12-2003

3D Visualization Modules for Chemical Engineering – A Web-Based Approach Using Java and OpenGL

Sharad Anant Gupta
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Sharad Anant Gupta entitled "3D Visualization Modules for Chemical Engineering – A Web-Based Approach Using Java and OpenGL." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Peter Cummings, Major Professor

We have read this thesis and recommend its acceptance:

Robert Ward, Jian Huang

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Sharad Anant Gupta entitled "3D visualization modules for chemical engineering – A web-based approach using Java and OpenGL ". I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Peter Cummings

Major Professor

We have read this thesis and
recommend its acceptance:

Robert Ward

Jian Huang

Accepted for the Council:

Dr. Anne Mayhew

Vice Provost and
Dean of Graduate Studies

(Original signatures are on file with official student records)

3D visualization modules for chemical engineering - A web-based approach using Java and OpenGL

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Sharad Anant Gupta
December 2003

DEDICATION

I would like to dedicate this thesis to my parents, Anant Gupta and Anita Gupta, and my family for all their love and support over the years. I would also like to dedicate this to my fiancée Akanksha, for her love and encouragement to pursue this goal.

ACKNOWLEDGEMENT

I would like to thank a number of people whose help and guidance have allowed me to reach this milestone. First, I would like to specially thank my thesis advisor, Dr. Peter Cummings for his guidance and endless support. His knowledge and expertise in the areas of Computer Graphics and Chemical Engineering proved invaluable in the completion of this thesis. His patience and faith in me were a constant source of inspiration.

I am grateful to Dr. Clare McCabe for her great support and guidance in the development of the 3D visualization modules and also to Dr. Ariel Chialvo for his support in the development of the molecular simulation module. I would like to sincerely thank Dr. David Kofke, Professor at SUNY-Buffalo, for his co-operation and support in the development of the molecular simulation module.

I would also like to thank my thesis committee members, Dr. Ward and Dr. Huang for taking time to review and direct my thesis work. Lastly, I would acknowledge the support of the NSF grant DUE-9752243, with the help of which this thesis project could be completed successfully.

ABSTRACT

The main objective of this work is to implement web-based educational modules for chemical engineering students. Phase behavior is a topic with which the students seem to struggle with, particularly for mixtures, where a 2-D representation of the phase diagram falls far short of the understanding a 3-D model can provide. Using the platform-independence of Java and the graphics capability of OpenGL, three phase diagram Java applets have been developed. Users can view these web-based 3D applets by installing a plug-in. These modules provide users with an ability to rotate the 3D models, slice through them, zoom into them and view their various 2D projections. Also, a molecular simulation applet for measuring chemical potential of binary mixtures has been developed, using a Java-based molecular simulation application-programming interface (API).

First, the thesis presents a brief overview of phase diagrams and explains why modeling them using computer graphics is useful. While visualization involves the merging of data with the display of geometric objects through computer graphics, it is important to study the software issues involved in web-based visualization. The paper explains the visualization framework by describing the visualization pipeline and then using it as a guideline for the development of the modules.

Next, the paper describes the development of the molecular simulation applet using a molecular simulation API - *Etomica*. The Java applet provides for dynamic modification and interrogation of the simulation, while it is in progress, which enables students to see directly the effect of changing state conditions or molecular interactions on the behavior of the molecules and on the outcome of the simulation.

It is hoped that by using these web-based 3D phase diagrams the chemical engineering students would gain a better understanding of the complicated 3D models, making this package a useful instructional aid. It is also hoped that the molecular simulation applet would be an effective tool to help students understand molecular simulations.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part One – Introduction

1 Introduction

1.1 World Wide Web (WWW) as a medium of education

With the rapid growth of the Internet and the broad acceptance of the WWW as a global medium for disseminating and processing information, the Internet and various web-based technologies have shown a promising future in the field of education [2]. Nowadays, while computers and laptops are being widely used by undergraduate and graduate students, more and more universities are establishing wireless networks for providing easy and convenient access of the Internet [1]. The wide usage of multimedia over the web has opened new vistas in education by taking full advantage of our basic "senses" of learning such as visualizing 3D objects and nonlinear nature of thought processes [4]. The Internet no longer just provides information in the form of static web pages. More and more interactivity has resulted from the wide usage of various web-based technologies.

Today there exist excellent reviews and articles designed to guide professors, particularly new faculty, through the increasingly turbulent waters that are college teaching [1]. A consistent message in these and other publications on teaching effectiveness is that we are in times of change and that the traditional lecture-based delivery of course material needs to be substituted or supplemented with more innovative instructional methods. Many of these studies suggest that active involvement of students in the delivery of course material greatly facilitates learning and retention. A number of studies have illustrated the effectiveness of using technology in the classroom [1].

Various references [2, 3] provide a detailed review about the use of multimedia and web - based visualization as an effective tool of education. Reference [4] describes the benefits of Java, as a programming language for developing web-based educational modules. While the use of Java applets can encourage asynchronous distance learning and thus help overcome the limitations (involving both time and space) inherent in traditional instructional techniques, there are various concepts and phenomena that inherently involve dynamic, multimedia and interactive information. Such information cannot usually be communicated effectively using paper or chalkboards or transparencies. The "mental picture" that a teacher has, is better conveyed to a learner if the information is presented in its appropriate form [4]. References [2]-[4] provide detailed insight into the pedagogical and technological issues involved in web-based educational modules.

Visualization of three-dimensional phase diagrams and molecular simulation are some of the key areas in chemical engineering that would benefit greatly from the use of computer graphics and web-based visualization.

1.2 Objective

The goal of this thesis is to develop four web-based educational modules for chemical engineering students. The modules can be classified into the following two types:

- **Phase diagrams (Type I and II)**

  Thermodynamics and phase behavior is a topic, which undergraduate chemical engineering students seem to struggle with, particularly for mixtures, where a 2-D representation of the phase diagram falls far short of the understanding a 3-D model can provide. The applets will allow the student to *walk* through the phase diagram analyzing the familiar *p-T-x (pressure-temperature-composition)* and *V-T-x (volume-temperature-composition)* surfaces in binary mixtures. The static diagrams drawn in every thermodynamics or physical chemistry textbook can sometimes be difficult to understand and not convey the necessary information to the student. Students presented with simple *p-T (pressure-temperature)* or *p-x-y (pressure-liquid composition-vapor composition) or T-x-y (temperature-liquid composition-vapor composition)* projections of the phase diagram often have difficulty relating to other representations of the inherently three-dimensional *p-V-T-x (pressure-volume-temperature-composition)* surface. A visual tool would enable the student to more easily connect the different representations through manipulation and rotation of the model [1].

- **Molecular simulation module for measuring chemical potential of binary mixtures**

  Using the *Etomica* molecular simulation, a visual development environment developed by Dr. David Kofke at SUNY-Buffalo, the objective is to build the molecular simulation module for measuring chemical potential of binary mixtures. Etomica is a Java application-programming interface (API) that provides the class files for constructing a molecular simulation module. It provides for dynamic modification and interrogation of the simulation, while it is in progress, which enables students to see directly the effect of changing state conditions or molecular interactions on the behavior of the molecules and on the outcome of the simulation. As a result, hypothetical in-class discussions can be replaced with real-time computer experimentation by students. The use of this module will facilitate an active-learning-based component of the molecular modeling course [1].

Part Two - Visualization of Phase Diagrams

The first part of this thesis involves the development of web-based module for the display and exploration of 3D phase diagrams. There are six major types of binary phase diagrams, out of which two web modules for type I and one for type II were developed as part of this thesis.

We begin with a brief overview of phase diagrams and explain why modeling them using computer graphics is useful.

2 Phase Diagrams

2.1 Phase Diagram of Pure Substances

A phase diagram is a graphical way of depicting the interrelationship between pressure, temperature and volume, as well as the equilibrium between the phases (solid, liquid and gas) of fluids. For pure substances, the phase diagram is a surface in 3D Pressure (p), Volume (V) and Temperature (T) space. The Figure 2.1 depicts the projection of the phase diagram of a pure substance onto the 2D *p-T (pressure-temperature)* plane.

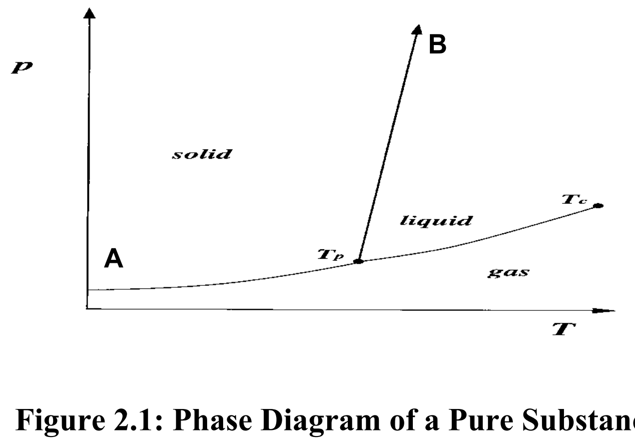


Figure 2.1: Phase Diagram of a Pure Substance

The lines represent the conditions in pressure and temperature along which there is phase equilibrium. For example, from the triple point (T_p) to the critical point (T_c), the line (called the vapor-pressure curve) shows the combination of pressure and temperature along which we have vapor liquid equilibrium (one phase at high density and the other at low density, in equilibrium with each other). One practical everyday consequence of this line is that if one picks a pressure and finds the corresponding temperature on the line between T_p and T_c, that temperature is the boiling point. Note that the boiling point is at lower temperatures at lower pressures – an effect familiar to people living at altitude, since water boils at lower temperatures in such situations. Along the line between T_p and B, we have equilibrium between solid and liquid (the solidification line) and along the line between A and T_p we have equilibrium between solid and gas (the sublimation line).

2.2 Phase Diagram of Binary Mixtures

However, when studying the phase behavior of a binary mixture, a new component, composition dependence (*x*) is introduced [conventionally, if the phase is vapor (or gas) the composition is denoted by (y)]. This is a number between 0 and 1 that represents the mole fraction of one of the substances in the mixture. If N_1 moles of species 1 and N_2 moles of species 2 are present, then mole fraction (X_i) is given by:

$$X_i = \frac{N_i}{N_1 + N_2}$$

Focusing on phase equilibrium, corresponding to the lines in Figure 2.1, for binary mixtures the region of phase equilibrium is enclosed in 3D surfaces. Textbooks usually portray such systems via a projection onto the *p-T (pressure-temperature)* plane along with slices that are parallel to either the *T-x (temperature-composition)* or *p-x (pressure composition)* plane.

Reference [5] provides an excellent overview of phase diagrams. Figure 2.2 depicts the three isothermal *p-x,y* sections (each at fixed temperature) for the system cyclo*hexanone*-cyclo*hexane* at the temperatures 450 K, 500 K, and 550 K. The fine lines represent the dew point curves while the heavy lines represent the bubble point curves. In the *p-x,y* slices, the fluid is liquid above the bubble point line and vapor below the dew point line, and exhibits vapor-liquid equilibrium in the region between the lines. Figure 2.3 shows the three isobaric *T-x,y* sections (each at a fixed pressure) for this system taken from the same region of the phase. In the *T-x,y* slices, vapor exists above the dew point line (fine line) while the liquid exists below the bubble point line (heavy line).

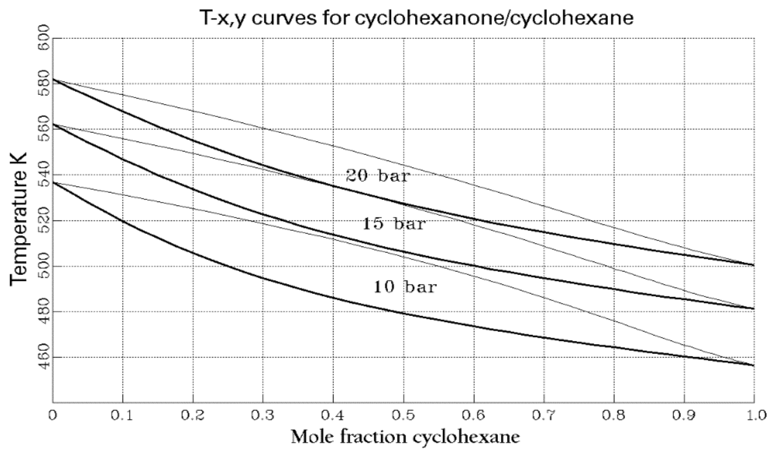


Figure 2.2: P-x,y Curves for Cyclo*hexanone*/Cyclo*hexane.* From [5].

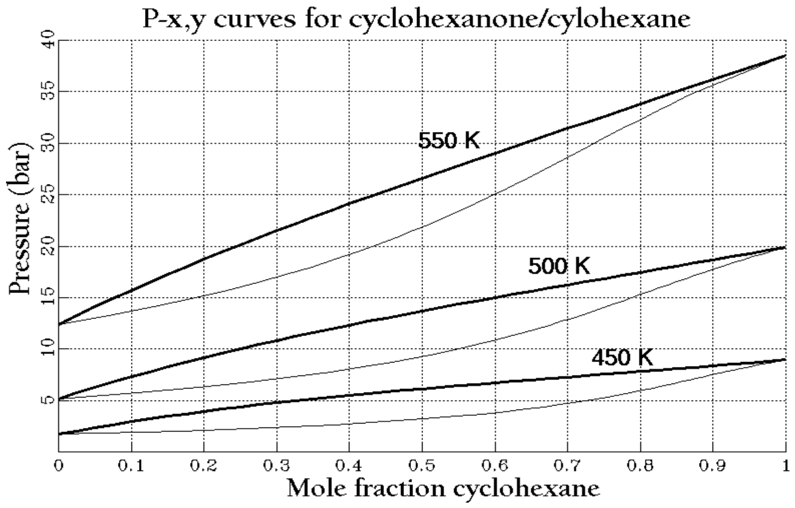


Figure 2.3: T-x,y Curves for Cyclo*hexanone*/Cyclo*hexane.* From [5].

Although two-dimensional plots are useful, the complete character of binary phase equilibrium is not well understood unless the three-dimensional nature of the equilibrium is conveyed using the pressure-temperature-composition plot as shown in Figure 2.4. The graphs shown in the Figure 2.2 and Figure 2.3 are in fact slices of the 3D phase equilibrium region shown in Figure 2.4.

In 3D space, the bubble-point locus and the dew-point locus become surfaces that connect along the sides of the diagram to form the vapor-pressure curves. The upper ends where the surfaces connect form the critical curve [5]. In the Figure 2.4, the solid lines show the isothermal sections, while the dashed lines show the isobaric sections. Bubble-point curves are red, Dew-point curves are green and the Vapor-pressure curves are white. The white tie lines that connect the intersections of isothermal and isobaric sections indicate the state where the liquid and vapor states coexist in equilibrium [5].

