LA-SiGMA REU - SUBR

Optimizing Stereographic Visualization of Atomistic Configurations

Cole Vanderlick[1], Sanjay Kodiyalam[2], Amitava Jana[2]
[1]REU Student; Petroleum Engineering, Louisiana State University, Baton Rouge, LA 70803
[2]Mechanical Engineering, Southern University and A&M College, Baton Rouge, LA 70813

7/25/2012

Abstract

A previous CAVE-library based C/C++ Visual Studio project for stereographic visualization of atomistic configurations is enhanced using the GLU library and OpenGL functions. Bonds are better rendered as cylinders. The performance of the code is optimized by the use of display lists. Displaying only atoms without bonds with display lists increases the execution speed as compared to the case without display lists and enables interactive display of ~35,000 atoms. The increase is not as significant when displaying bonds alone without atoms as the corresponding limit on the number of atoms is ~3,000. The project is developed on a desktop and can be used for stereographic visualization in a CAVE.

Introduction

Visualizing atomistic configurations from simulations provides information complementary to numerical data and helps in identifying the mechanisms underlying the phenomena being studied [1]. Molecular dynamics simulations can have a large number of atoms: In many cases exceeding a million atoms. This makes it challenging for interactively visualizing the corresponding atomic configurations. For interactive visualization the frame rate must at least be 10 frames per second (fps) implying that the time for displaying the configuration once must be $\leq$ 0.1 seconds. In this work an earlier version of a CAVE-library based Visual Studio [2] project in enhanced to improve the display as well as increase the number of atoms that can be handled interactively. The code is developed on a desktop and can be used for stereographic visualization in a cave automatic virtual environment (CAVE).

The CAVE at Southern's College of Engineering [2] (Fig. 1) is an 8 ft x 8 ft x 8 ft space with four displays – three on screen-walls and the fourth on the floor. Active stereographic viewing is enabled via the use of eye-ware synchronized to the rapidly alternating display of images corresponding to the left and right eyes. The user's viewpoint is detected via a sensor connected to the eye-ware and the corresponding perspective transformation for each of the displays is automatically carried out by the CAVE-Library. A second sensor is connected to a joystick and can be used for interacting with the user's visualization application. The CAVE is driven by a two node cluster with the Master node collecting and handling the sensors' information and the display node driving the projectors.

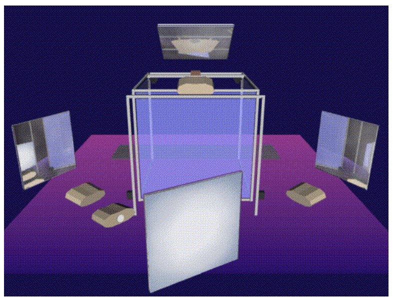

Fig 1. Schematic [3] of the CAVE at Southern's College of Engineering. The display is via mirrors used to set the correct optical distance (= projector's throw) within the space available in the room.

Methodology

The CAVE-library based C/C++ project has separate threads for the main and display loops [2]. Each of the display loops handles one display while the main loop can be used to handle non-display related operations such as computations or IO. On the desktop, and on the Master node in the CAVE, there would be only one display thread with the image generated is non-stereo "CAVE simulator" mode (Fig. 2) while in the CAVE, on the display node, there would be four display threads. Currently the main thread generates a fixed BCC lattice but is otherwise an empty loop that may be used for reading in simulation results with time varying atomic coordinates and other atomic attributes like velocity [1].

The display loop begins with a one-time call to a function initializing the display by setting directional lighting (parallel light rays are used) with ambient, diffuse, and specular qualities. The navigation function within the display loop enables translation, rotation, and scaling of the entire scene. The display function applies the navigation transformation and calls a "Make Display List" function. The Make Display List function uses the atomic position and atomic type arrays and creates the display of atoms of two types, corresponding to the two atoms of the BCC lattice (the "corner" atoms and the atoms at the "body center"), and bonds between the two types of atoms.

