Variable Resolution Decreases Display Times in Stereographic Visualization of Atomistic Configurations

Robert Stewart[1], Sanjay Kodiyalam[2], Amitava Jana[2]

[1]Department of Physics, The University of Akron

[2]Department of Mechanical Engineering, Southern University Baton Rouge

LA-SiGMA
Louisiana Alliance for Simulation-Guided Materials Applications

Abstract

- Visualizations of atomistic configurations that are produced using the CAVE library, OpenGL software, and the C/C++ programming language are further optimized by utilizing an additional processor and implementing variable resolution.
- Benchmarking the original version of the code on a 2xDual core, 3.22 GHz, Opteron when visualizing a BCC lattice with all 8-bonded atoms and considering the limit of interactive frame rate to be 10 frames per second, the maximum number of displayable atoms is ~10,000.
- The computations used to generate the bond list are moved to a previously unburdened processor and the resolution of each atom and bond is changed to vary with visual angle, increasing the limit of displayable atoms to ~14,000.
- When compared to minimum resolution, the current variable resolution method takes ~10% more CPU time to render the same number of atoms. This suggests Further reduction of the resolution will not significantly decrease the CPU time of the program, and may reduce visual quality.
- The display process is still limiting the total CPU time required to run the program , leaving room in the main process to increase the program's functionality

Introduction [1]

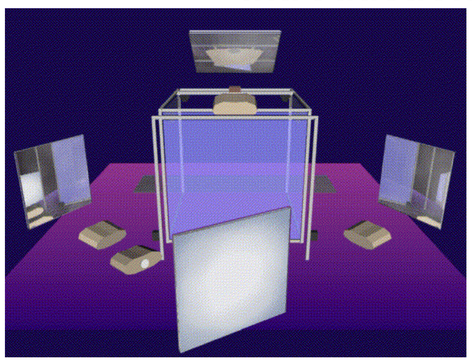


Fig 1. Schematic of the CAVE at Southern's College of Engineering. (Figure 1 from http.cs.uic.edu/~kenyon/conference/GILKY/CAVE_DOD.html)

- The CAVE at Southern's College of Engineering (CoE) (Fig. 1) is where this work was carried out.
- The CAVE is an 8 ft x 8 ft x 8 ft space with 4 displays (3 displays on screen-walls and 1 display on the floor).
- Active stereo with eye ware synchronized to separate left & right eye images displayed in rapid sequence allow the user to see in stereo.

[1] C. Vanderlick, S. Kodiyalam, A. Jana, "Optimizing Stereographic Visualization of Atomistic Configurations", LA-SiGMA REU Paper, 2012.

Methodology

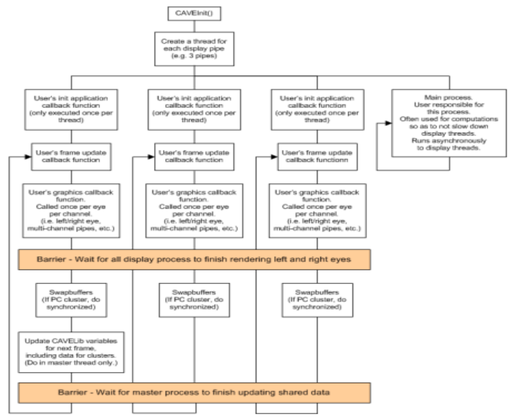


Fig 2. Flow of a CAVELib application (figure 2 from CAVELib User and Reference Guide, December 2007 edition)

- Computations are separated from display routines in display pipe threads.
- The main process performs all computations and stores the data in shared memory which is visible to display threads
- While the display threads are reading the data, the main process generates the next set of data.

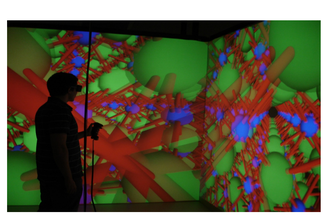
Main Process

```
while (!CAVEgetbutton(CAVE_ESCKEY))
    {
        /* Nap in the while loop if not doing any processing,
        cuts down on needless looping */
        lpbegin=clock();

        if(readingBondList2)
        {
            CAVESetWriteLock(CLDLd);
            //cout << "in reading 2" << endl;
            MakeLinkedList();
            MakeBondData1();

            CAVEUnsetWriteLock(CLDLd);
            readingBondList2 = FALSE;
        }

        if(readingBondList1)
        {
            CAVESetWriteLock(CLDLd2);
            //cout << "in reading 1" << endl;
            MakeLinkedList();
            MakeBondData2();

            CAVEUnsetWriteLock(CLDLd2);
            readingBondList1 = FALSE;
        }


        lpend=clock();
        time = (double) (lpbegin-lpend)/CLOCKS_PER_SEC;
        if(time < waittime) {tms = (float) (waittime- time)*1.0e6; CAVEUSleep(tms);}

    }
```