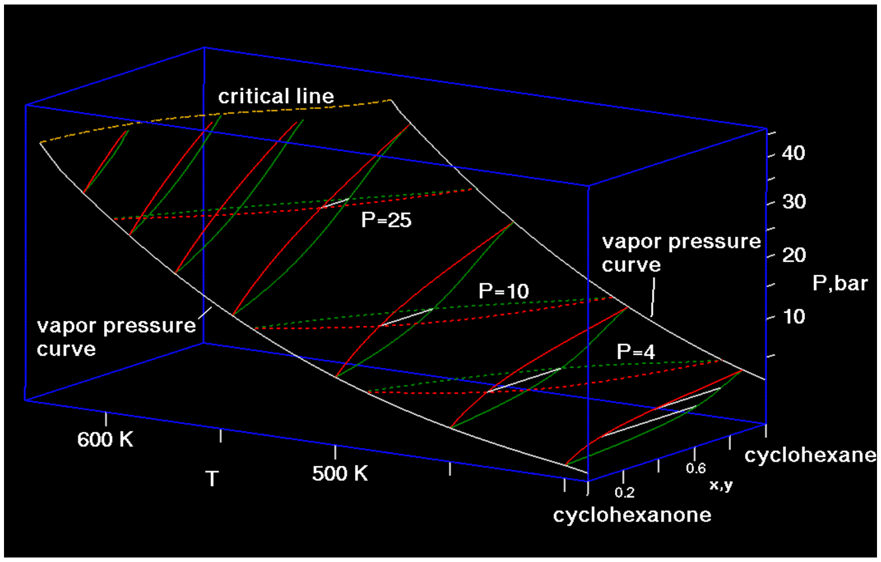


Figure 2.4: P-T-x,y Curves for Cyclo*hexanone*/Cyclo*hexane*. From [5].

2.3 Interpreting Phase Diagrams

“Referring to Figure 2.5, imagine starting with pure cyclo*hexanone* as a liquid at a particular temperature (say 500 K) and at a pressure somewhat above the white vapor pressure curve (marked "start"). Then add increasing amounts of cyclo*hexane* at the same conditions (P,T). The mixture moves along the directed yellow line, remaining a liquid until the composition crosses the bubble point curve (at B), after which a vapor phase appears. Adding still more cyclo*hexane* moves the composition toward the dew point curve (at D) where the liquid phase disappears completely. Higher cyclo*hexane* compositions at this temperature and pressure yield only a vapor phase”[5].

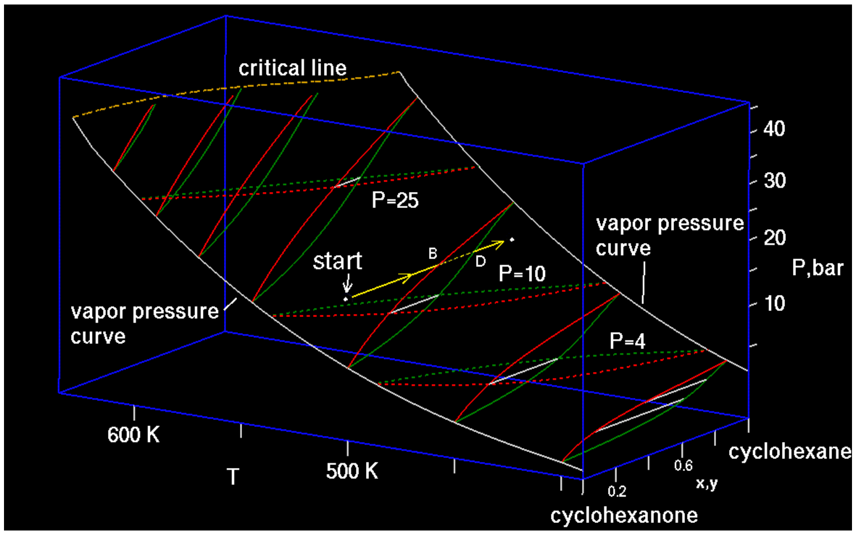


Figure 2.5: Interpretation of P-T-x,y Curves for Cyclo*hexanone*/Cyclo*hexane*. From [5].

3 Visualization Framework

Now, that we have an idea about phase diagrams, it is not difficult to see why computer graphics/visualization could provide an effective solution to the problem of visualizing 3D phase diagrams.

The task of generating 3D phase diagrams can be divided into writing computer programs for the following 2 parts:

1. *Data Generation*: Computer programs that would generate actual data points, depicting the phase behavior of the substances.
2. *Computer Graphics Modeling*: Computer programs that would read the actual data values and visualize them into the relevant geometry and provide various interactivity features.

Fortunately, the first part was taken care by one of our research group members, Dr.Clare McCabe. The computer program written by Dr. McCabe generated constant pressure slices for the phase diagrams of all the three modules based on the so-called SAFT equation of state [18]. Each slice generated over 6000 data points and there were more than 15 data files for each of the phase diagrams.

As far as the second part is concerned, it is important to first understand the issues involved in data visualization, particularly in a web-based environment. It is then important to choose the right software for web-based visualization. A critical goal in this research is that the visualization of the phase diagram be platform-independent. To date, there is no interactive platform-independent capability for visualizing and interacting with 3D phase equilibrium regions for binary mixtures. Previous efforts in this area have resulted in visualizations for SGI and for PCs [5].

3.1 Understanding the Visualization Pipeline

Data is comprised of numbers. Data are not geometric objects and they cannot be displayed directly by using simple graphic systems. We can visualize information only when we combine data with geometry. For instance, in order to display data in the form of two dimensional graphs or charts, we must convert the numbers (data) into geometric objects like lines, dots etc. "A simple definition of visualization is the merging of data with the display of geometric objects through computer graphics" [6].

According to references [6-8], Figure 3.1 shows the general visualization pipeline along with assigned client-server scenarios concerning web-based visualizations.

The visualization pipeline can be divided into 3 stages:

3.1.1 Step I - Data Gathering and Analysis

In scientific applications, data is generated in a variety of ways, including simulation results, mathematical modeling, physical measurements, etc. Data may already exist in the form of application data, e.g., business data, file system data etc. The raw data rarely provides enough information in and of itself – poor insights follow from the inspection of large columns of data. Therefore, typically in many engineering applications, data is analyzed by plotting various graphs using applications like Microsoft Excel, Xgraph, etc.

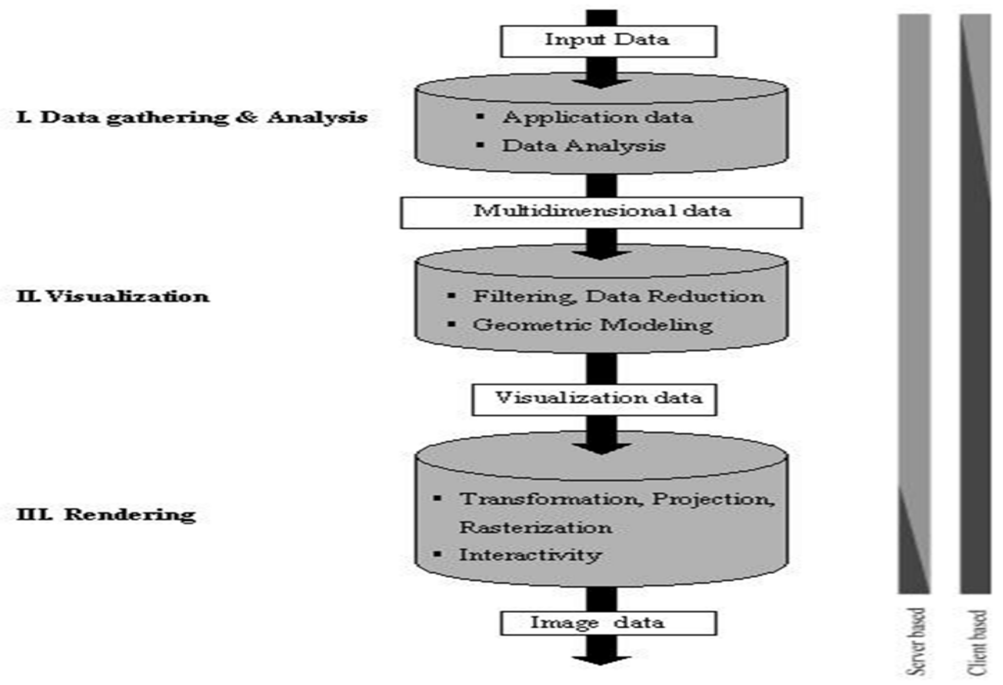


Figure 3.1: Conceptual Picture of the Components of a General Visualization Pipeline

Such methods are sufficient for 2-D data; however, for 3-D and higher dimensional data, more specialized methods are required.

3.1.2 Step II - Filtering and Geometric Modeling

The multidimensional information (data) could be extremely large in size. For instance, medical imaging data such as that from magnetic-resonance imaging, can be at a resolution of 512x512x200 points (or ~50M words or 100MB at 16 levels of color per pixel) [6]. As a result, we cannot assume that any operation, however simple it appears, can be carried out easily. Therefore, the first part of the step II consists of applying various data reduction or filtering techniques such as sampling (uniform, random) in order to reduce the number of data points. The second part of step II consists of mapping of the data to a geometric model. Without mapping data onto a geometric model such as lines, curves, 3D charts/objects, in many cases the data would not be useful. This is particularly true of scientific data, which, whether experimental or simulated, is likely to be continuous (i.e., lie on surfaces in some appropriate dimensional space). A contrary example might be statistical data in the social sciences, such as income by zip code, in which case plotting the raw data may be insightful.

3.1.3 Step III - Rendering and Interactivity

This final stage of the visualization process involves the creation of discrete geometry model. This stage processes the viewing transformation, the projection and finally the rasterization of the image data. Various interactivity features such as zooming, clipping plane, lighting etc. can be provided for enhancing the visualization.

The web-based visualization scenario can be classified into two types [7]:

- **Server-based or "Fat Server"**

  In server-based web visualization, all the graphics processing parts from step (I) to step (III), including discretization, is done on the server side. The data is in the pure geometric form, for example, triangle strips, and is transferred over the net to the client. The client simply displays the transferred image data. As indicated by the shaded bars on the right side of the visualization pipeline shown in Figure 3.1, most of the visualization process is carried out at the server side. This scenario requires a high-speed visualization server along with a high-speed network connection, in order to provide a high level of graphical interactivity.

- **Client-based or "Fat Client"**

  In client-based web visualization, most of the processing stages are carried out at the client end. As indicated by the shaded bars on the right of Figure 3.1, the borderline between step (I) and step (II) typically represents the boundary of the server's task and the client's task. As long as the client end uses high-end graphics acceleration hardware, a high degree of interactivity can be easily achieved.

3.2 Software

As the graphics capabilities of even the cheapest personal computers, driven by computer gaming, now rival and surpass the high-end graphics workstations of less than a decade ago, increasingly client-based solutions are becoming the norm. In our approach, we use the "Fat Client" or client-based web visualization scenario.

Key Issue: How do we make the 3D phase diagrams easily and widely accessible for students?

Alternatives:

Platform independence and client-side rendering was our primary objective. Table 3.1 describes the issues involved and the available software solutions. Java [9], which is available on a wide variety of platforms, provided a feasible and attractive choice. Applets are programs that run in the Internet browser and therefore provide easy and wide accessibility.

Table 3.1: Issues and Solutions in Software Selection

Issue	**Software Solution**
Wide accessibility	Java Applets
3D graphics	1. Virtual Reality Modeling Language 2. Open GL

Virtual Reality Modeling Language (VRML) [8] is a web-based 3D modeling language that utilizes the 3D graphics acceleration hardware for the visualization process. VRML maintains platform-independence by using Java. However, a plug-in needs to be installed in order to view the VRML module. On the other hand, OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. OpenGL is highly optimized in hardware and software and is targeted to a variety of platforms.

According to the study conducted in [8], the Figure 3.2 depicts a detailed performance evaluation of the OpenGL, and the VRML-based implementation of a client-side system. For small numbers of triangles, the VRML implementation shows slightly better performance. However, for large numbers of triangles, corresponding to complex surfaces, we observe a superior performance of the OpenGL implementation.

We therefore chose to use OpenGL along with Java for our implementation, because in contrast to VRML, besides superior performance, OpenGL also allows direct access to the rendering pipeline.

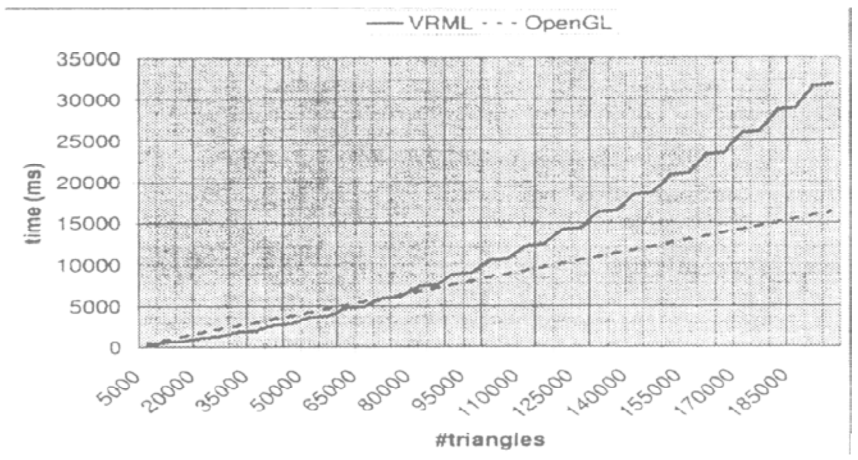


Figure 3.2: Transfer and Display Times of Triangles Using VRML and OpenGL. From [8].

Now the buzz question, "Will Java affect the performance of my code?"

Although Java is generally slow compared with C, performance is not greatly affected here because Java simply provides the platform-independence for these visualization modules. The compute-intensive part of the modules we are developing lies in the visualization, and OpenGL performs all the graphics calculations in a manner optimized for the client hardware.

How to integrate Java with OpenGL?

There are two primary choices for implementing OpenGL with Java:

1. Java3D (High Level API)
2. Java-OpenGL bindings (Low level programming)

An excellent review [10] provides a deep understanding about the issues involved in using Java with OpenGL.

3.2.1 Java3D

The Java 3D^{TM} application-programming interface (API) provides a set of object-oriented interfaces that support a simple, high-level programming model. This enables developers to build, render, and control the behavior of 3D objects and visual environments.
Java 3D is meant to give Java developers the ability to write applets and applications that provide three dimensional, interactive content to users. It uses either DirectX or the OpenGL low level API to take advantage of 3D hardware acceleration.