For enabling a comparison with previous work [2] atoms and bonds are rendered in two mutually exclusive ways using OpenGL functions and either (1) Calls to sphere rendering [2] and cylinder rendering functions using the GLU library [4] for the atoms and bonds respectively, or (2) Calls to display lists creating the same objects with additional OpenGL calls as needed. The OpenGL functions are for translating to the position of the atom to be rendered [2], and for rotating to orient the bond to be rendered. When using a display list for the bonds the additional OpenGL operation needed is scaling to set the bond length to the distance between atoms – which would be generally be variable as in simulation data while the display list used has the bond length fixed at unity. This approach guarantees that the results for CPU timing reported here for the BCC lattice configuration will be valid for time varying configurations from a simulation provided that any additional time required for reading the configurations can be associated entirely with the main loop. The CPU time for executing the display loop is measured in the display function as an average over hundred frames [2]. Similar to the studies in [2] the variation of this time is determined in the following cases: (a) When displaying a fixed total number of atoms only (no bonds), with the number of primitives (defined as the smoothness measure) used to display each atom; (b) When displaying the atoms alone (all at an equal and fixed smoothness measure), with the total number of atoms; and (c) When displaying the bonds alone (no atoms, and all bonds at a equal and fixed smoothness measure) with the total number of atoms.

Results

As suggested in [2] the display is improved by rendering the bonds as cylinders (Fig. 2b) rather than as lines (Fig. 2a). An advantage of this change was the behavior of the rendered object with respect to directional lighting: The entire scene had to be rotated to be able to see the lines clearly where as the cylinders are seen clearly in the original orientation with directional light coming as parallel rays from the front.

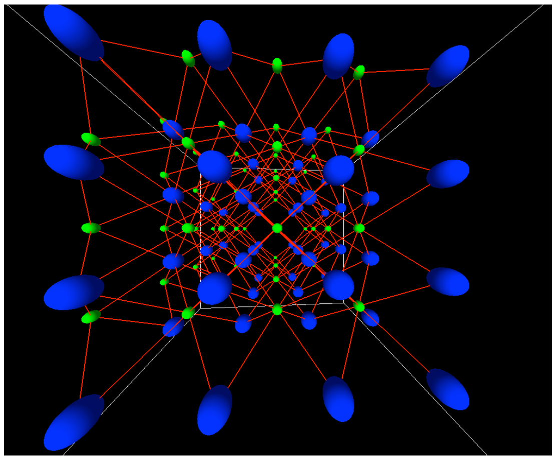
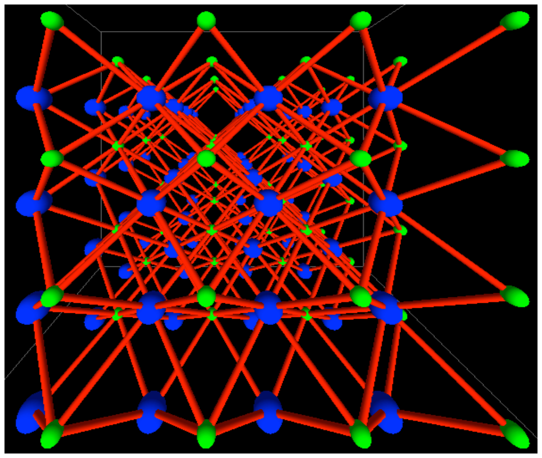

Fig2. Perspective views of atoms & bonds in the BCC lattice configurations in the "CAVE simulator" mode display on the desktop. The image on the left shows the display of bonds as lines (as in [2]) and is rotated relative to the image on the right that shows the display of bonds as cylinders.

In order to compare the execution speed of the current version of the code with previous work [2] the maximum value of the required smoothness measure is initially determined by visualizing a lattice configuration with only a few atoms. A smoothness measure of 20 (slices) x 20 (stacks) = 400 is found to be sufficient for a satisfactory spherical display of an atom. Similarly, for a bond, a smoothness measure of 20 (slices) x 1 (stack) = 20 is found to be sufficient for its display as a cylinder. These numbers determine the range and values of smoothness measures in the subsequent CPU execution timing studies.

Figure 3 shows the variation in the execution time per frame for the display of atoms without bonds as a function of the smoothness measure with a fixed number of atoms (31250) – with and without [2] using display lists. While being in different ranges of the smoothness measures with only a small overlap, the linear fits to the curves suggest that using display lists increases the execution speed by more than an order of magnitude. Without display lists the frame rate falls below the acceptable value of 10 fps at a smoothness measure of 6 while with display lists this happens only at a smoothness measure of ~500.

