11-2012

VISUALIZATION OF MOLECULAR DYNAMICS SIMULATION DATA

Suroj Shrestha

VISUALIZATION OF MOLECULAR DYNAMICS SIMULATION DATA

by

Suroj Shrestha

Undergraduate honors thesis under the direction of

Dr. Bijaya B. Karki

School of Electrical Engineering and Computer Science

Submitted to the LSU Honors College in partial fulfillment of
the Upper Division Honors Program.

November, 2012

Louisiana State University
& Agricultural and Mechanical College
Baton Rouge, Louisiana

CONTENTS

ABSTRACT

The research involves the visualization of molecular dynamics simulation data. A visualization application has been designed to compute and render some properties like mean square displacements and diffusion coefficients. The software has been further improved by making it more user-friendly and adding more interactive features to it. The application has also been designed in a way such that it can easily be integrated to AtomViz - a space-time multi-resolution atomistic visualization system.

1. INTRODUCTION

Molecular dynamics is a form of computer simulation in which atoms and molecules are allowed to interact for a period of time by initially defining a group of properties that represent atomic positions, atomic types and constraints [1]. The simulation requires inter-atomic forces which are determined by quantum mechanics, which basically deals with electrons, but can be formulated on interaction between atoms. Once we know about the forces, solving Newton equation will give atomic trajectories, i.e. position of atoms as a function of time:

$$\vec{F}_i = m_i \vec{a}_i \qquad\qquad \vec{a} = \frac{d^2 \vec{r}_i(t)}{dt^2}$$

Here, $\vec{F}$, m, $\vec{a}$ and t represents force, mass, acceleration and time. $\vec{r_i}(t)$ represents the atomic trajectories, where i ranges from 1 to n, the number of atoms. This atomic trajectory data can be used to calculate and analyze different properties of the system.

As an example, the simulation data of a silicate liquid, Anorthite ($CaAl_2Si_2O_8$), which consists of time-varying 3D atomic positions, is used. The mean square displacement and diffusion coefficient of the system is computed and line graphs representing the computed result are rendered. A mechanism is provided for user interaction in order to filter data based on user's choice.

In future, the visualization application developed in this project is expected to be released as a feature in AtomViz. AtomViz is atomistic visualization software which provides a range of features by integrating various analyses and rendering tasks together to support interactive visualization of data at space-time multi-resolution [1]. The features of AtomViz are grouped into five different modules each of which has wide array of options that the user can use/edit to render different properties for analysis.

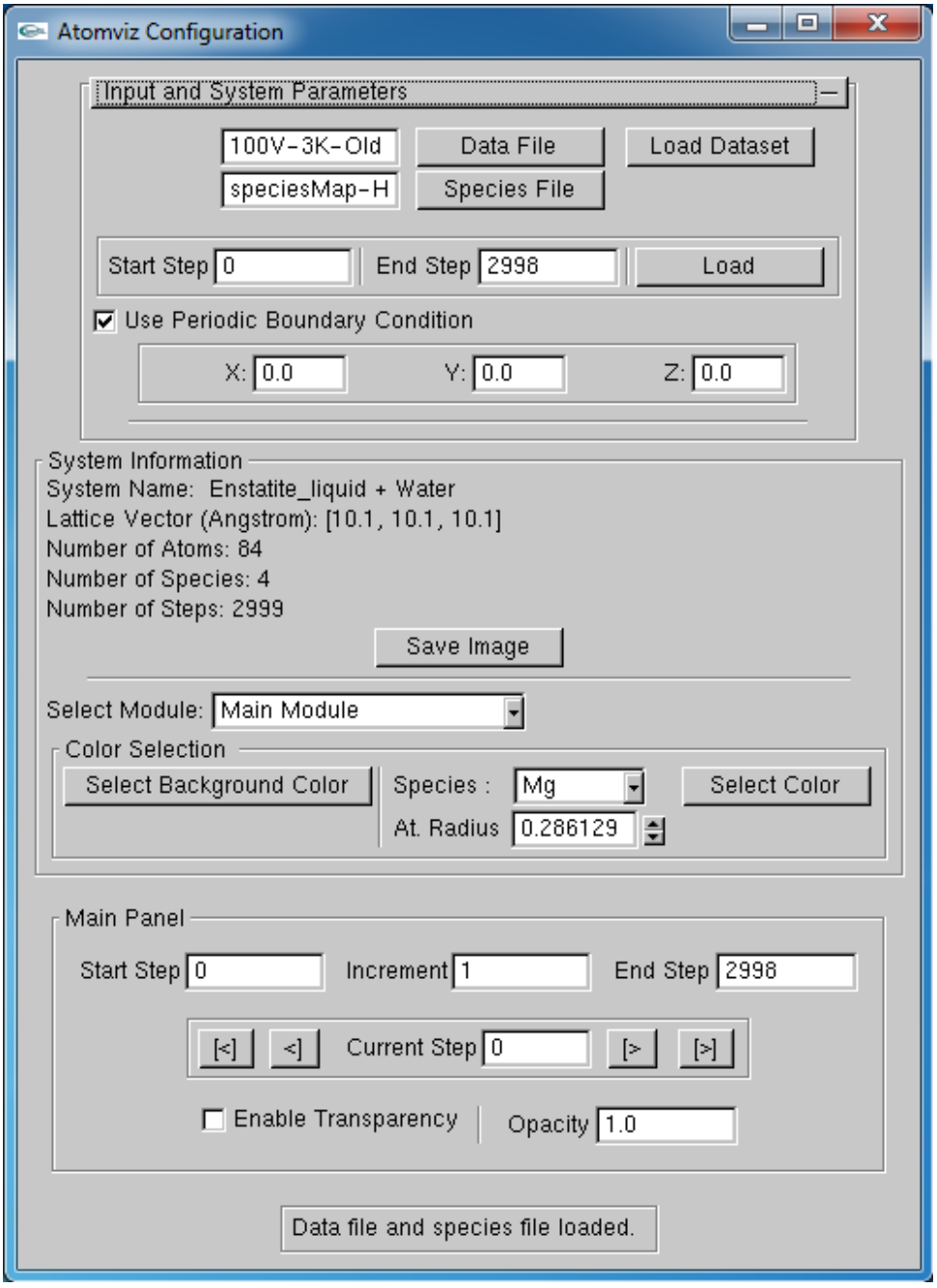


Fig. 1 AtomViz Configuration window

2. INPUT DATA

Two types of input data were used:

a) Simulation Data

The simulation data are position-time series representing snapshots of 3D atomic configuration. Along with the simulation data, the data file consists of information about the given atomic system. The first line denotes the number of snapshots while the second line denotes the number of atoms in the system. The third line represents the dimensions of simulation box. The fourth line contains the temperature information. The name of the system is on the sixth line. The line containing "Konfig= X" represents the start of a new timestamp block which follows directly on the next line. X represents the number of snapshot and it ranges from 1 to the number of total snapshots (i.e. the number on the first line).

```
39600
 208 208   1
  0.1454916E+02  0.1446440E-08  0.1446440E-08  0.1446440E-08  0.1000000E-14
   4000.00000000000
  CAR
 Anorthite (CaAl2Si2O8)
 Konfig=          1
   0.00772205  0.42366096  0.60483548
   0.00445881  0.91288983  0.07879707
   0.26240530  0.52077847  0.08008419
   0.89410668  0.48514116  0.37186790
   0.01346595  0.85065288  0.76645434
   0.68337465  0.46817682  0.03252640
   0.88042977  0.26926252  0.86689496
   0.25580762  0.55684758  0.84674008
   0.53692309  0.43823347  0.61348018
   0.57608560  0.85025025  0.70189742
   0.91502497  0.92987331  0.25239519
   0.28901775  0.19453286  0.33449322
   0.33981783  0.92397990  0.81659920
   0.37159425  0.70115370  0.68572105
   0.10258614  0.01388086  0.41862337
   0.38253367  0.97830475  0.27793212
   0.61911734  0.17068544  0.07325418
   0.27142447  0.33330719  0.11517901
   0.27049039  0.02180024  0.01586586
   0.80734928  0.07863083  0.13211576
   0.40445485  0.76091987  0.05114227
   0.42793230  0.16554722  0.69583700
   0.15594996  0.63185453  0.41525251
```