3.2.2 Java-OpenGL Bindings

Java-OpenGL bindings map the complete OpenGL and GLU API to Java and implement window handle functions (native and Java), while using the Java-Native-Interface (JNI) of Java or the JDirect-Interface of MS-JVM, Win32, X-Window and Mac. Although Java-OpenGL bindings are language bindings that provide all the advantages of OpenGL, they do possess certain disadvantages. First and foremost it that there are few standardized Java-OpenGL binding products available for all relevant platforms in the market. The following are a few:

- Magician (well documented but not available for free).
- GL4java (freely available for a wide variety of platforms; well documented and has automatic web-installation for Win32 and Mac machines) [11].

These language bindings are meant to give graphics programmers the ability to make OpenGL function calls in Java by using simple prefixes (“gl” in case of GL4Java) to the corresponding OpenGL calls. Examples are given in Table 3.2

Table 3.2: OpenGL vs. GLJ4Java Function Calls

OpenGL functions	Corresponding GL4Java functions
glBegin(GL_POLYGON)	gl.glBegin(GL_POLYGON);
glEnd()	gl.glEnd( );

3.2.3 Java3D vs. Java-OpenGL Bindings

The Figure 3.3 shows the difference between programming using Java3D and Java-OpenGL bindings. Using the Java-OpenGL bindings, one can directly access the rendering pipeline. However, this flexibility is not available with Java3D. Besides, an OpenGL programmer can use the language bindings in a manner very similar to using OpenGL, without having to learn the entirely new Java3D framework. The Table 3.3 describes the differences between the Java3D and Java-OpenGL. Although Java3D provides a few benefits over Java-OpenGL, we chose to use Java-OpenGL because it gave us the flexibility of writing OpenGL programs in Java.

3.2.4 Our Choice

After a careful review of the literature [7], [8], [10], we decided to choose the GL4Java (Java-OpenGL) bindings for implementation of the phase diagrams.

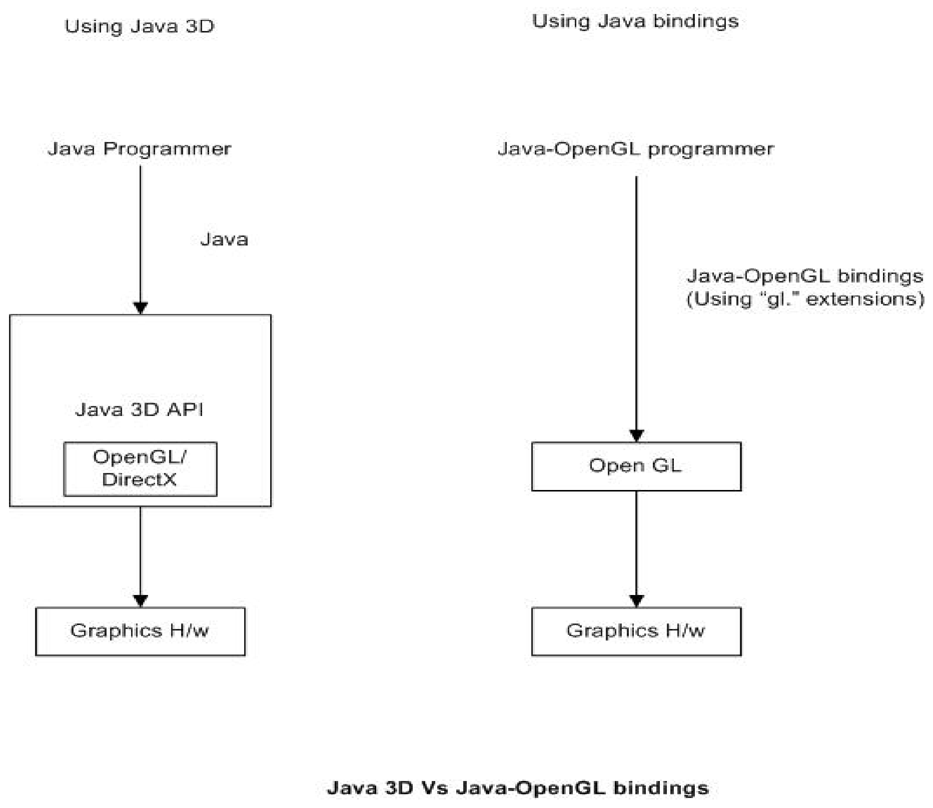


Figure 3.3: Using Java3D vs. Using Java-OpenGL Bindings

Table 3.3: Comparison between Java3D vs. Java-OpenGL Bindings

Java 3D	**OpenGL (Java-OpenGL bindings)**
High-level programming Java 3D is intended to help Java programmers without much graphics or multimedia programming experience, to use 3D in their applications [10].	*Low level programming* OpenGL is designed to optimize for the best possible speed and give programmers the greatest possible control over the rendering process [10].
A standard extension API. Java platform licensees are given the option to implement the API if they like, but they're not required to implement it. As a result of this, the portability of Java 3D codes is greatly reduced, since most Java3D vendors have to struggle keeping up with the constant changes [10].	*A complete API* OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms [10].
Severe availability constraints These are the result of Java 3D's status as an extension API. The only major vendor currently providing a Java 3D implementation is Sun, with its implementations for Solaris and Win32 [10].	*Available Everywhere* *OpenGL is available for every flavor of Unix, Windows, MacOS and many other operating systems [10].*

Table 3.3 Continued

Java 3D	**OpenGL (Java-OpenGL bindings)**
Documentation deficits. Java3D is not as well documented as OpenGL. The complexity of the 3D graphics API has made it difficult to document and provide easy support to its users [10].	*Adequate Documentation* Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain [10].

4 Visualization Pipeline

Given our understanding of the visualization pipeline, we can categorize the development of applets to describe the phase diagrams, based on the three steps portrayed in Figure 3.1.

4.1 Step I – Data Gathering and Analysis

This is the first step involved in any visualization system. Before visualizing any data, it is important to gather the data and carefully analyze it.

The data files generated by Dr. McCabe contained constant pressure slices of the phase diagrams. A constant pressure slice (here P=0.7 in dimensionless units) appeared in the format shown in Table 4.1 (all units in this table are dimensionless). The corresponding Excel plot for the constant pressure slice is shown in Figure 4.1.

Table 4.1: File with Pressure Slice (P=0.7) Data

Line Number	Temperature	Pressure	Mol. Fraction (I)	Mol. Fraction (II)
1	2.17558574	0.70000000	1	1
…	…	…	...	...
3000	1.500	0.70000000	0.085	.0.78857243
…	…	...	…	...
6000	0.93778574	0.70000000	0	0

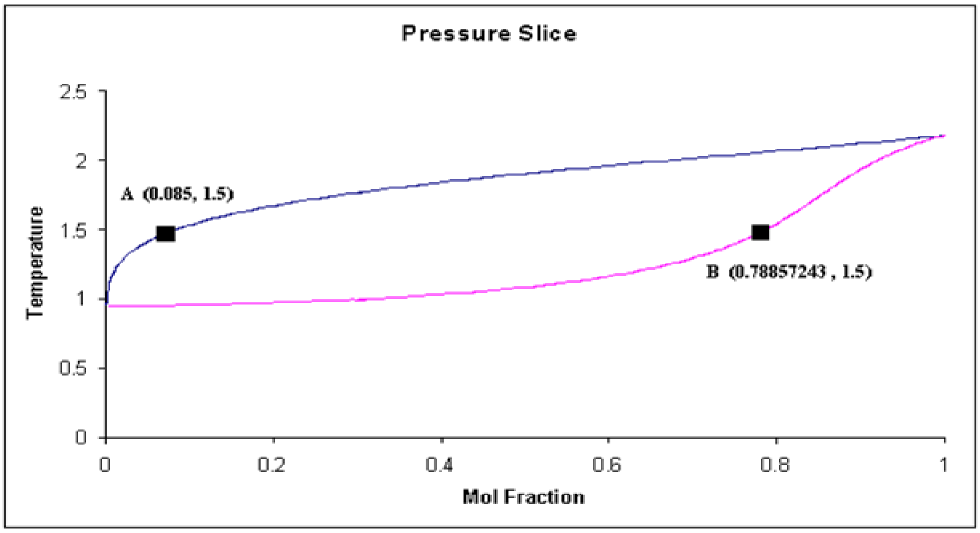


Figure 4.1: Temperature-Mole Fraction Plot of the Constant-Pressure (P=0.7) File Data

4.1.1 Observations

1. Each line in the text file corresponds to two points on the Excel sheet. For instance, the points A (0.085,1.5) and B (0.78857243, 1.5) represent sample points that have been derived from the line (3000). In 3 dimensional space, the same points (3-D points defined as (Concentration, Temperature, Pressure)) would be A (0.085,1.5, 0.7) and B (0.78857243, 1.5, 0.7) respectively.
2. After analyzing each slice, an Excel plot as shown in Figure 4.2 was plotted in order to understand the characteristic of the data.

4.1.2 Analysis

After carefully observing each of the data files for the phase diagram, we observed a consistent uniformity in the data values. No sporadic data values were observed.

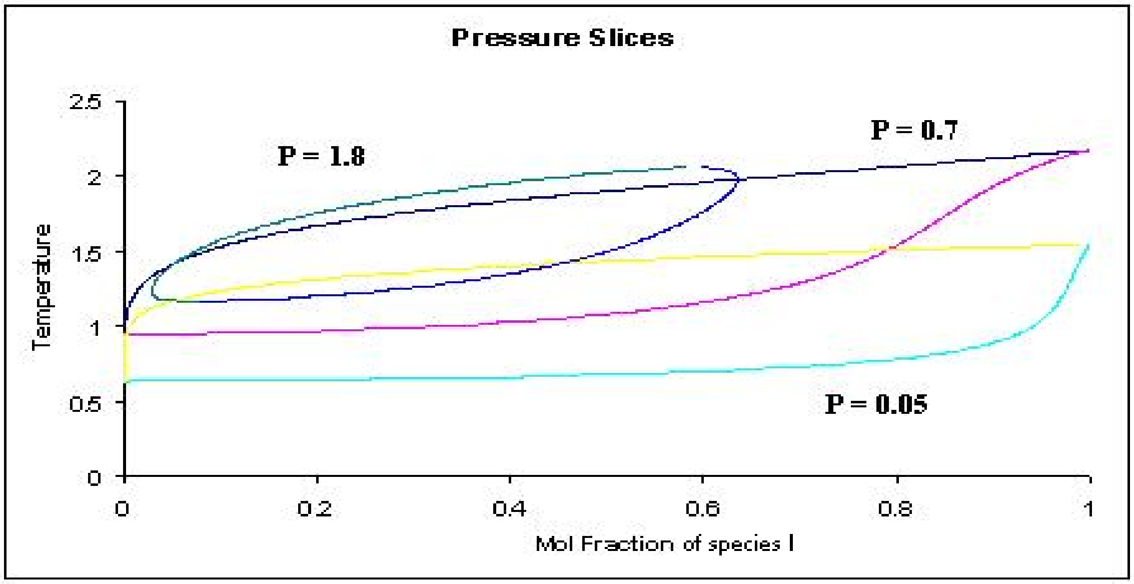


Figure 4.2: Temperature-Mole Fraction Plots of the Constant-Pressure (P=0.7, P=0.05, P=1.8) Data

4.2 Step II – Filtering and Geometric Modeling

In this step, we consider the various issues involved in visualization of large data sets. We further discuss about the various data reduction and geometric modeling techniques.

4.2.1 Large Data Set Visualization

Visualization, by definition, means interaction with people to give them a better insight about information [12]. Reference [12] explains how as humans, we perceive images and extract information from them. It further explains how important it is to ensure the right density of points in an image – with too few points we see an image as spurious connections; however, with too many points the entire image appears as a blob. Therefore, we often find sampling acceptable because our visual processing depends upon approximate rather than exact properties of data [12]. Also, by sampling the data, we reduce the processing speed involved in plotting data. This allows for faster rasterization and better interaction of the visualized module. The results discussed in Reference [12] further explain the following two problems involved in large data visualization:

1. Visual limits – It can be difficult to visualize data sets either due to the perceptual limitations of the user or due to the hardware limitations of the display device.
2. Computational limits – When the data volume increases, the processing power, data storage or network traffic increases as a result of the high interaction of the visualization system.

4.2.2 Filtering (Sampling)

Several sampling techniques are described in Reference [12], amongst which uniform and random sampling techniques seemed appropriate for our data visualization system. Our data was largely uniform and monotonic in nature. Therefore, simple sampling techniques such as uniform sampling (selecting every nth data point from the data set) or random sampling (selecting a random data point from the data set) are suitable for the data filtering process. Our code provides the user with an option of selecting the number of data samples and the sampling technique (simple/random).

4.2.3 Geometric Modeling

There are two techniques described in Reference [6] that are useful for modeling volumetric structured data sets (three-dimensional arrays of voxel or volume element values that correspond to equally spaced data samples):

1. **Direct volume rendering:**

   In direct volume rendering, every voxel is used in representing a model.

   The principal drawback of this method is the large amount of data involved in the process.

2. **Isosurface rendering:**

   In this technique, only a subset of voxels is used for modeling. For a function f(x,y,z), an isosurface is a function defined by an implicit function f(x,y,z) = c, where c is the isosurface value. For the discrete visualization of voxels, isosurface methods seek to find the appr\oximate surface.

In our modules, we had over 6000 data points in each of the 15 data files. Rendering these many data points using the direct volume rendering technique would be highly inefficient. Therefore, we chose the isosurface rendering technique in our visualization modules.

We had two choices for modeling the geometric objects:

i. Draw polygons connecting the sampled data points.

ii. Use curve/surface-fitting techniques to reconstruct the underlying function (in this case, 3D surface).

The following were the reasons for our choice of using interpolation techniques over drawing polygons :

a. **Representation**

   Points, line segments and polygons form the basic graphic primitives. Using a very large number of such basic primitives generates smooth curves or surfaces.

However, by mathematically describing a small number of parameters such as "control points", smooth curves and surfaces can be easily generated. According to [13], when data fitting becomes a primary concern for data produced by simulations or measurements, then data points may be used as control points in the representation model.

b. Storage – less control points

Imagine rendering a surface using approximately 1000 triangles. The same surface can be modeled using surface fitting techniques with just 16 control points. While the 1000 triangles would only approximate the true surface, the control points can accurately represent the true surface. Storing 16 control points far reduces the overhead involved in saving 1000 triangles along with the normal vector information at each vertex. Also, in modules that provide interactive features like model rotating, volume slicing etc, using fewer control points for modeling greatly increases the rendering speed.