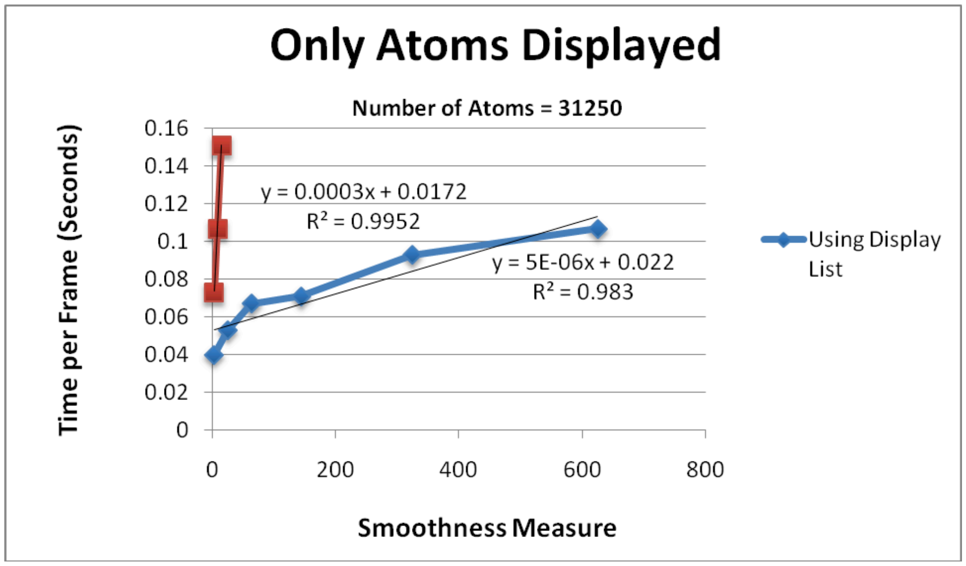

Fig 3. Variation in the execution time for the display of atoms with the atoms' display smoothness measure. The linear fits are a rough guide to indicate the performance of the code with and without display lists.

Figure 4 shows the variation in the execution time per frame for the display of atoms without bonds as a function of the number of atoms – with and without [2] using display lists. The linear fits indicate that the execution speed increases by roughly an order of magnitude. This is reflected in the limits on the number of atoms that can be handled interactively: Without display lists this limit is ~5,000 where as with display lists it is ~30,000.

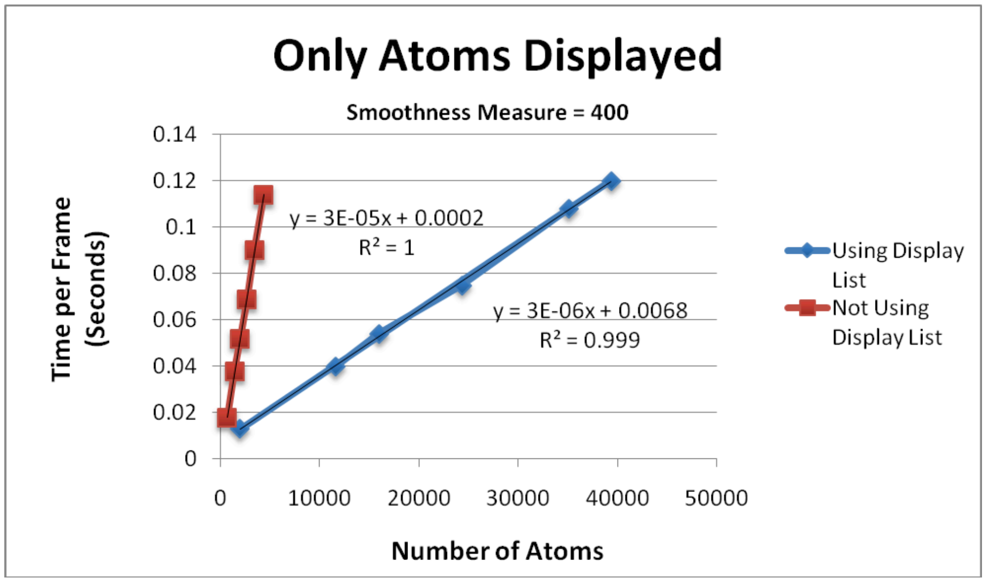

Fig 4. Variation in the execution time for the display of atoms with the total number of atoms. Using display lists for the atoms increases the execution speed by roughly an order of magnitude.