Display Process

```
void ReadBondData1(void) //Used the liked lists made in MakeLinkedList()
{

        glPushMatrix();

        int bondNumber = 0;
        int cui;
        cui = CAVEuniqueIndex();

        for (int atomNumber=0;atomNumber<NAtoms;atomNumber++)
        {

                glPushMatrix();
                glTranslated(xList[atomNumber],yList[atomNumber],zList[atomNumber]);


                                for (int j=0;j<bondsOfAtom[atomNumber];j++)
                                {

                                        glPushMatrix();

                                        glRotated(angleData[bondNumber],-
                                        yList[bondNumber],xList[bondNumber],0.0);
                                        glScaled(1.0,1.0,heightData[bondNumber]);
                                        glCallList(DLs[cui][Natyp*0]);

                                        glPopMatrix();

                                        bondNumber++;

                                }
                glPopMatrix();


                        }
        glPopMatrix();
}
```

- The visual angle of each atom and bond is calculated using the distance between the observer and the object .The diameter of the object is used as its size.
- Each object is assigned a resolution based on its visual angle. The maximum resolution is set at 400 primitives for atoms and 20 primitives for bonds. With a minimum resolution of 9 primitives for atoms and 3 primitives for bonds.
- The rules for resolution assignment are developed by running the program in the CAVE environment to determine how much the resolution can be lowered without a significant drop in visual quality.



Above: Testing resolution in the CAVE

Results

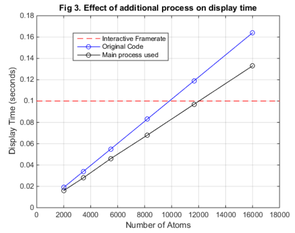


Fig. 3 Comparison between making the bond list within the display process(solid blue line) and shifting the bond list calculations to the main process (solid black line).

- Moving the bond list into the main process decreases total CPU time by ~20%

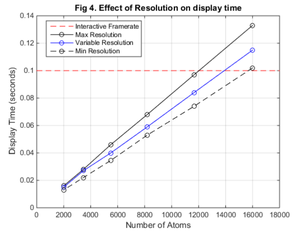


Fig. 4 Display times are compared for all atoms and bonds at maximum resolution (solid black line) minimum resolution (dotted black line) , and variable resolution (solid blue line).

All times in Fig. 4 taken with the bond calculations shifted to the main process.

- Current variable resolution method takes ~10% more CPU time than minimum resolution.

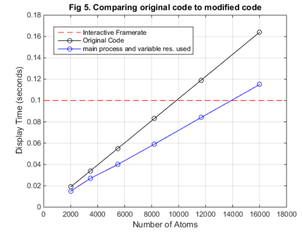
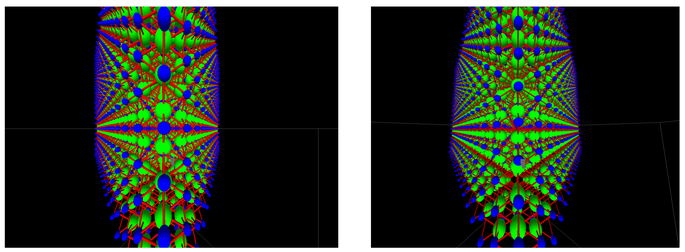


Fig. 5 Difference in display times for original code (solid black line) and the modified code which uses the main process and variable resolution (solid blue line).

- Displayable atoms increased from ~10,000 to ~14,000.



8-Bonded BCC Lattice, Left: Full resolution, Right: Variable Resolution

Conclusions and Future Work

- Implementing variable resolution and moving bond calculations to the main process increases the number of displayable atoms from ~10,000 to ~14,000
- The main process is not limiting the total CPU time required to run the program, leaving room to increase functionality.
- Additional features to increase program functionality: transparent barriers, periodic boundaries, and atomic stress visualizations.

Acknowledgements

This material is based upon work supported by the National Science Foundation under the NSF EPSCoR Cooperative Agreement No. EPS-1003897 with additional support from the Louisiana Board of Regents.