4.2.4 Non-Uniform Rational B-Splines (NURBS)

Since we decided to use interpolation techniques for the geometric modeling of the phase diagrams, it is important to choose a suitable representation for the same. References [8], [13] and [14] provide good reviews on geometric models for large data set visualization. NURBS modeling provides a suitable choice in our case, because it provides representation of almost any kind of geometric models, using a far fewer number of control points.

4.2.4.1 Bezier Surfaces

The Bernstein polynomial of degree **n** (or order **n**+1) is given by

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

The Bernstein polynomials form the blending functions for the Bezier surfaces.

The Bezier surface is defined by

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{ij}$$

where P_{ij} is a set of *m*n* control points, *u* varies from 0 to 1 and *v* varies from 0 to 1.

Bezier surfaces posses a number of properties [6], of which the convex-hull and continuity properties are particularly important.

4.2.4.2 B-Spline Surfaces

Although Bezier curves and surfaces are powerful, a potential problem with them is that the control points have global scope. A change in one control point affects the global shape of the curve or surface.

An alternative to Bezier curves/surfaces is B-Spline curves/surfaces. The "B" in B-Spline stands for basis. Basis is almost similar to the blending function for Bezier surfaces, except for one small difference. In B-Splines, the blending function can be zero outside a particular range. The effect of defining such a range is that it limits the scope over which a control point has influence.

Thus, a B-Spline is defined by control points *and* the range in which each control point is active. These ranges are specified, indirectly, through something called a knot vector. The knot vector is a non-decreasing sequence of numbers between 0 and 1.

Knot Vector = {u_0, ..., u_m | for all i 0 <= u_i <= 1 and u_i-1 <= u_i}

Given a knot vector with “m” knots as above, and assuming we have defined “n” control points (P_0, ..., P_n), the degree of the curve is defined as p = m - n - 1.

4.2.4.3 NURBS

A NURBS is defined by control points *and* a knot vector. A Bézier curve is a B-Spline where the first p+1 knots = 0, the last p+1 knots = 1. NURBS posses all the properties (convex hull, continuous and local shape controllability) of 3D splines. They have an important additional property of shape invariance under transformation . This property ensures that NURBS curves are handled correctly in perspective views. We therefore decided to choose the NURBS surfaces for our Geometric Modeling step.

4.3 Step III – Rendering and Interactivity

The third and final step of the visualization process involves rendering of the geometric model and providing interactivity to the modules. The following discussion describes rendering NURBS surfaces with OpenGL.

4.3.1 NURBS Rendering

The NURBS interface requires a NURBS context object that is passed into each call. So, the first thing to do is to create a NURBS context object:

```
nurbs = gluNewNurbsRenderer();
```

gluNurbsCurve is the function that actually evaluates the NURBS. The first argument is the NURBS context object. The next two are the number of knots and a pointer to the knot vector. Then comes the width of each control point (3D vs.. 4D) and a pointer to the control points. The next argument is the "order" of the curve. The order is the degree + 1. And, looking back above, the order is equivalent to the number of knots minus the number of control points. The final argument is the "type" which is the same as the evaluator's target argument.

Summing up, the Java-OpenGL code would look like:

```
float knots[8] = {0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0};
gl.gluBeginCurve(nurb);
gl.gluNurbsCurve(nurb, 8, knots, 3, &ctlpoints[0][0], 4, GL_MAP1_VERTEX_3);
gluEndCurve(nurb);
```

The Figure 4.3 shows a snapshot of the NURBS surface modeled using OpenGL.

4.3.2 Interactivity

The module provides students with the following interactive features for better visualization of the phase diagrams.

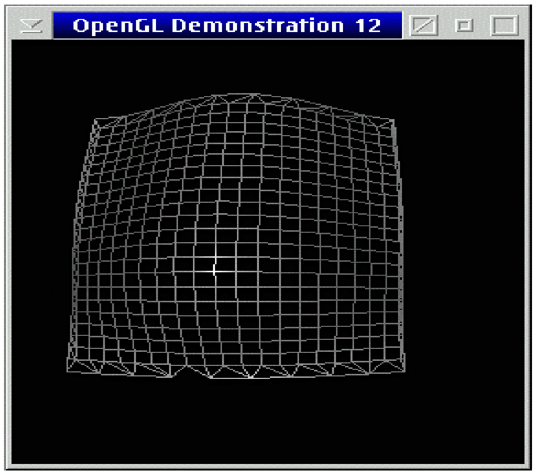


Figure 4.3: OpenGL Representation of NURBS Model Using the 16 Control Points

1. Click and drag the phase diagram to get different views of it.

2. Right click (or control-click on single button mouse systems) to select/ deselect various options:

   a. Volume Slice feature

      This feature enables the students to slice a cutting plane through the entire phase diagram. One can view the 2D slices of the phase diagram as it appears at different temperatures or pressures.

   b. One can view the following projections by right clicking the mouse.

      a. Temperature-Pressure Projection

      b. Pressure-Concentration Projection

      c. Temperature-Concentration Projection

3. One can also use the following keyboard keys for:

" Z " to Zoom in

" S " to Zoom out

" L " to Switch Lighting

5 Analysis and Implementation

In this section, the various processes involved in the development of the visualization system are depicted using data flow diagrams.

5.1 Analysis (Data Flow Diagrams)

5.1.1 Context-Level Diagram (CLD)

As shown in Figure 5.1, the Context-level diagram represents an overview of the visualization system.

- Student and Teacher represent the external entities that interact with the system.
- The Teacher provides visualization data, which are data files (text files) containing the 2D projections of the phase diagrams.
- The Student visits the website, installs the Java-OpenGL plug-in and views the phase diagrams.
- The phase diagram provides interactivity features like volume slice, zoom, lighting etc.

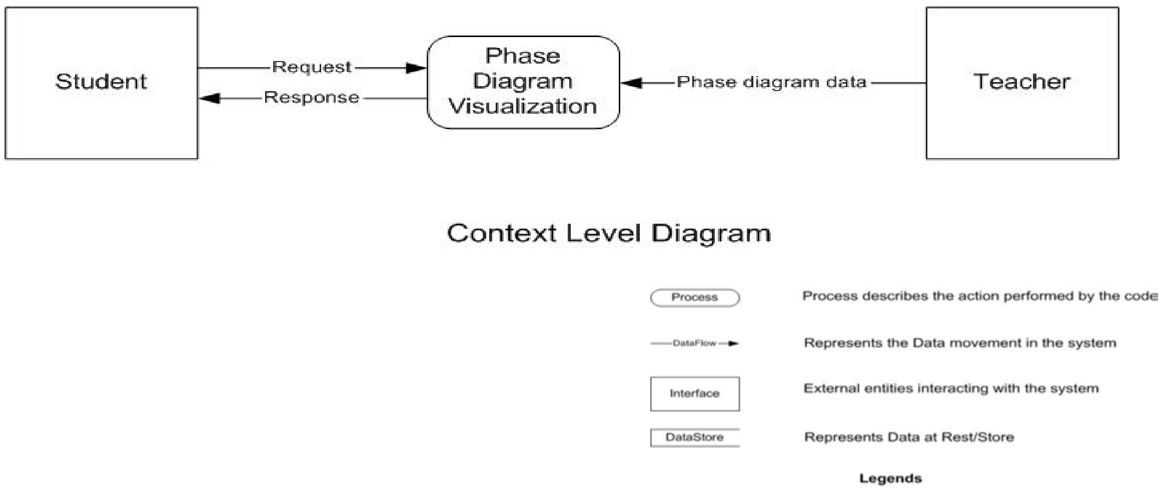

Student
Request
Response
Phase Diagram Visualization
Phase diagram data
Teacher
Context Level Diagram
Process
Process describes the action performed by the code
DataFlow
Represents the Data movement in the system
Interface
External entities interacting with the system
DataStore
Represents Data at Rest/Store
Legends

Figure 5.1: Context-Level Diagram of the Phase Diagram Modules

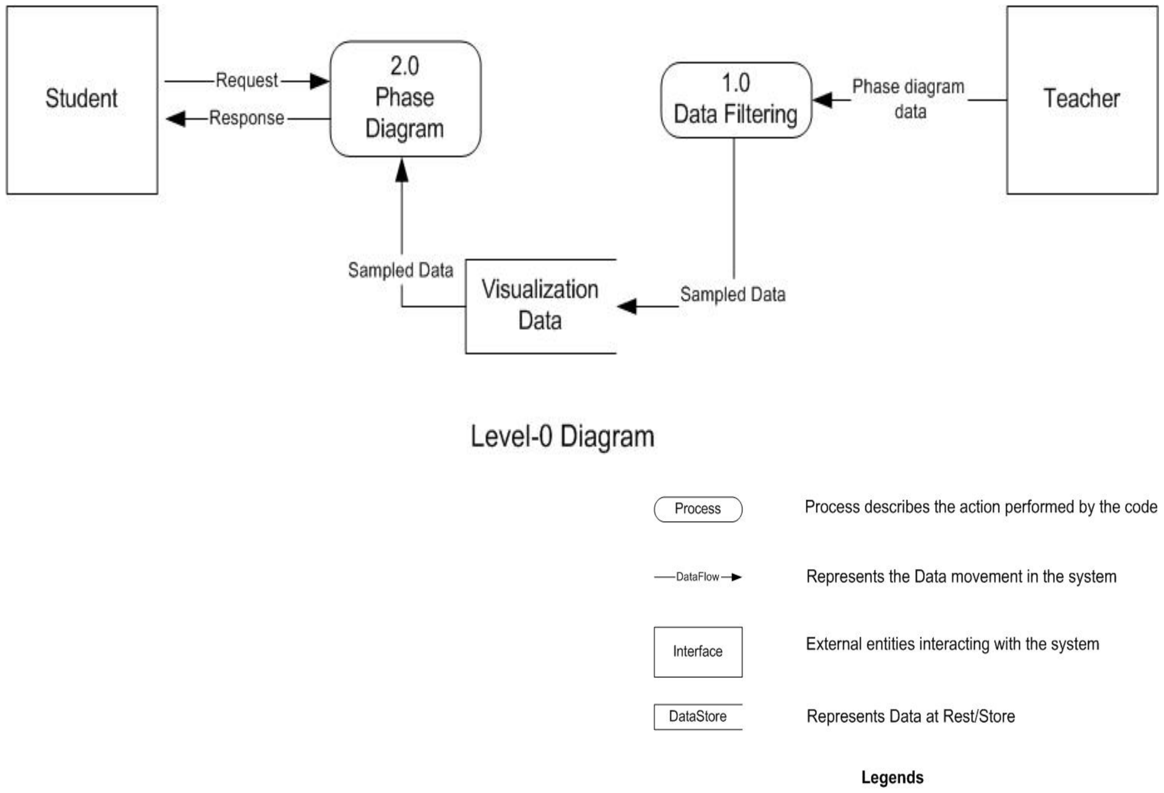


Figure 5.2: Level-0 Diagram of the Phase Diagram Modules

5.1.2 Level-0 Diagram

As shown in Figure 5.2, the Level-0 diagram represents the lower level process model. The visualization system is divided into 2 processes:

1. Data Filtering (Process 1.0):

   This process involves data sampling. The large data files are parsed and sampled at regular intervals. The reduced data is stored in data files (text files), which is then used for visualization.

2. Phase diagrams (Process 2.0):

   This process involves the geometric modeling of the visualization data. It also involves providing interactivity features.

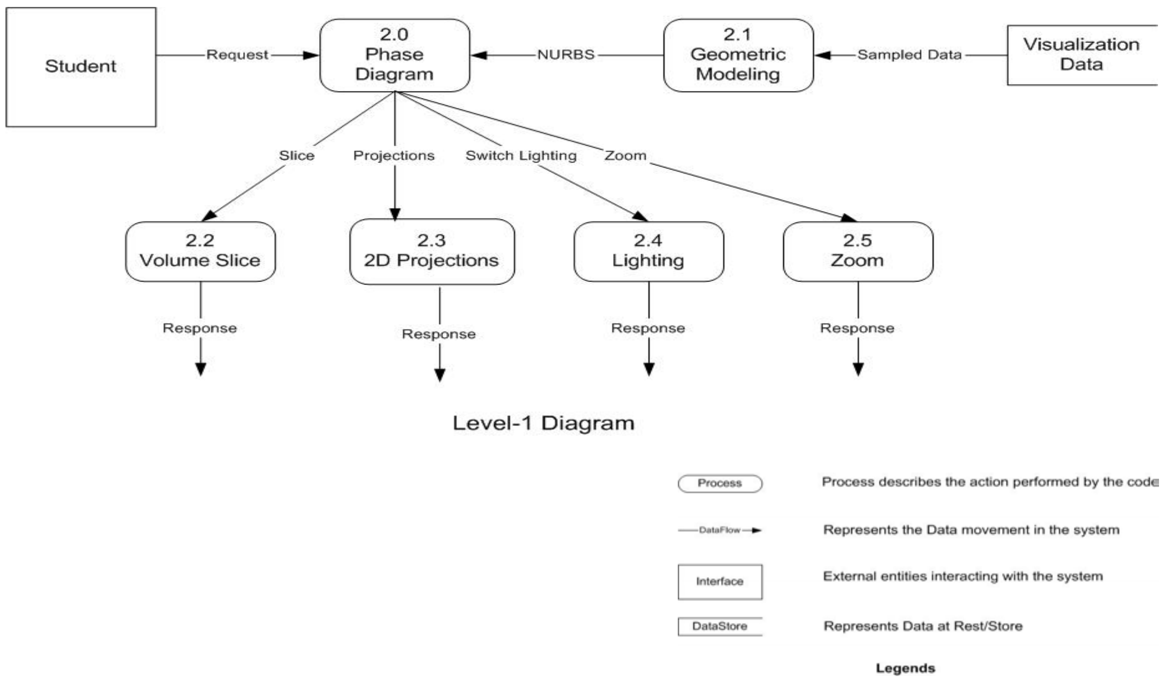


Figure 5.3: Level-1 Diagram of the Phase Diagram Modules

5.1.3 Level-1 Diagram

As shown in the Figure 5.3, the Level-1 diagram is a further decomposition of the phase diagram Process (2.0)

- The phase diagram (Process 2.0) involves the geometric modeling (Process 2.1) of the visualization data and providing various interactive features (Processes 2.x)
- Geometric Modeling (Process 2.1) involves the construction of the NURBS surfaces using the visualization data.
- A student can interact with the phase diagram by slicing (Process 2.2) through the phase diagram, changing the lighting of the system (Process 2.4), zooming into the phase diagram (Process 2.5) or viewing the *p-x (pressure-concentration)*, *p-T (pressure-temperature)* and *T-x (temperature-concentration)* projections (Process 2.3).