Figure 5 shows the variation in the execution time per frame for the display of bonds without atoms as a function of the number of atoms – again with and without [2] using display lists. As compared to the case of rendering only the atoms (Fig. 4) the overall reduction in execution time on the use of display lists is not as significant. This is due to the quadratic dependence of the execution time on the number of atoms as evidenced in the 2nd order polynomial fits in Fig. 5. Interestingly the coefficients to the linear dependence on the number of atoms differ by roughly one order of magnitude as in Fig. 4 whereas the coefficients to the quadratic dependence are equal. While the measured execution time has not be resolved into components corresponding to different parts of the code, the observations on the coefficients may be identified with code features: The quadratic dependence from the task of determining the existing bonds via a double loop over the atoms in the search for atom pairs that are separated by less than a user specified bond length, and the linear dependence from the task of rendering the bonds. The limit on the number of atoms for interactive display of the bonds may be seen to be ~3,000.

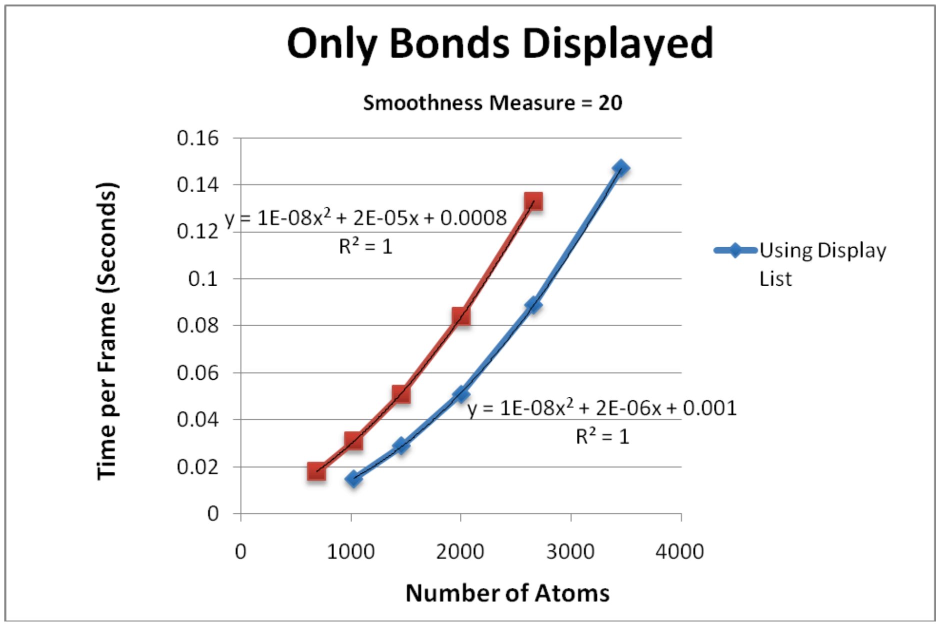

Fig 5. Variation in the execution time for the display of bonds with the number of atoms. Using display lists increases the execution speed. However the quadratic dependence limits the number of atoms that can be handled interactively to much smaller values as compared to when only atoms are displayed (Fig. 4).

Conclusion

As suggested in [2] stereographic visualization of atomistic configurations is enhanced by rendering bonds as cylinders rather than as lines. As compared to the cases without the use of display lists [2], the use increases limits on the number of atoms for interactive rendering of atoms alone or bonds alone to ~30,000 and ~3,000 atoms respectively. As suggested in [2] the display of bonds can be made more efficient by using linked lists and neighbor lists.

Acknowledgments

This work was funded by the Louisiana Board of Regents, through LASIGMA [Award Nos. EPS-1003897, NSF (2010-15)-RII-SUBR, and HRD-1002541]. One of the authors (Cole

Vanderlick) thanks Dr. Diola Bagayoko for the opportunity to conduct research as an REU under LA-SiGMA.

References

1. S. Kodiyalam, M. Benissan, S. Akwaboa, P. Mensah, A. Jana, and D. Bagayoko, "Molecular Dynamics Simulation and Visualization of Thermal Barrier Coatings," Proceedings of the 2012 RII LA-SiGMA Symposium, July 23rd, Baton Rouge, Louisiana.
2. G.R. Wright, S. Kodiyalam, A. Jana, "Stereographic Visualization of Molecular Configurations in a CAVE" LA-SiGMA 2011 REU Report.
3. Figure from http.cs.uic.edu/~kenyon/conference/GILKY/CAVE_DOD.html
4. OpenGL http://alien.dowling.edu/~vassil/thebluebook/