Fig. 2 Simulation data file

b) Species Data

Species Data give information about different atomic species that make up the volcanic liquid and how they correspond to the simulation data. It consists of atomic name, atomic weight, starting index and ending index of the species involved.

```
#Species
#AtomicName AtomicWeight StartIndex EndIndex
Ca 20 0 15
Al 13 16 47
Si 14 48 79
O 8 80 207
```

Fig. 3 Species data file

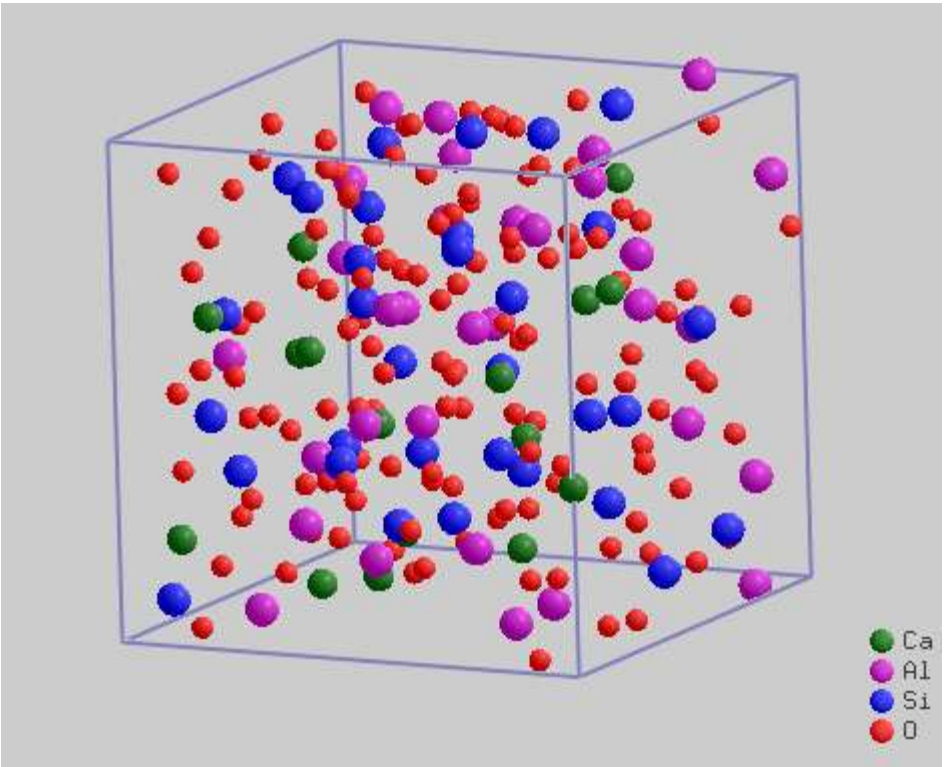


Fig. 4 A snapshot of atoms in a simulation box generated by rendering raw data using AtomViz

3. COMPUTATION

With the given data two quantities are calculated:

a) Mean square displacement (MSD) determines the extent of motion of particles. MSD for atom *i* is calculated with respect to some reference position. This position is taken at time *t = 0*. MSD of an atom *i* at time *t* is given by:

$$MSD_i(t) = \left[\vec{r}_i(t + t_0) - \vec{r}_i(t_0) \right]^2$$

where $\vec{r_i}(t)$ gives the position of atom *i* at time *t*.

The partial MSD is calculated by averaging over atoms of a given species [2]:

$$MSD_\alpha = \frac{1}{N_\alpha} \sum_{i=1}^{N_\alpha} \left[| \vec{r}_i(t + t_0) - \vec{r}_i(t_0) | \right]^2$$

where N_α is the number of atoms of species α.

To get better statistics, MSD can be averaged over multiple time origins.

b) Diffusion coefficient (D) is the rate at which a diffusing substance is transported between opposite faces of a unit cube of a system when there is unit concentration difference between them [3]. D is calculated as a function of time and it explains how the atoms travel over time. In reality, D is calculated in a long time interval in order to obtain a realistic data. D can be calculated using following formula [4]:

$$D_i(t) = \frac{MSD_i}{6t}$$

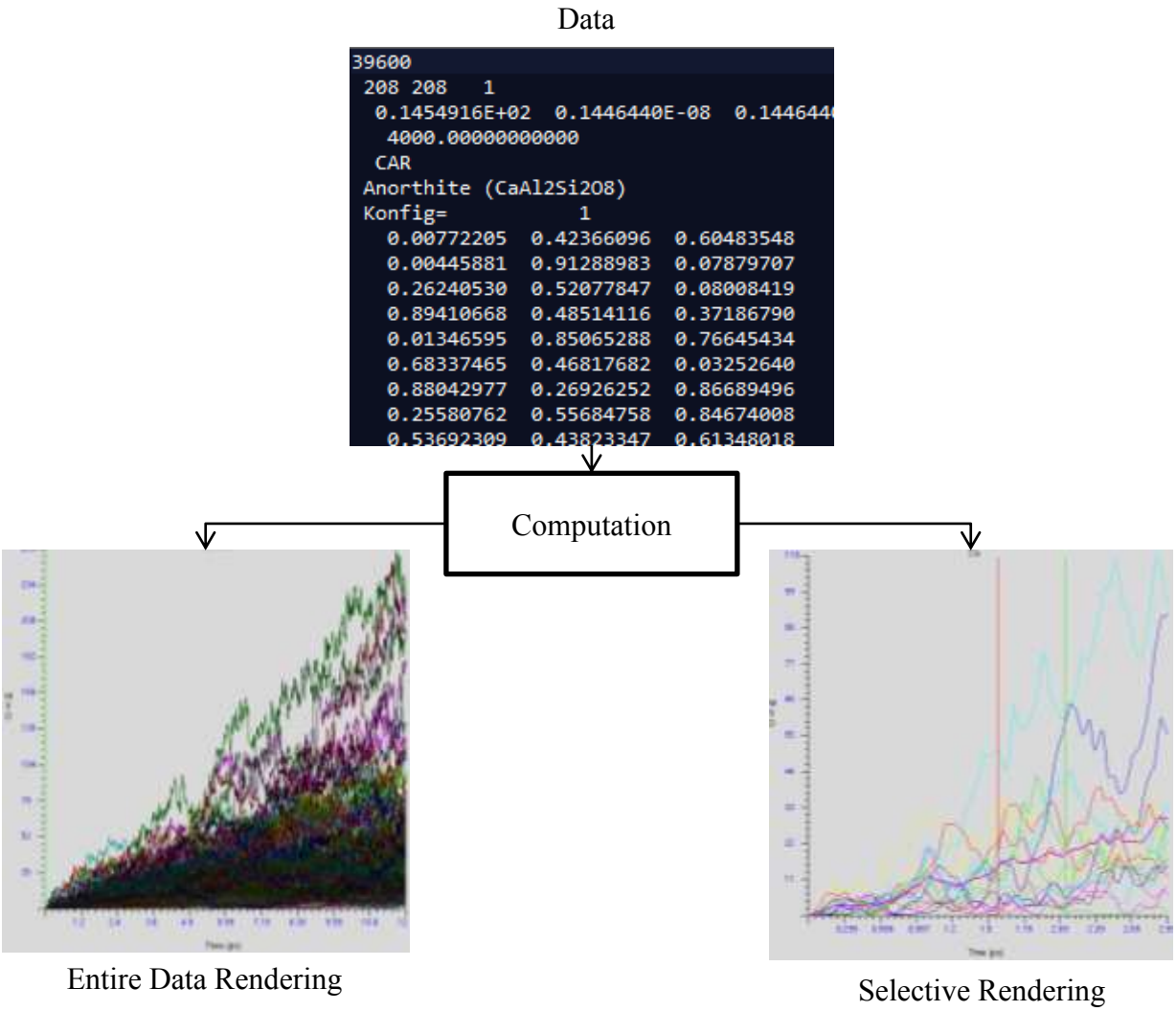
The idea is to start calculating these quantities for individual atoms and then average it out to find the mean quantities for species and finally calculate a system wide average.