5.2 Implementation – Class Diagram

Here the class diagram gives an overall picture of the various java class files developed as part of the system. Also, the pseudo-code implementation, describe in the Appendix A.1, gives an overview of the various methods, constructors, etc. used in the class files.

Based on the Data Flow Diagrams describes earlier, Figure 5.4 represents the implementation class diagram.

1. **Data Sampling Class (Data Filtering Process)**
    - ParsePressure2.java – This class represents the Data Filtering process (Process 1.0). It reads large data files (text files), samples data points at regular intervals and stores the samples in simple text files.

2. **Phase diagram Classes (phase diagram modeling process)**
    1. Surface2.java – Main Applet class that creates an instance of the SurfaceCanvas2.java class
    2. SurfaceCanvas2.java – This class performs the following functions:
        - i. Geometric Modeling (Process 2.0): Reads the reduced data points from the files and models the phase diagram using NURBS modeling.
        - ii. Phase diagram rendering and interactivity: Provides the interactive features (Volume Slice, 2D projections, Lighting, Zoom etc.)
    3. MatrixFuncs.java – Provides for the rotation of the phase diagram around an axis.

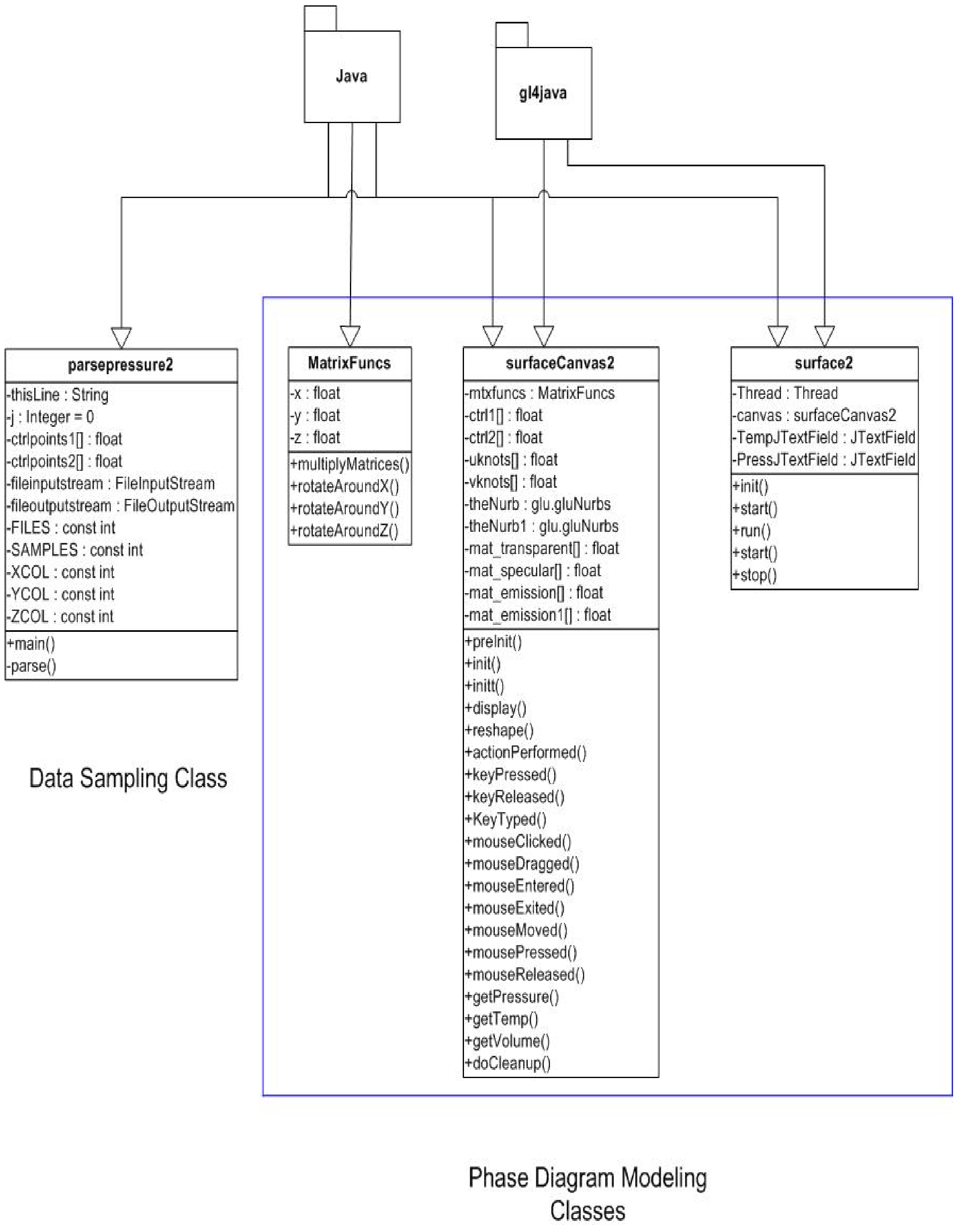


Figure 5.4: Class Diagram of the Phase Diagram Modules

6 Results and Discussions

6.1 Type I (Butane + Methane)

As shown in the Figure 6.1, the constant-pressure slices are plotted for the entire 3D phase diagram for the butane-methane mixture. The Figure 6.2 shows all the constant-pressure slices for the same mixture in order to give a 3D perspective of the model. The Figure 6.3 shows an orthogonal view of the 3D phase diagram applet, constructed using the visualization data. The Figure 6.4 shows the *p-T (pressure-temperature)* projection view of the applet, while the Figure 6.5 shows the *p-x (pressure-concentration)* projection view, and the Figure 6.6 shows the *T-x (temperature-concentration)* projection view of the applet for the butane-methane mixture.

6.2 Type I (Butane + Propane)

As shown in Figure 6.7, the constant pressure slices are plotted for the entire 3D phase diagram for a butane-propane mixture. The Figure 6.8 shows all the constant-pressure slices for the same mixture in order to give a 3D perspective of the model. The Figure 6.9 shows an orthogonal view of the 3D phase diagram applet, constructed using the visualization data. The Figure 6.10 shows the *p-T (pressure-Temperature)* projection view of the applet, while the Figure 6.11 shows the *p-x (Pressure-Concentration)* projection view, and the Figure 6.12 shows the *T-x (Temperature-Concentration)* projection view of the applet for the butane-propane mixture.

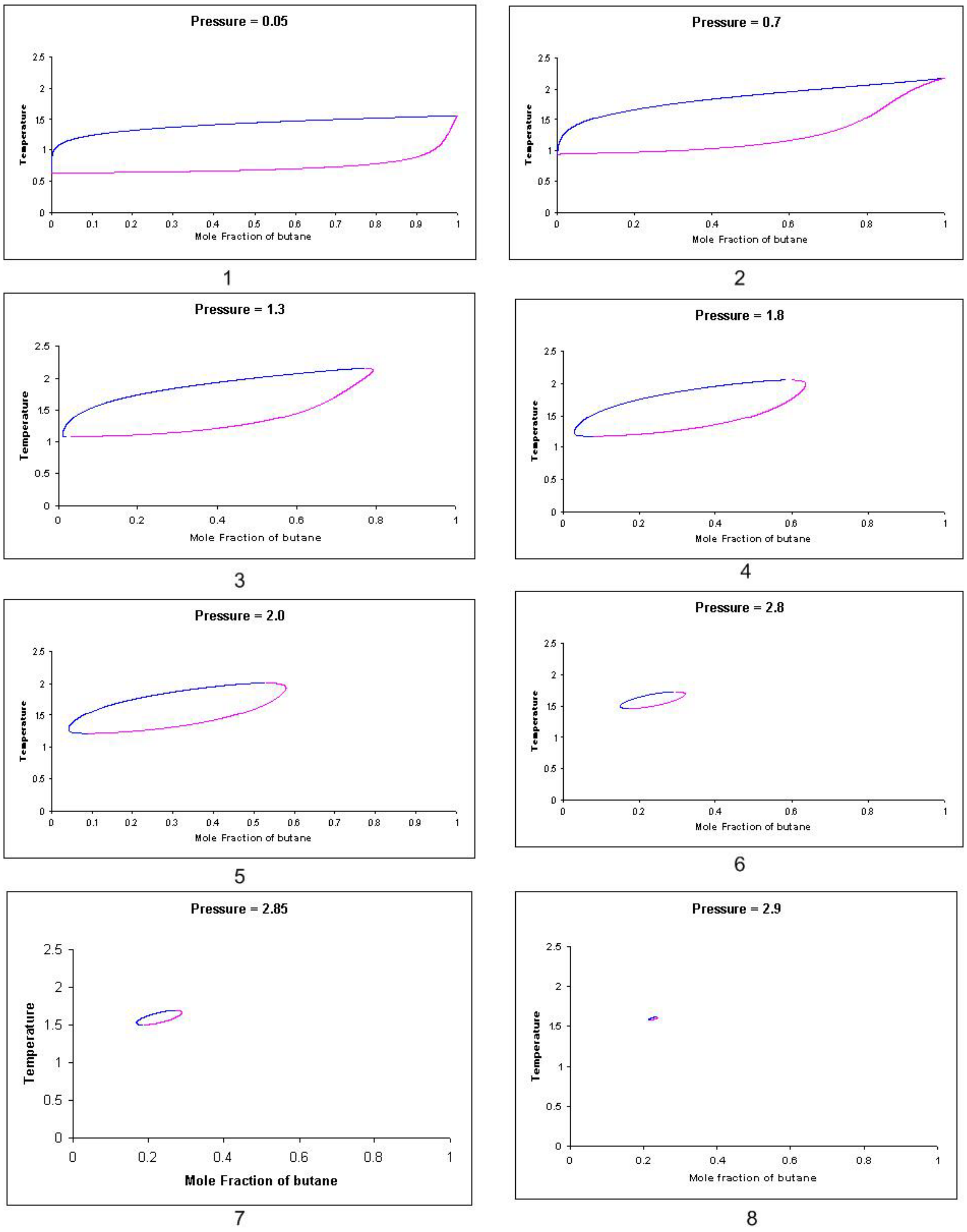


Figure 6.1: Individual Constant-Pressure Slices for a Butane-Methane Mixture

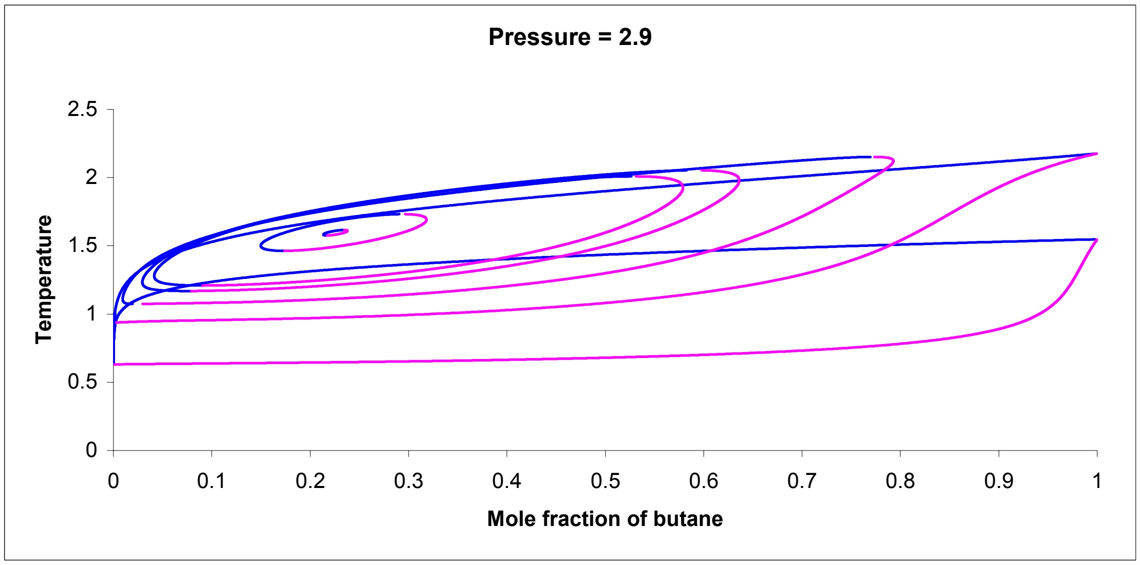
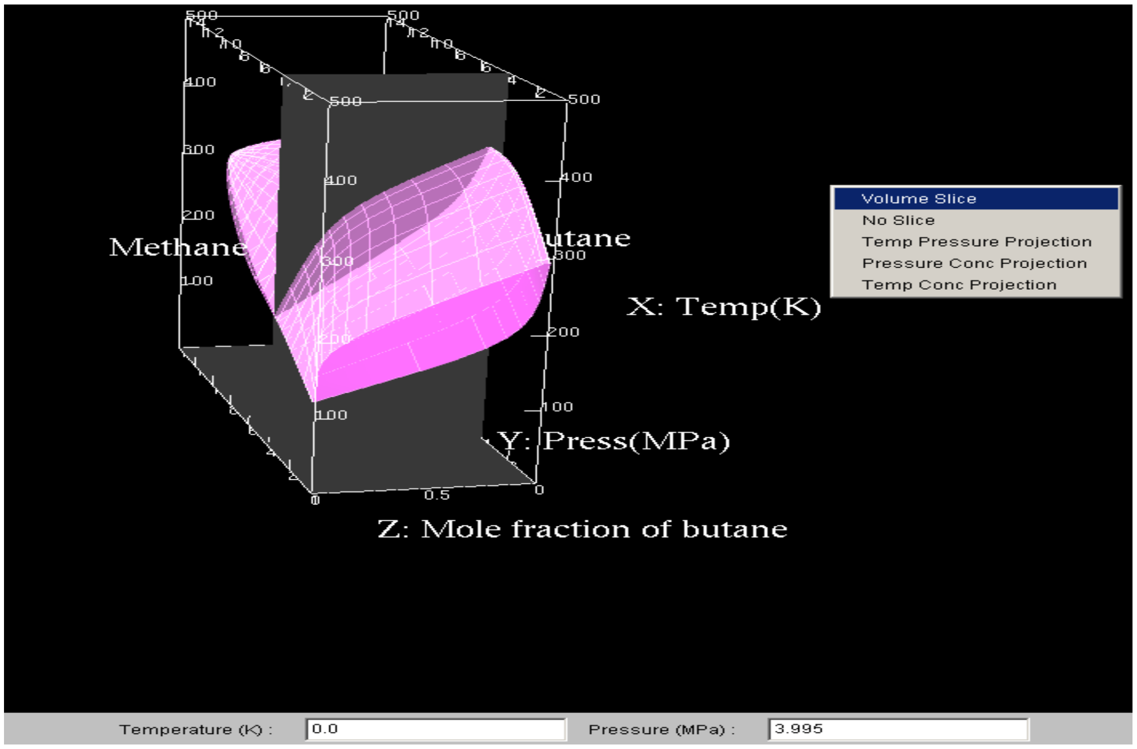
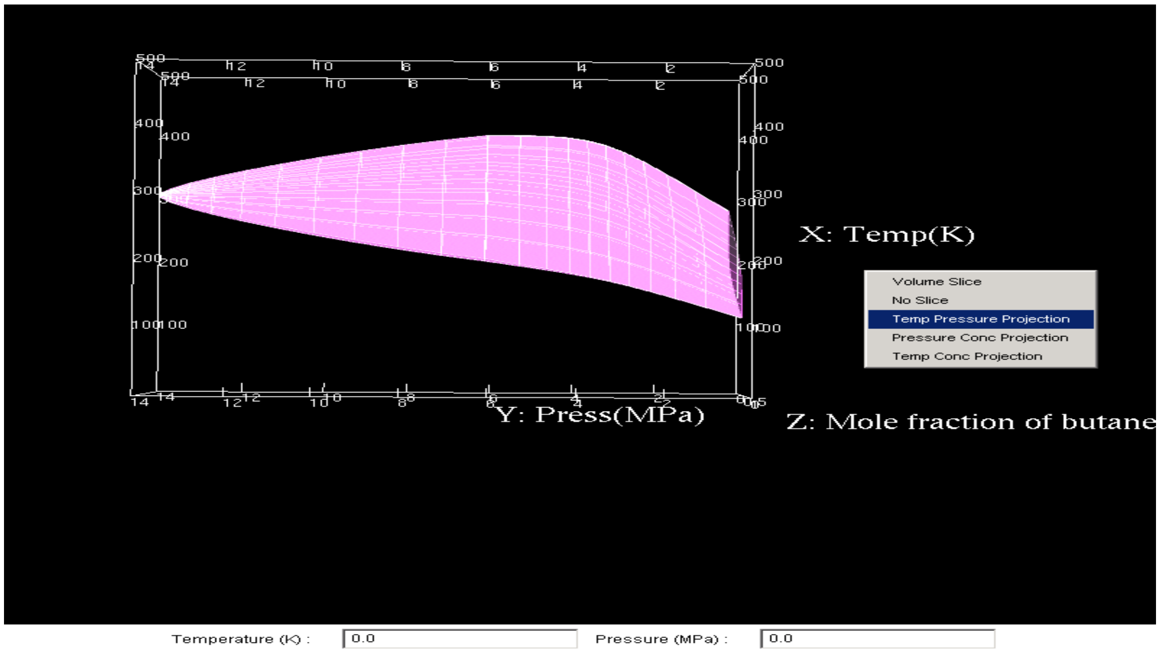


