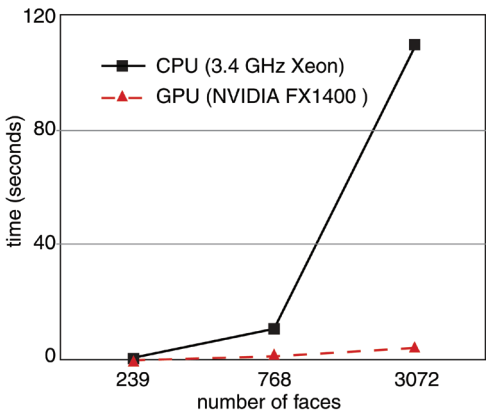

Fig. 2.
Time taken for the Hemicube on the CPU and the plane projection method on the GPU. The GPU computes the view factors at least 30 times faster than the CPU.