4. VISUALIZATION

Visualization is a process of making a computer image or graph for gaining an insight into data or information. The visualization application developed in this project deals with rendering of line graphs that represents mean square deviation and diffusion coefficient of the system.

The problem with line graphs of the system with large number of atoms is that the graphs become too crowded when we try to visualize whole system or group of atoms of a particular species. Using MS Excel's chart tools to produce graphs is an option but the process becomes tedious when data has to be transferred every time from the output of a simulation to the spreadsheets. Selecting and displaying various portions of graph in Excel can also be hectic for a large system. Also there is a constant need of transporting data from the line graph to other visualization system (such as AtomViz) for further analysis.

Considering the aforementioned problems, we have developed an interactive visualization system that lets user to filter and render the line graphs on the fly. It can be integrated to AtomViz to ease further analysis of data extracted from the graph.



5. DESIGN AND IMPLEMENTATION

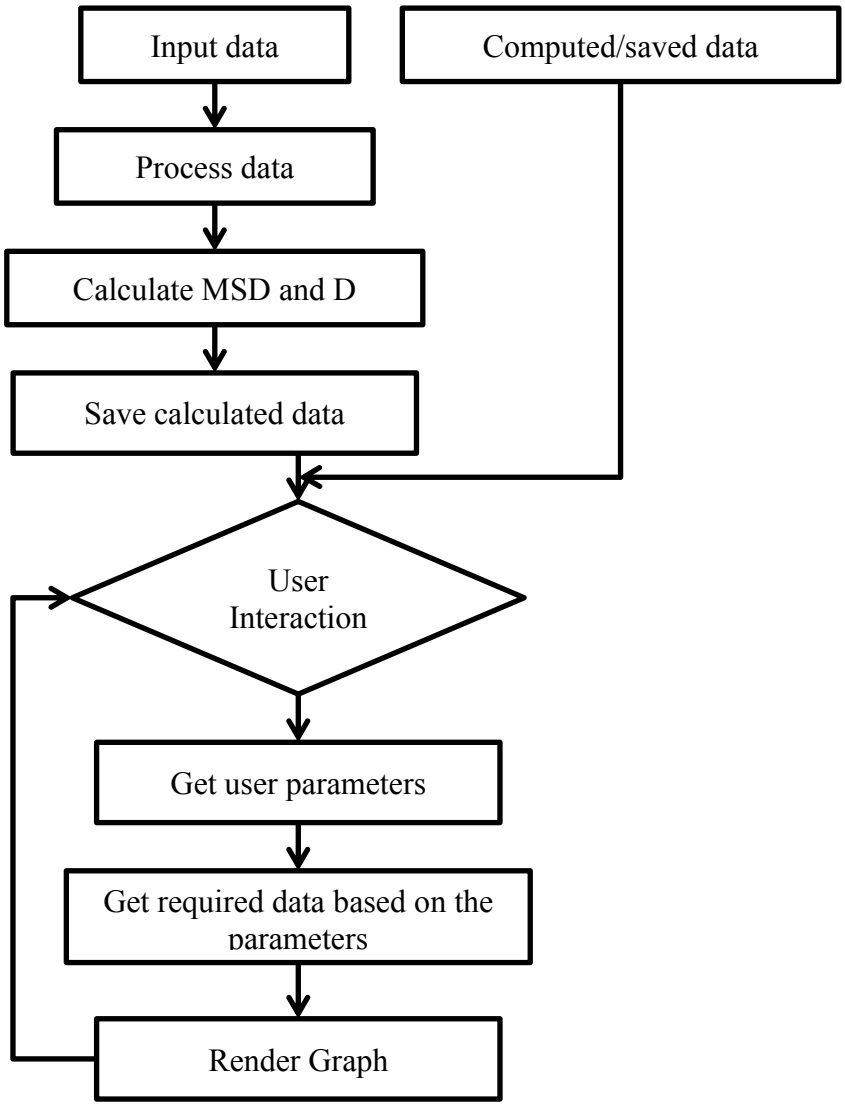


Fig. 5 Basic flow of visualization process

The visualization system was implemented using C++ programming language. The input data is stored in a 2-dimensional array. The data were then processed to remove periodicity. MSD and D were calculated and stored in a vector of structures that contained species index (spInd:int), time stamp (time:float), mean square displacement at the corresponding timestamp(msd:float) and diffusion coefficient (dc:float). If the user chooses to use a saved file, the data is read and stored on the same vector. Based on the selection of the user, data is filtered from this vector and passed to the chart class for rendering.

The chart class reads the data and creates graph/chart and color codes them as necessary.

The application uses OpenGL for rendering. OpenGL was the most appropriate tool as it is open source, widely used, cross-platform compatible and best documented graphics library.