Figure 6.2: All Constant-Pressure Slices for the Butane-Methane Mixture



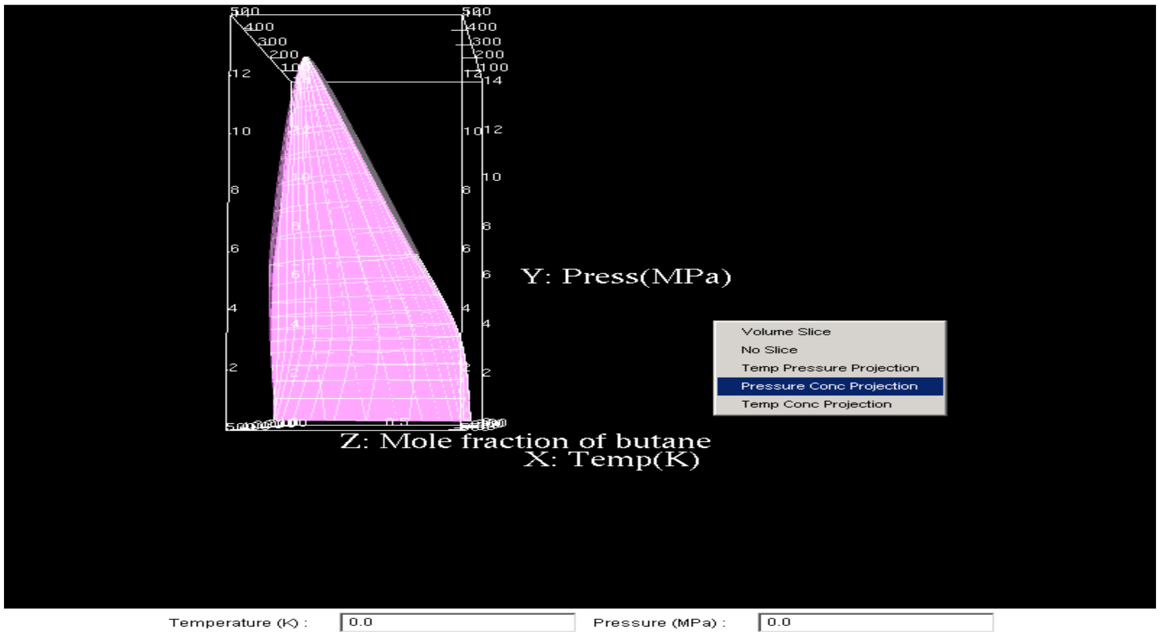
Temperature (K) : 0.0 Pressure (MPa) : 3.995

Figure 6.3: An Orthogonal View of the 3D Phase Diagram Applet for the Butane-Methane Mixture



Temperature (K) : 0.0 Pressure (MPa) : 0.0

Figure 6.4: Pressure-Temperature Projection View of the Applet for the Butane-Methane Mixture



Temperature (K) : 0.0 Pressure (MPa) : 0.0

Figure 6.5: Pressure-Concentration Projection View of the Applet for the Butane-Methane Mixture

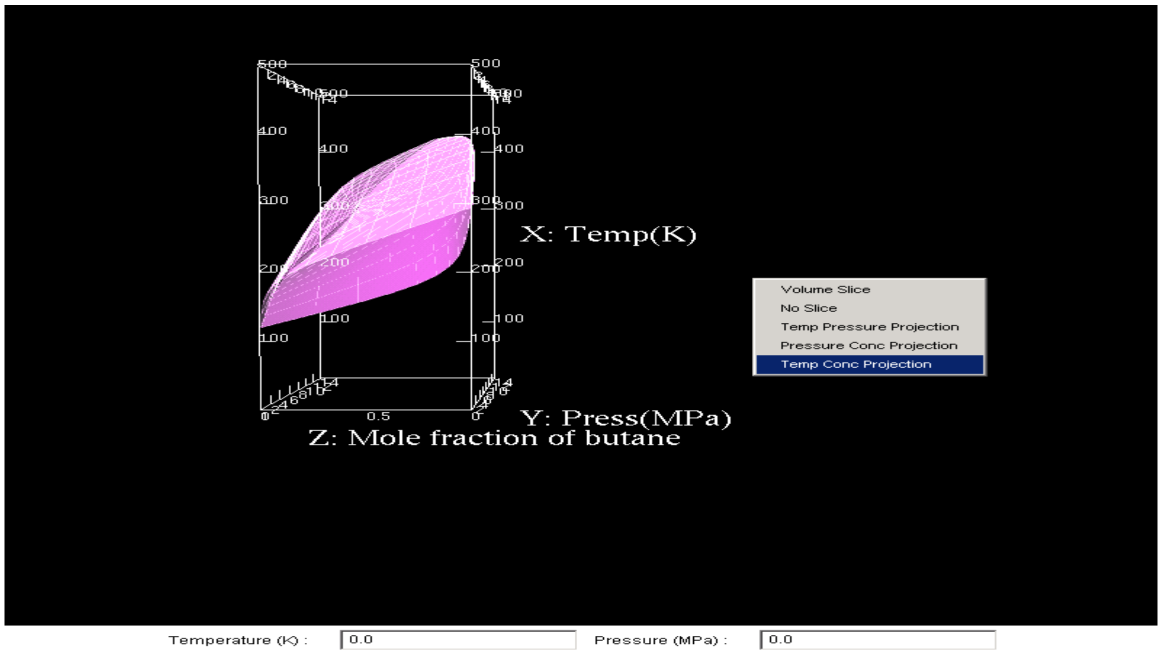
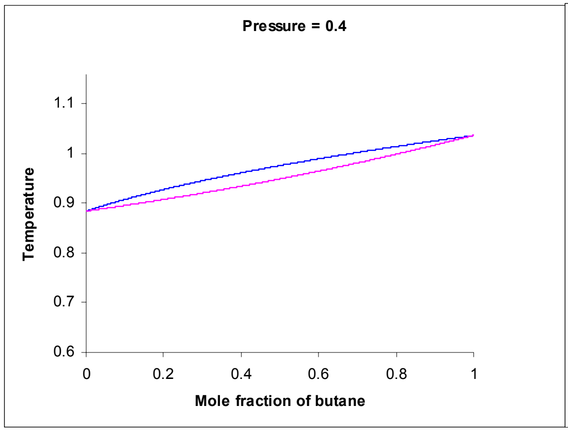
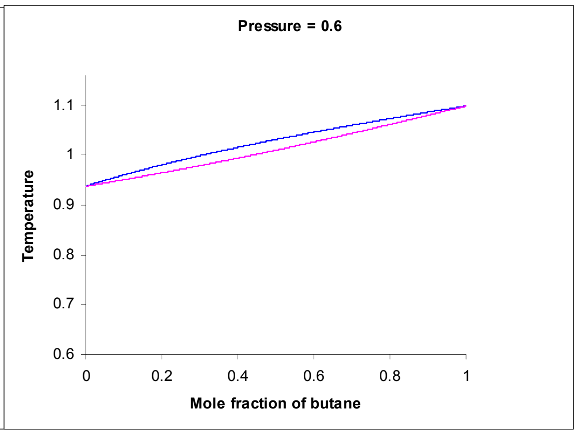


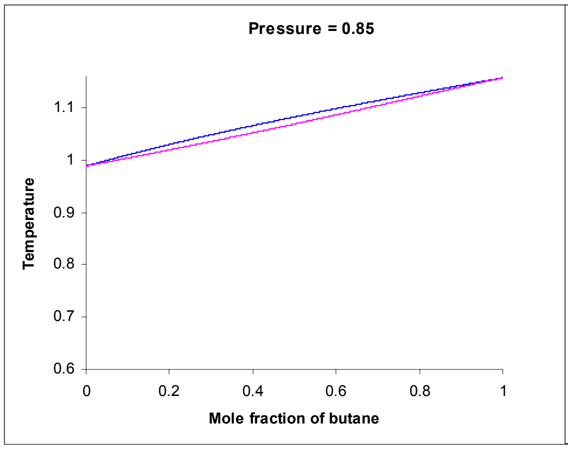
Figure 6.6: Temperature-Concentration Projection View of the Applet for the Butane-Methane Mixture