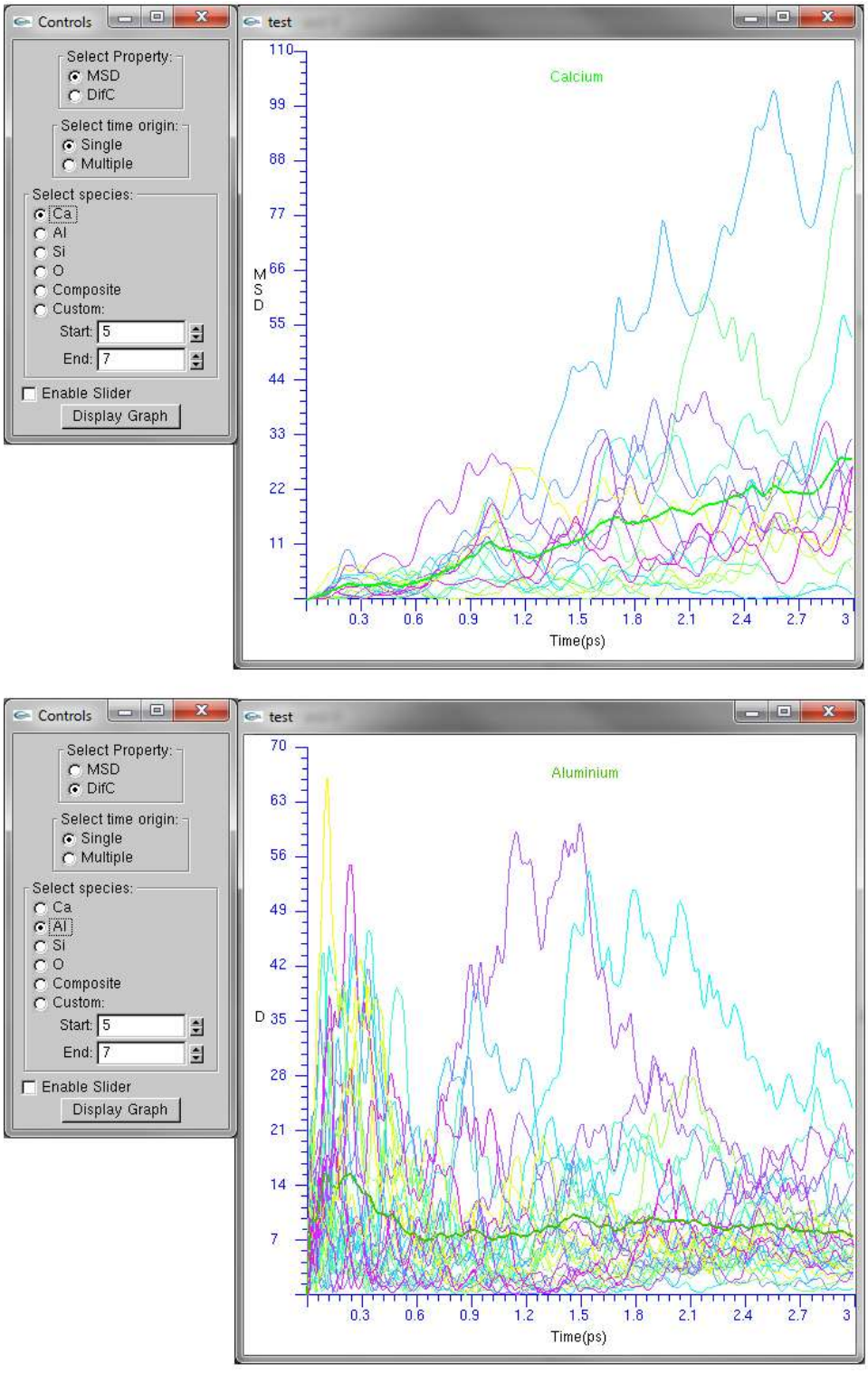


Fig. 6 Instances of visualization application in action

Features

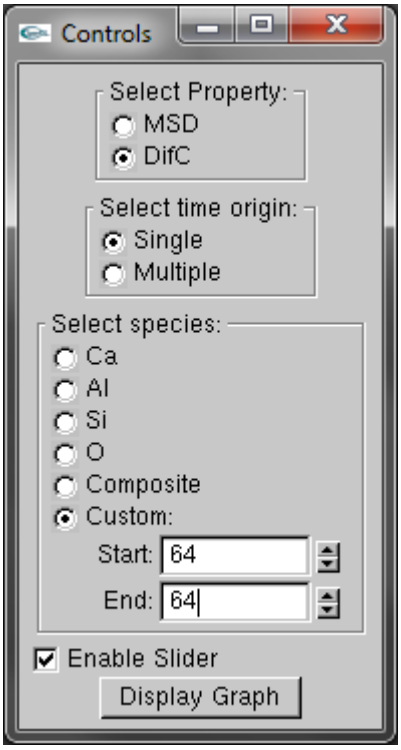


Fig. 7 Control window showing various options

a. Modular Design and Flexibility

The application consists of independent classes to handle data and render graphs. The DataMan class deals with reading/processing data, computing the dynamic properties and saving the results into a file. Various accessor functions are provided to retrieve specific sets of data based on user selection. The filtered data is then passed to Chart class that deals with all the aspects of drawing line graphs. The chart class also provides sliders to let user to select time ranges. Though the application was designed with Anorthite system's simulation data in mind, it is flexible enough to be used for other systems as well.

b. Varying color codes

Line graphs are drawn using varying color to differentiate from each other. Colors are generated based on the number of curves to be drawn. Each color components (red, green and blue) are varied independently and gradually to generate different color for each line.

```
void Chart::GenerateColors(int num){
	color.clear();
	float x = 1.0f;
	int i = 0;
	Chart::lineColor currColor;
	while (i < numAtoms+numSpecies+6){
		currColor.R = 0.0f; currColor.G = 0.0f; currColor.B = x;
		color.push_back(currColor);
		currColor.R = 0.0f; currColor.G = x; currColor.B = 0.0f;
		color.push_back(currColor);
		currColor.R = x; currColor.G = 0.0f; currColor.B = 0.0f;
		color.push_back(currColor);
		currColor.R = 0.0f; currColor.G = x; currColor.B = x;
		color.push_back(currColor);
		currColor.R = x; currColor.G = 0.0f; currColor.B = x;
		color.push_back(currColor);
		currColor.R = x; currColor.G = x; currColor.B = 0.0f;
		color.push_back(currColor);
		i+=6;
		x = x*19/20;
	}
}
```

c. Varying Line Thickness

Lines of different thicknesses are used in order to differentiate the data being graphed. For example, the lines of average graphs are thicker than other lines.

```
void Chart::RenderLines(){
	int currSpInd = data[0].spInd;
	int colorInd = 0;
	glColor3f (color[colorInd].R, color[colorInd].G, color[colorInd].B);
	if (currSpInd >= 208) glLineWidth(2.0);
	glBegin(GL_LINE_STRIP);
	{
		for(int i = 0; i < data.size(); i++){
			if (data[i].spInd != currSpInd){
				currSpInd = data[i].spInd;
				glEnd();
				if (currSpInd >= 208) glLineWidth(2.0);
				if (currSpInd == 212) glLineWidth(3.0);
				colorInd++;
				glColor3f (color[colorInd].R, color[colorInd].G,
					color[colorInd].B);
				glBegin(GL_LINE_STRIP);
			}
			glVertex2f(float(2 * data[i].time) / (2*maxX/1.8) - 1 +
			 .2, 2 *  data[i].value /(2*maxY/1.8) - 1 +.2);

		}
	}
	glEnd();
}
```

d. Properties selection

Users have an option to select properties i.e. Mean Square Displacement (MSD) or Diffusion Coefficient (DC).

e. Filtering Options

A user can either select species-level filtering by clicking the appropriate radio buttons or enter the range of atom indices to show the respective graphs.

f. Line Graph Detection

A user can click on the line graph in rendering window to view the atom index of the associated atom. If there are overlapping lines, it gives a list of all the atom indices who graph is present under the clicked area on the rendering widow.

```
list<int> Chart::FindAtomIndex(float pointX, float pointY){
	float errorX = 0.05;
	float errorY = 0.1;
	list<int> atomIndList;
	for(int i = 0; i < data.size(); i++){
		if(data[i].time >= pointX-errorX && data[i].time <=
		  pointX+errorX){
			if(data[i].value >= pointY-errorY && data[i].value <=
		  pointY+errorY){
				atomIndList.push_back(data[i].spInd);
			}
		}
	}
	return atomIndList;
}
```

g. Saving and loading computed result

Computation of properties from the simulation files are time consuming. So, in order to avoid the redundant computation, the computed data is saved for future use. Users have an option to load the saved data for rendering. This greatly reduces overall rendering time.

h. Time range selection

When enabled, sliders are provided on the graph to select exact time ranges in the important areas of the graph. These time ranges can be used in AtomViz for further analysis.

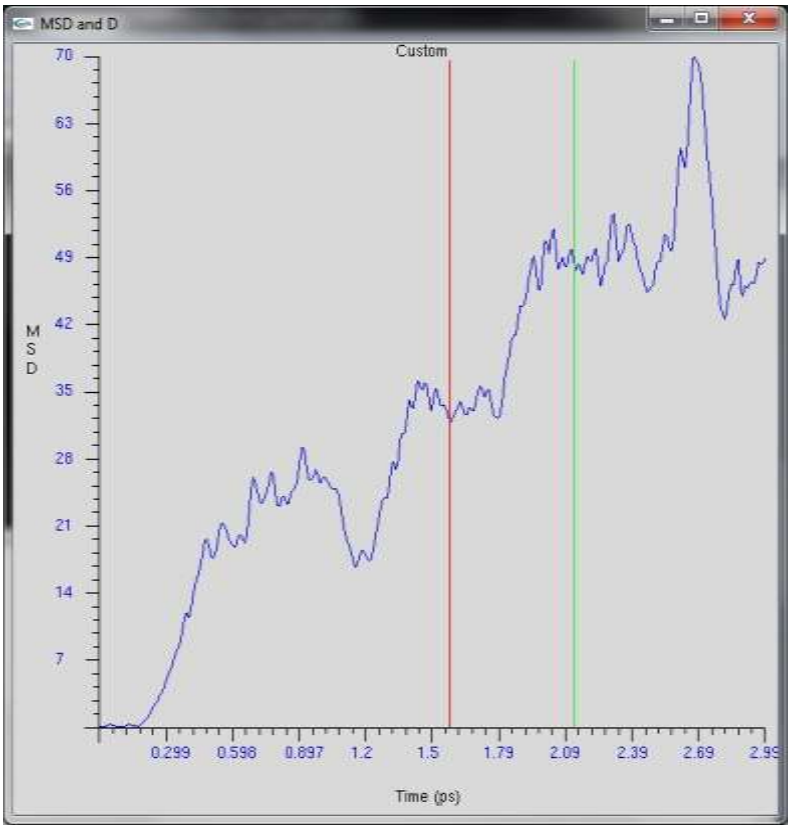


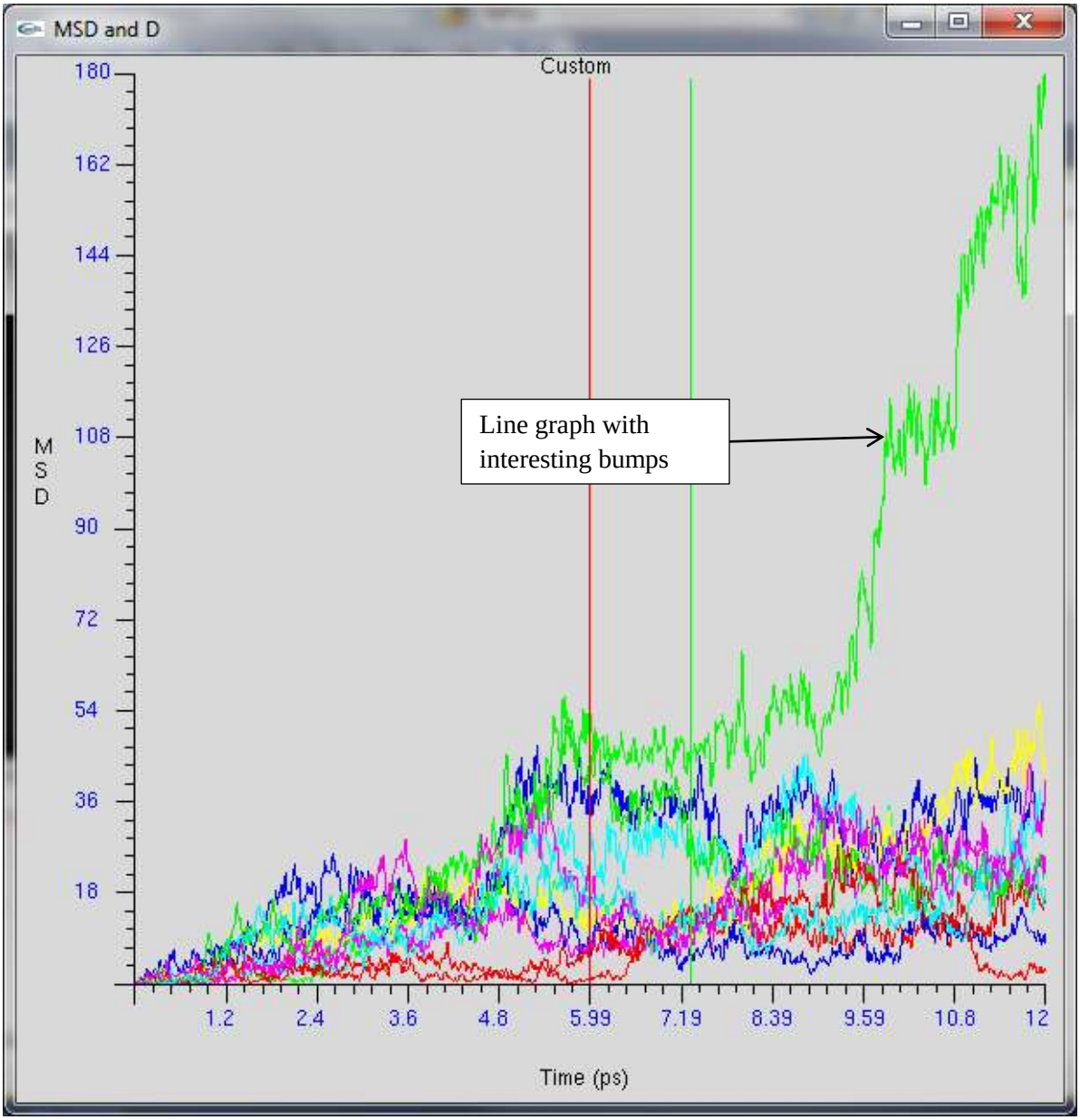
Fig. 8 Sliders placed on "interesting area" of the graph. The time range corresponding to the sliders can be used in AtomViz for further analysis at atomic level.

6. USAGE EXAMPLE

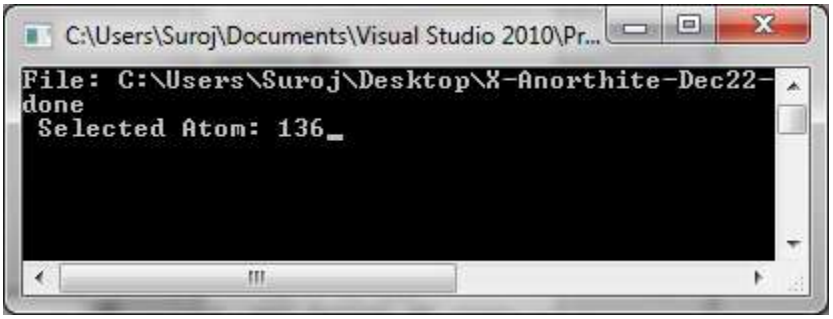
In this section, we demonstrate, in steps, one of the typical flows of the visualization software usage.