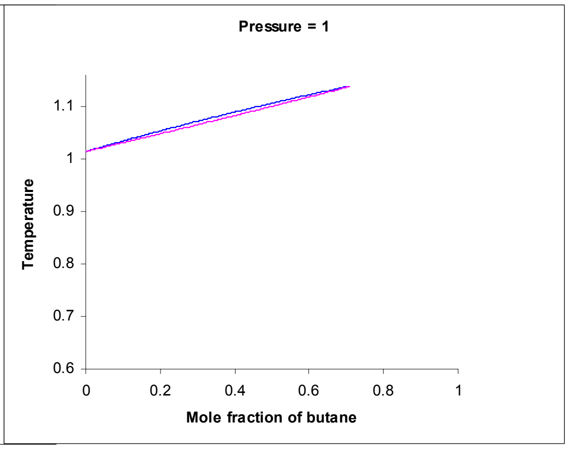
1



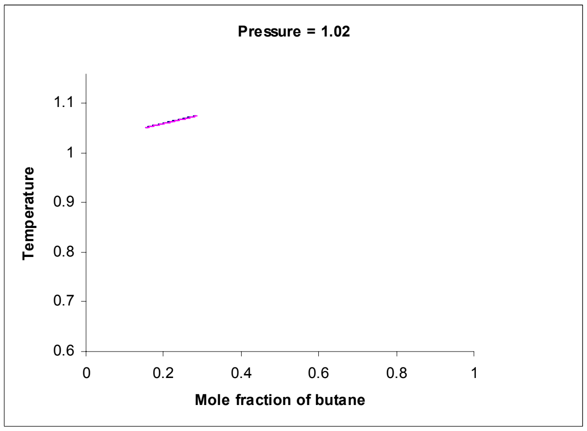
2



3



4



5

Figure 6.7: Individual Constant-Pressure Slices for the Butane-Propane Mixture

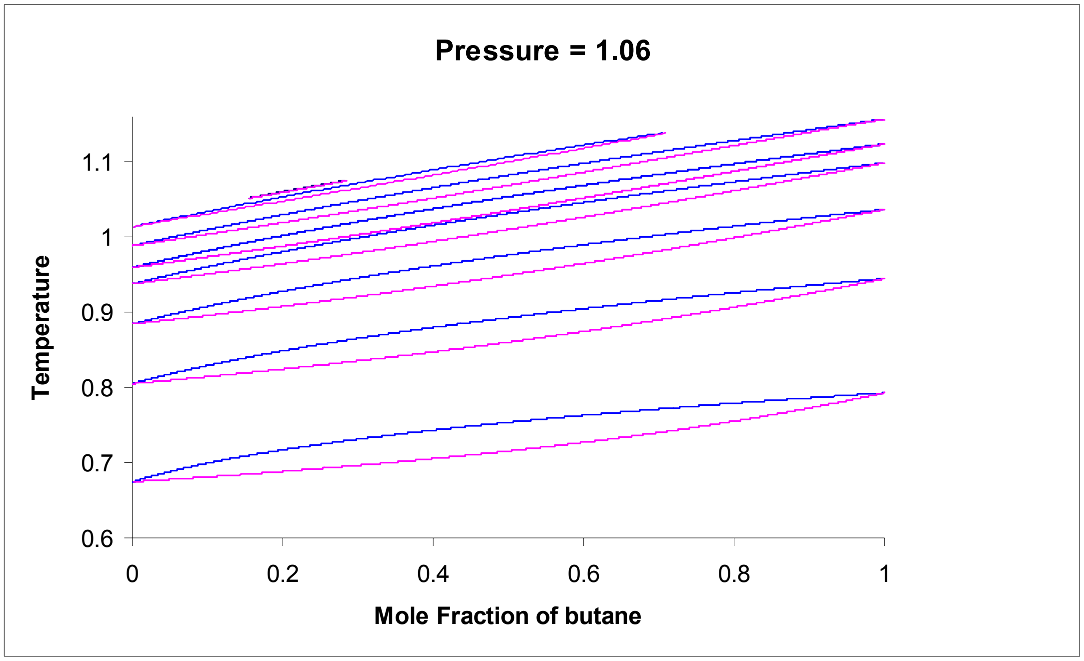


Figure 6.8: All Constant-Pressure Slices for the Butane-Propane Mixture

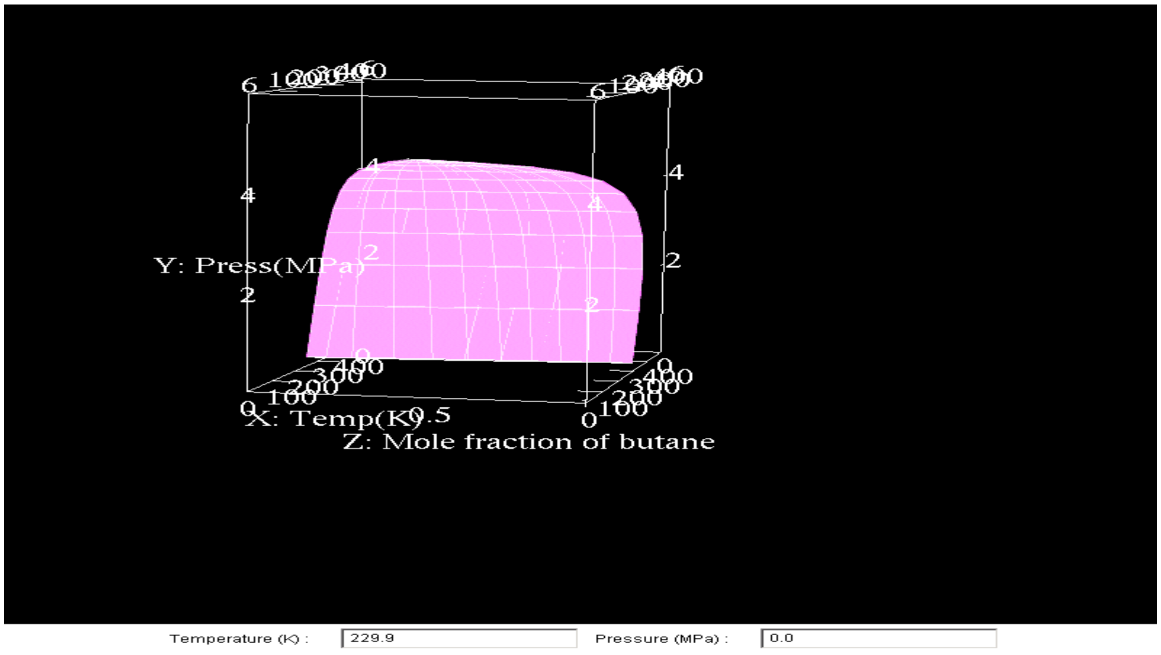
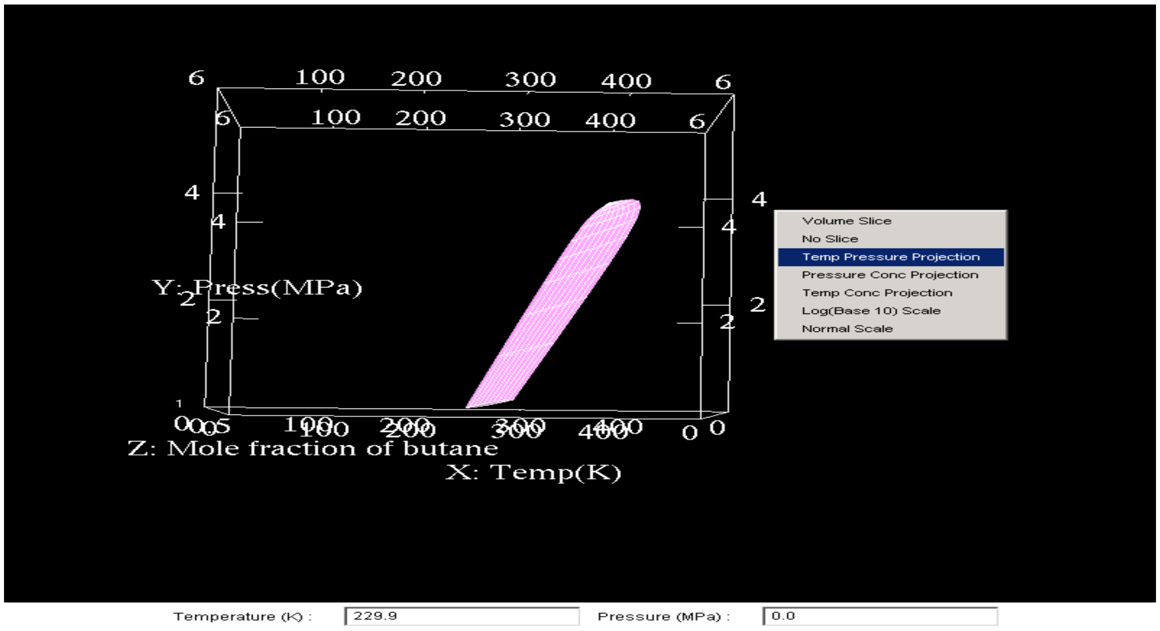
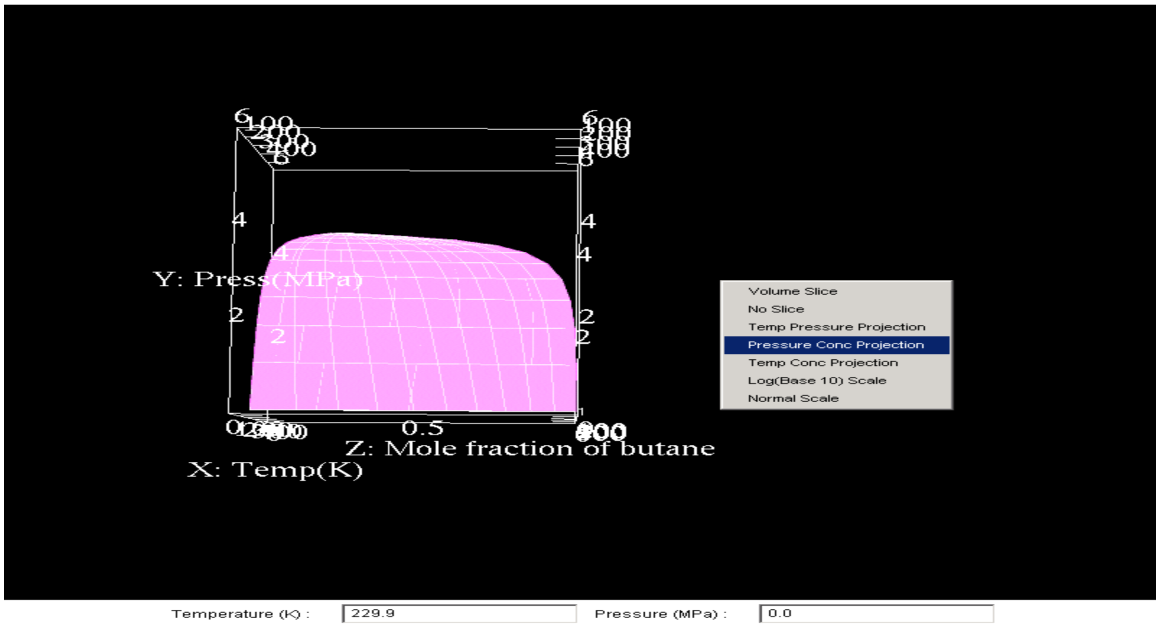


Figure 6.9: An Orthogonal View of the 3D Phase Diagram Applet for the Butane-Propane Mixture



Temperature (K) : 229.9 Pressure (MPa) : 0.0

Figure 6.10: Pressure-Temperature Projection View of the Applet for the Butane-Propane Mixture



Temperature (K) : 229.9 Pressure (MPa) : 0.0

Figure 6.11: Pressure-Concentration Projection View of the Applet for the Butane-Propane Mixture

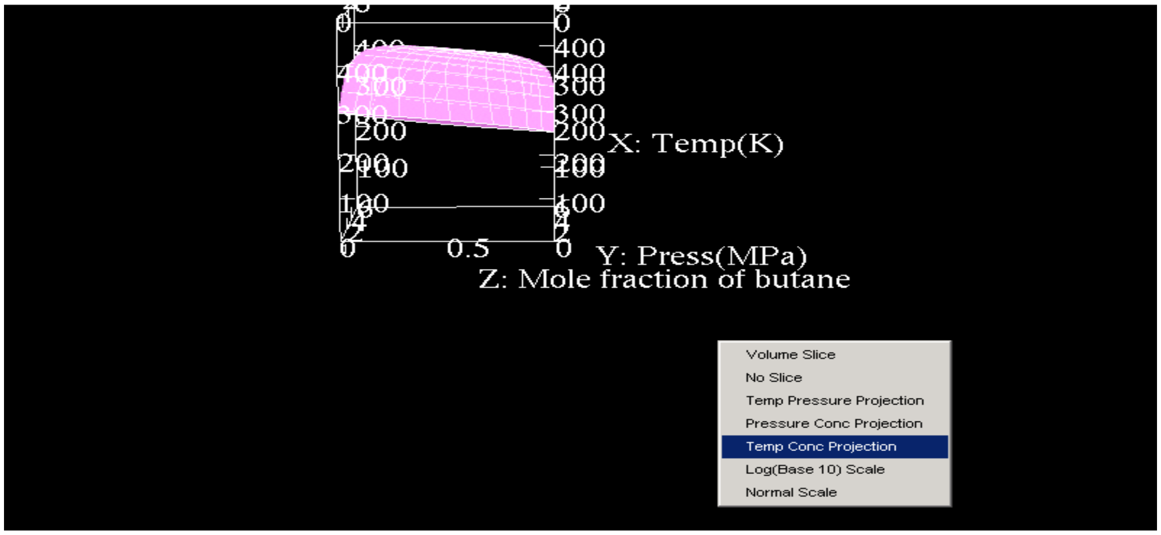
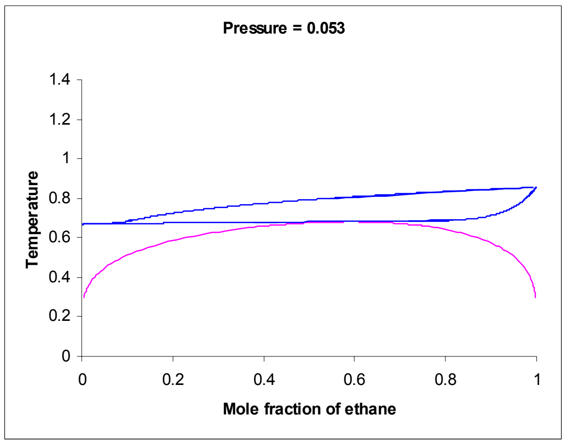


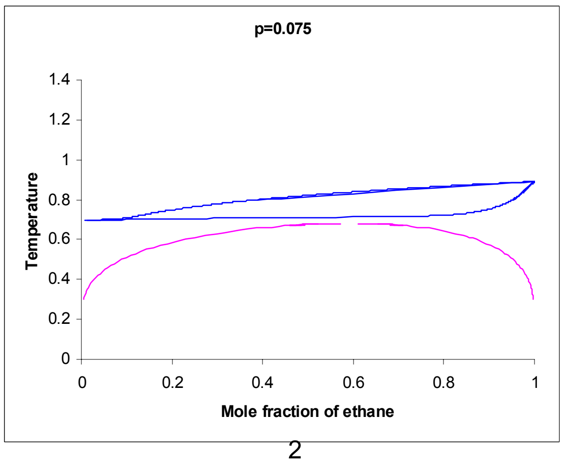
Figure 6.12: Temperature-Concentration Projection View of the Applet for the Butane-Propane Mixture

6.3 Type II (Perflouromethane + Ethane)

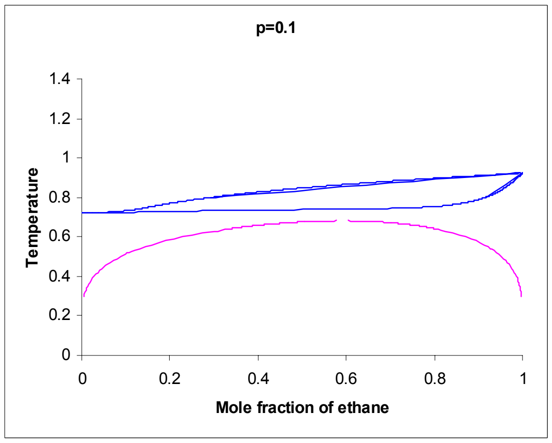
Type II phase diagrams are characterized by the existence of a region of liquid-liquid immiscibility at high pressure. Liquid-liquid immiscibility occurs when two liquids do not mix in all proportions – an extreme example is oil and water. As shown in Figure 6.13, the constant-pressure slices are plotted for the entire 3D phase diagram model for a perflouromethane-ethane mixture. The Figure 6.14 shows the all the constant-pressure slices for the same mixture in order to give a 3D perspective of the model. The Figure 6.15 shows an orthogonal view of the 3D phase diagram applet. The Figure 6.16 shows the *p-T* projection view, while the Figure 6.17 shows the *p-x* projection view, and the Figure 6.18 shows the *T-x* projection view of the applet for the perflouromethane-ethane mixture.