Step 1: Simulation and species data are obtained and graphs are generated.

Step 2: A line graph with interesting feature is observed. Since the graph appears in a group with other graphs, the atom index of the particular graph in interest is not immediately apparent:

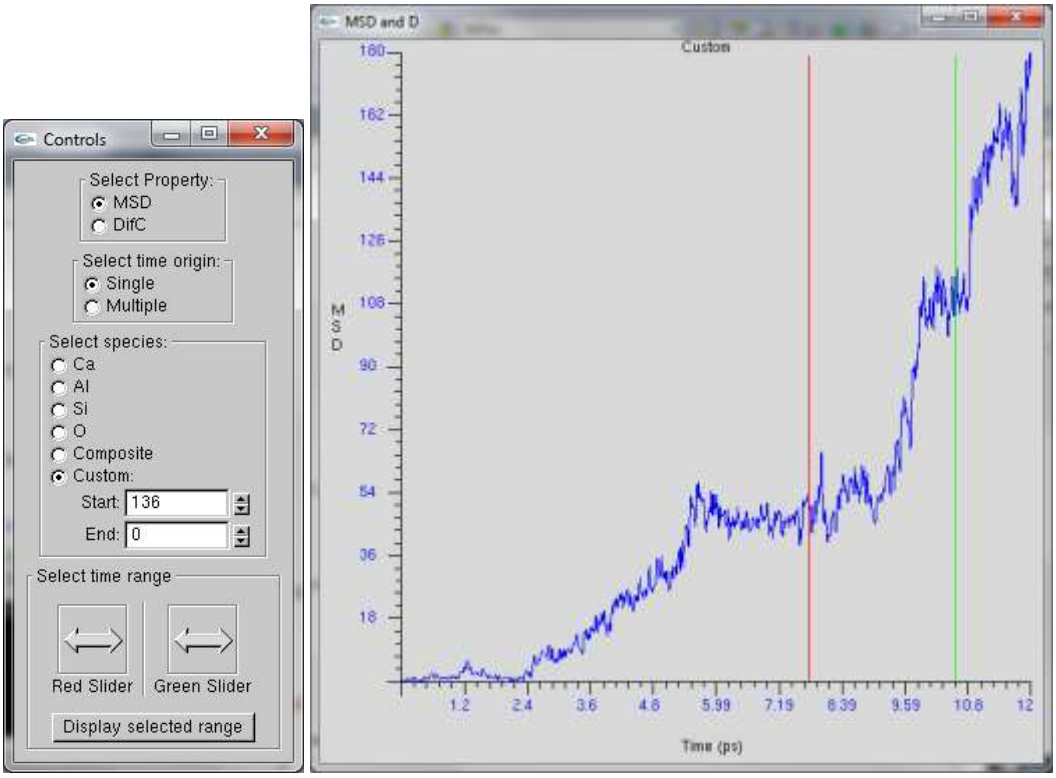


Step 3: Left mouse button on the line graph shows the atom index of the graph in the console:

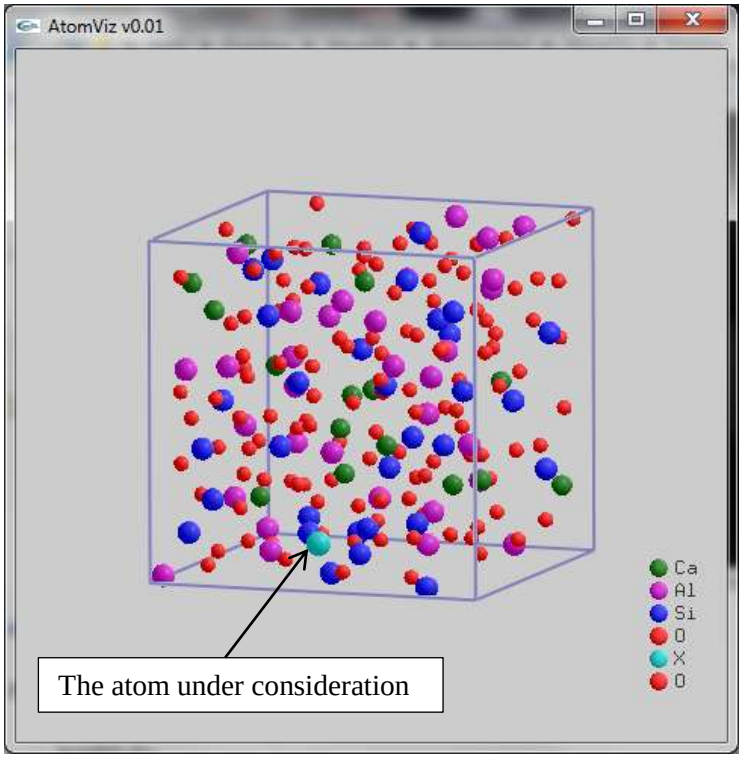


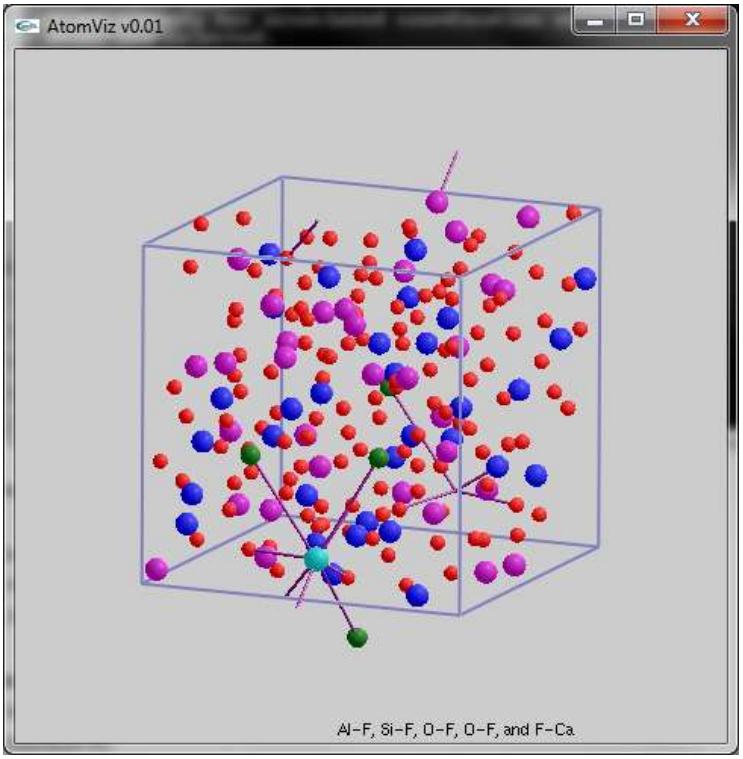
Step 4: The line graph of the detected atom index is isolated for confirmation.

Step 5: Sliders are used to find the time range on the graph:



Step 6: The time range and the atom index are used in AtomViz for further analysis such as viewing the actual movement of the atom in 3D and studying the bonding/coordination with surrounding atoms.





7. OUTPUTS

This section includes some screenshots of the rendering of line graph of Anorthite system with different parameters.

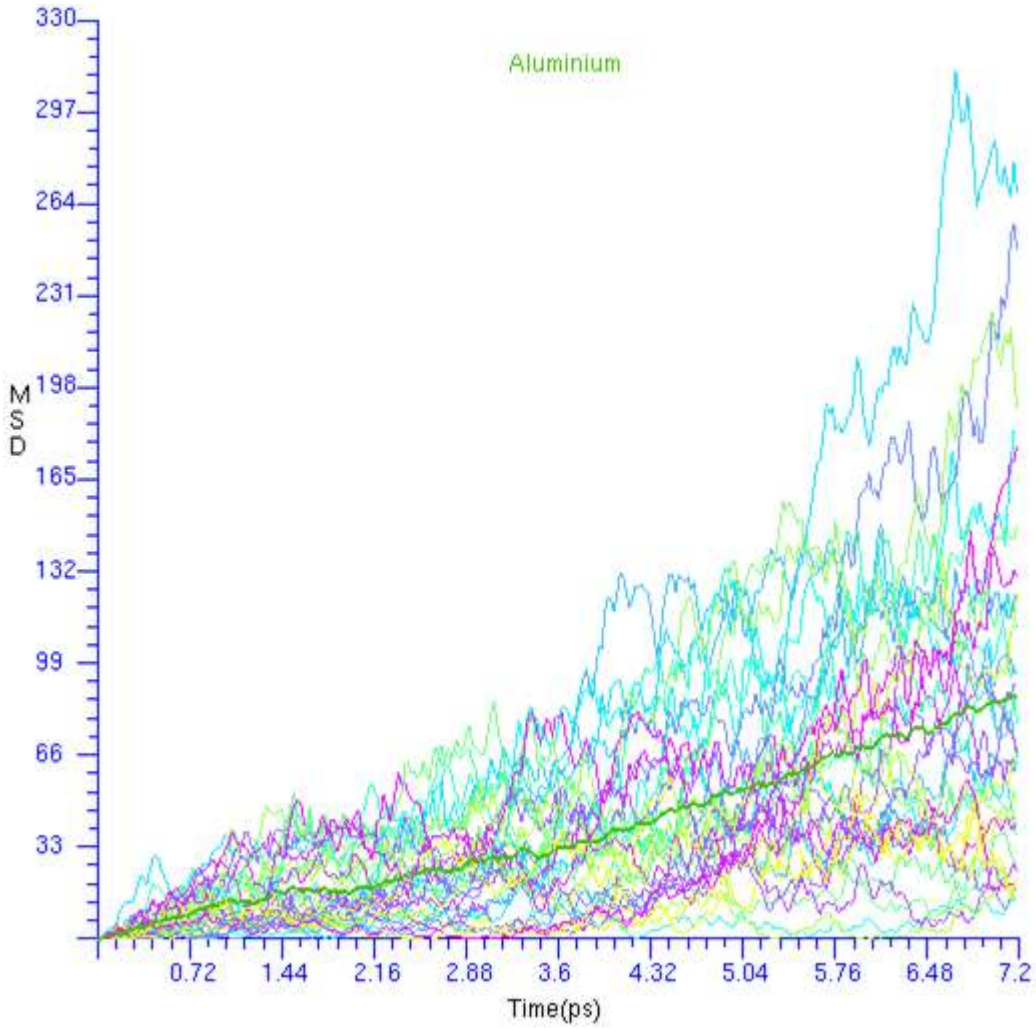


Fig. 9
System: Anorthite
Number of Snapshot: 7200
Temperature: 6000K
Species: Aluminium
Property: Mean Square Displacement
Time Origin: Single

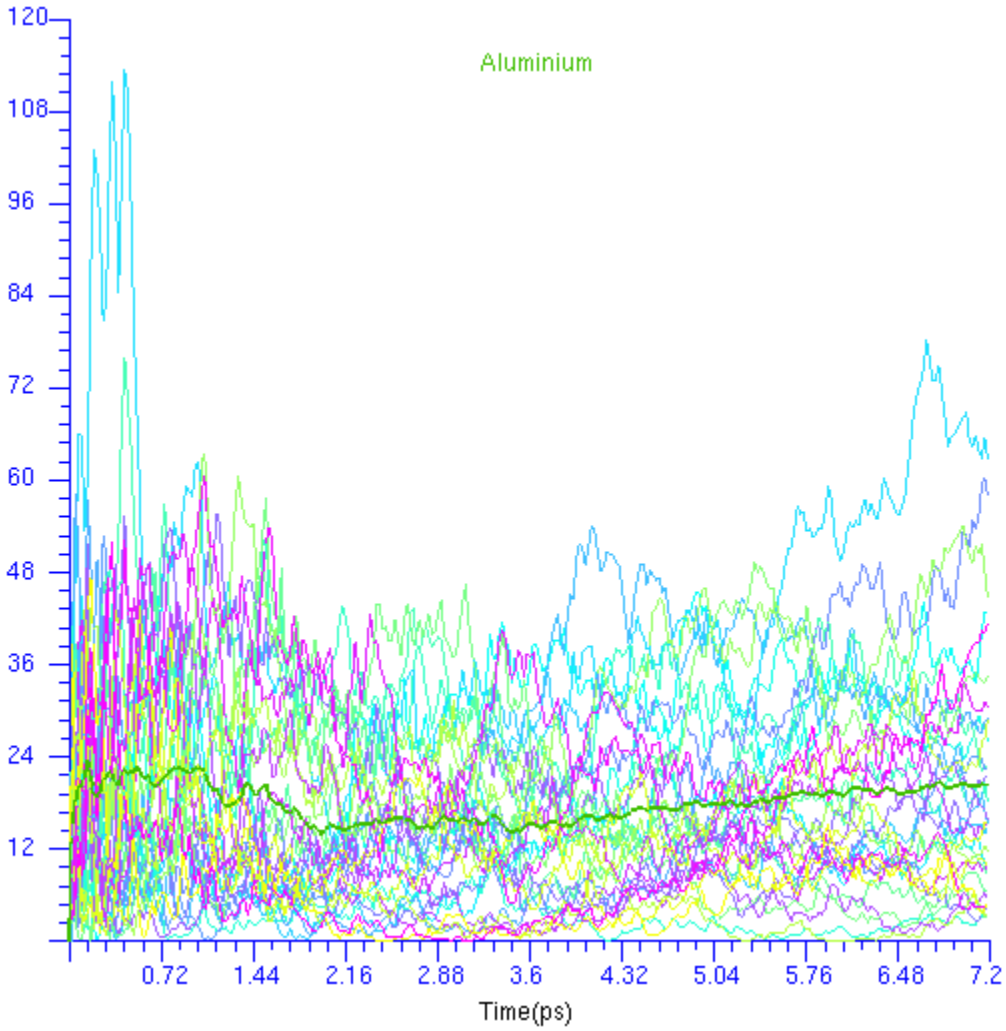


Fig. 10
System: Anorthite
Number of Snapshot: 7200
Temperature: 6000K
Species: Aluminium
Property: Diffusion Coefficient
Time Origin: Single

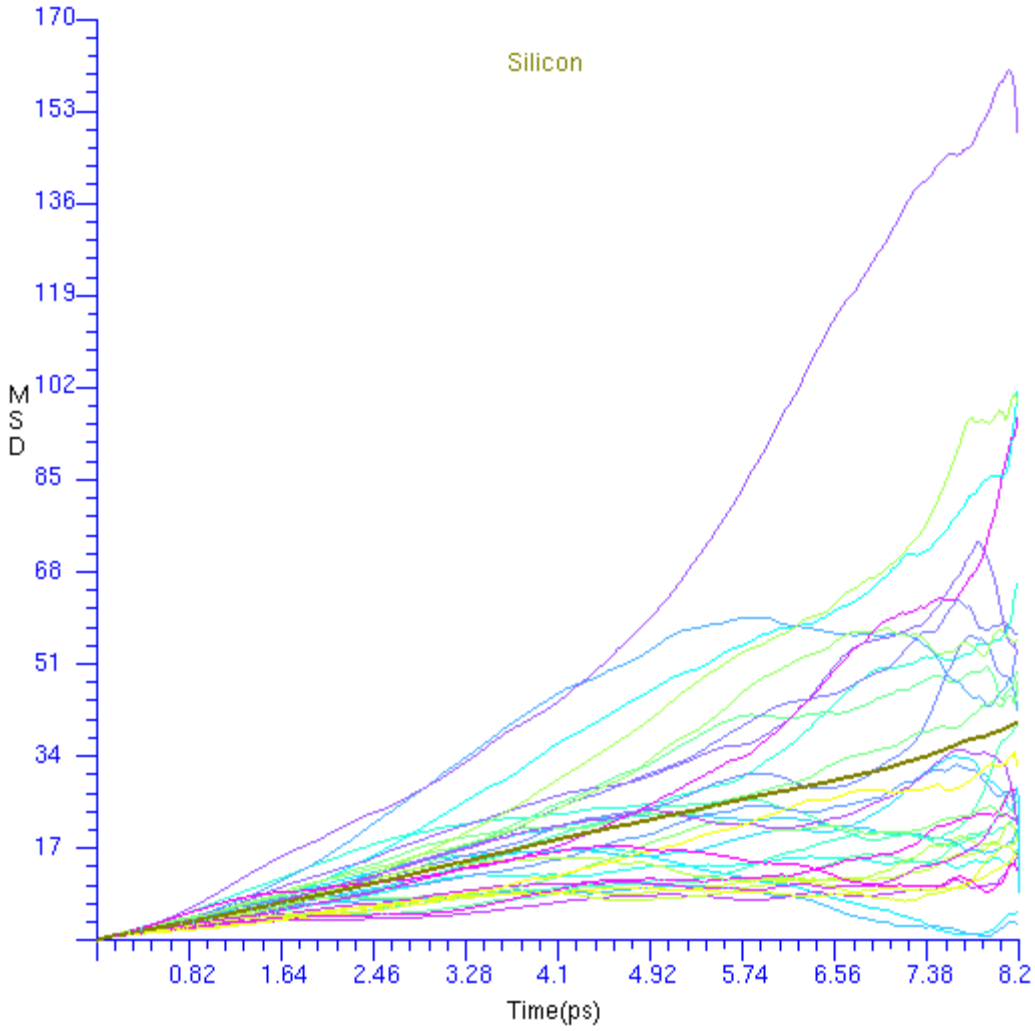


Fig. 11
System: Anorthite
Number of Snapshot: 8200
Temperature: 6000K
Species: Silicon
Property: Mean Square Displacement
Time Origin: Multiple

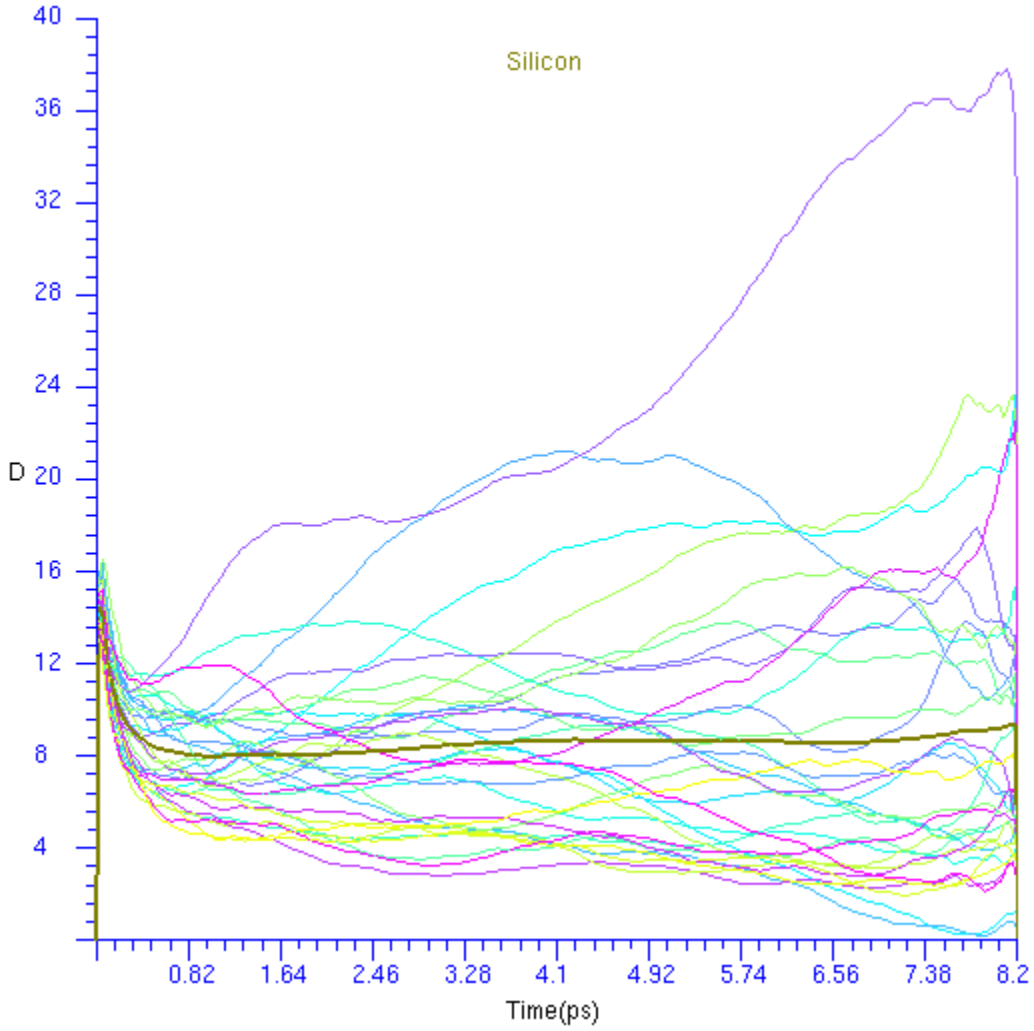


Fig. 12
System: Anorthite
Number of Snapshot: 8200
Temperature: 6000K
Species: Silicon
Property: Diffusion Coefficient
Time Origin: Multiple

8. FUTURE PLAN

The visualization program helps to understand the dynamics of liquids using the molecular dynamics simulation data. It provides on the fly rendering of mean square displacement and diffusion coefficient graphs from the position time series data. However, the program still needs improvement in various aspects. Future plans include:

- A feature to allow user to input multiple simulation files for analysis.
- A feature to detect atoms automatically with different graph-based features.
- Full integration to AtomViz, a space-time multi-resolution atomistic visualization software.

9. REFERENCES

[1] D. Bhattarai, B.B. Karki, Atomistic visualization: space–timemultiresolution integration of data analysis and rendering, 2009

[2] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford University Press, 1987

[3] Collins English Dictionary – Complete and Unabridged. S.v. "diffusion coefficient." Retrieved October 11 2012 from http://www.thefreedictionary.com/diffusion+coefficient

[4] D.C. Rapaport, The Art of Molecular Dynamics Simulation, Cambridge University Press, 1998