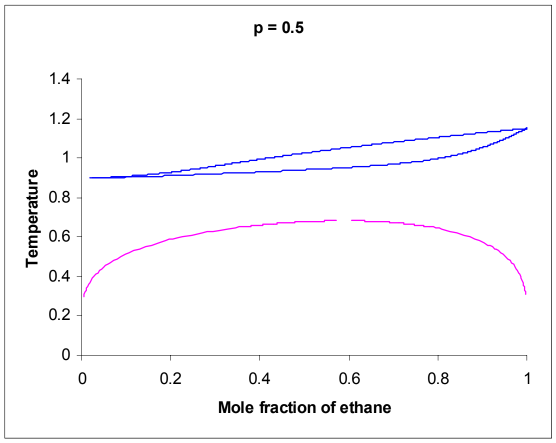
1



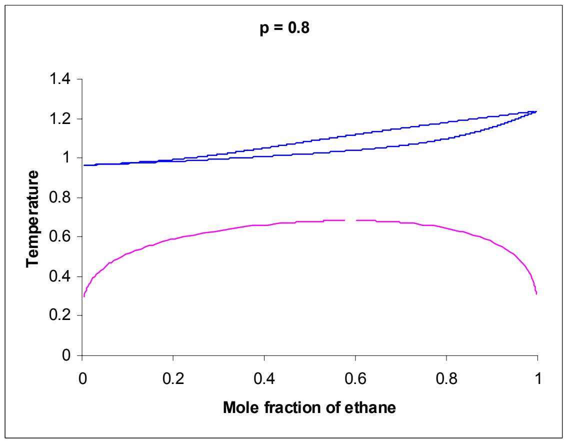
2



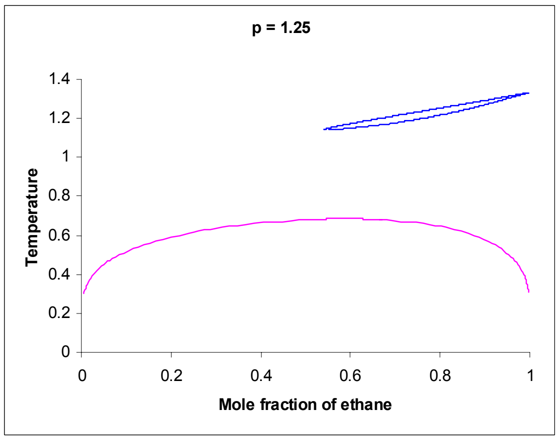
3



4



5



6

Figure 6.13: Individual Constant-Pressure Slices for the Perflouromethane-Ethane Mixture

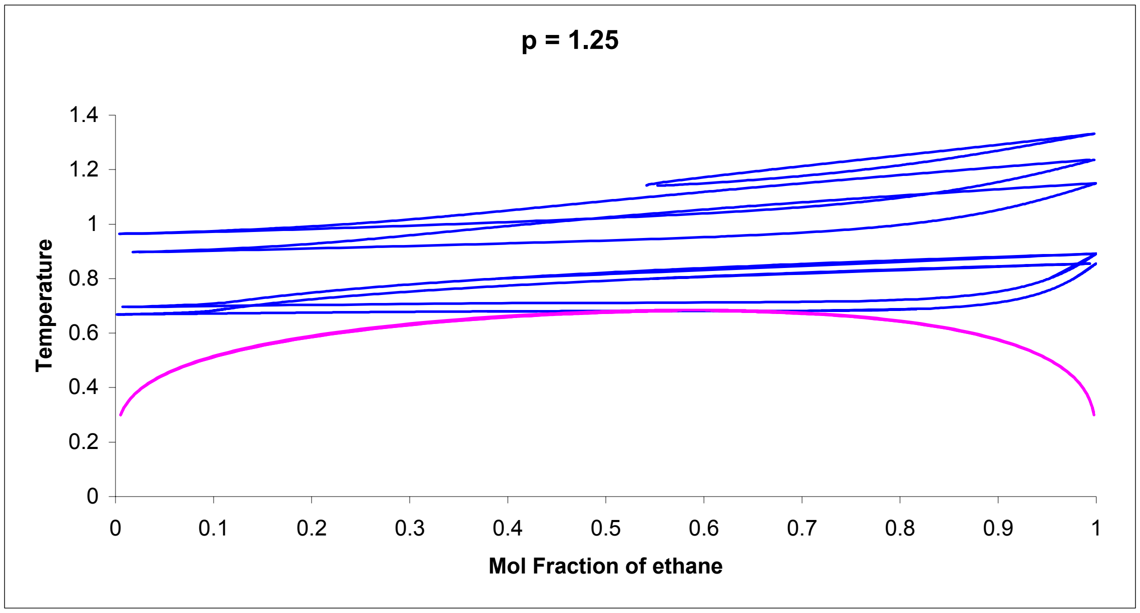
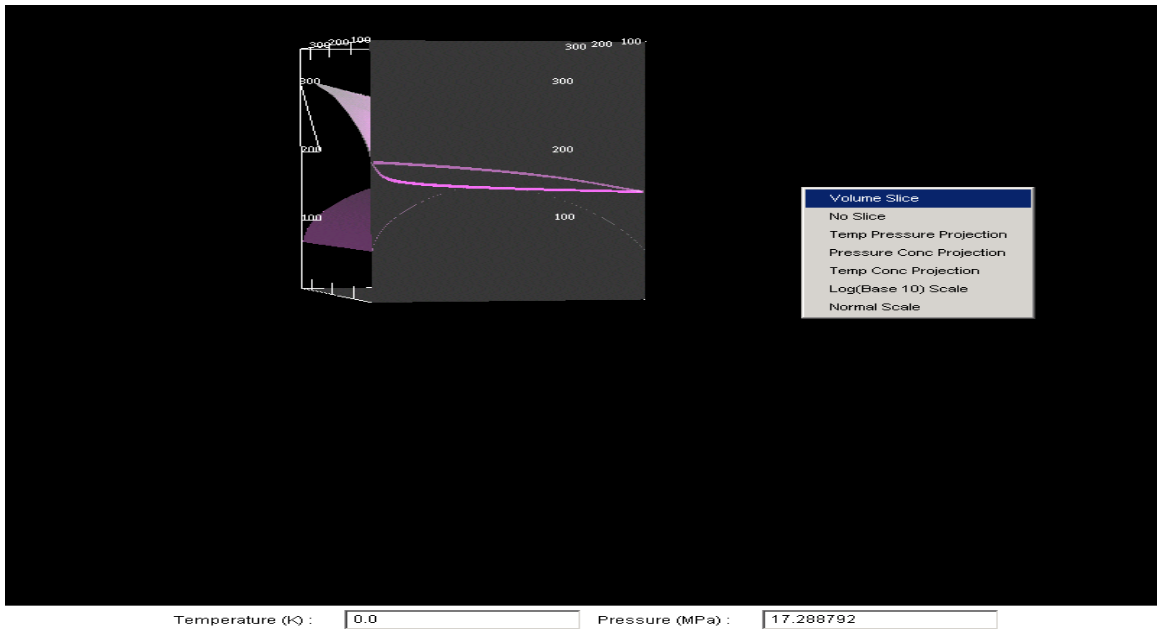
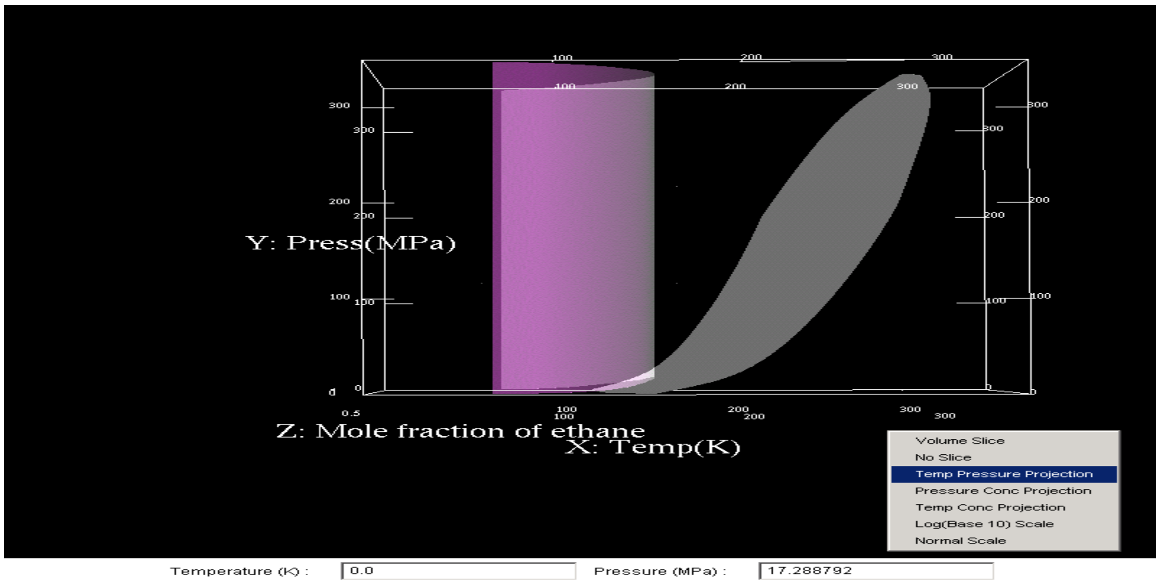


Figure 6.14: All Constant-Pressure Slices for the Perflouromethane-Ethane Mixture



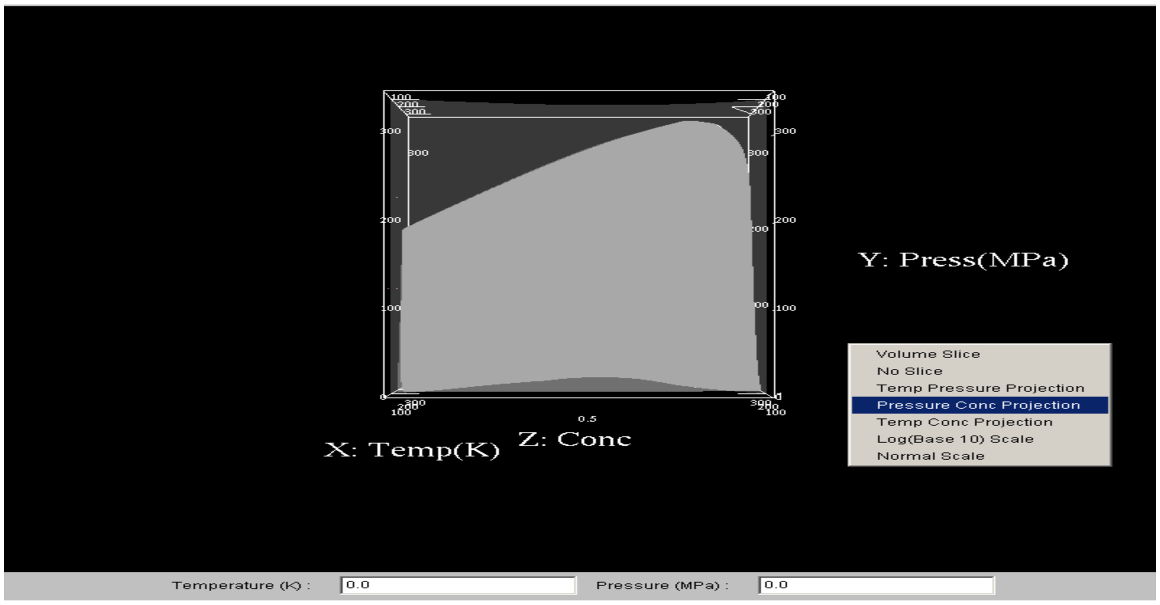
Temperature (K) : 0.0 Pressure (MPa) : 17.288792

Figure 6.15: An Orthogonal View of 3D Phase Diagram Applet for the Perflouromethane-Ethane Mixture



Temperature (K) : 0.0 Pressure (MPa) : 17.288792

Figure 6.16: Pressure-Temperature View of the Applet for the Perflouromethane-Ethane Mixture



Temperature (K) : 0.0 Pressure (MPa) : 0.0

Figure 6.17: Pressure-Concentration View of the Applet for the Perflouromethane-Ethane Mixture

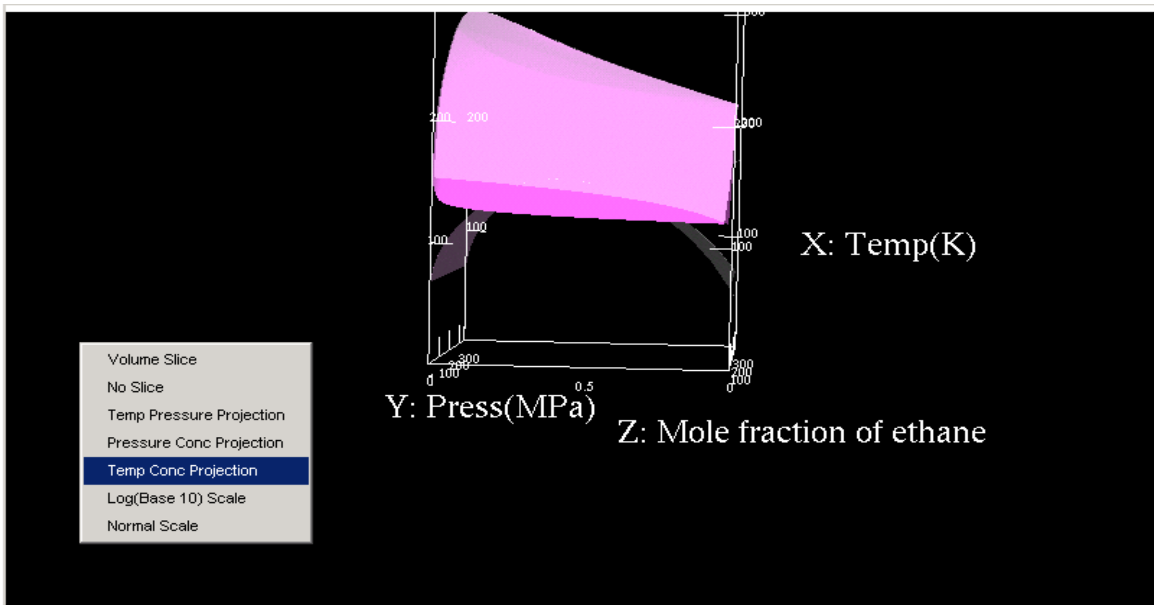


Figure 6.18: Temperature-Concentration View of the Applet for the Perflouromethane-Ethane Mixture

Part Three - Molecular Simulation Module – Chemical Potential of Binary Mixtures

7 Molecular Simulations

As part of the fourth and final module of this project, a molecular simulation module for measuring chemical potential in mixtures, using the particle swap method [15] has been developed. The module (pure Java applet) has been developed using the molecular simulation API, Etomica (detailed description in the Appendix A.3). Prof. Kofke was a collaborator on this section of the thesis.

7.1 Molecular Simulation Models

A molecular simulation is a computer experiment based on a molecular model. A molecular simulation API provides certain elements such as objects that represent molecules (diameter, mass, shape), potentials that describe interactions (e.g. collisions) between molecules and methods for measuring the physical conditions of various configurations or states of the simulation [16].

There are two widely used molecular simulation models:

1. **Molecular Dynamics (MD)**

   Molecular Dynamics is a deterministic method that integrates the equations of motion to examine the evolution of a system over a period of time.

2. **Monte Carlo (MC)**

   Monte Carlo simulations are stochastic processes that do not have an element of time. Instead they use ensemble averaging to generate a large number of randomly selected configurations of a system.

In both MD and MC simulations, averages are taken over the ensemble of configurations, and these can be used to “measure” the physical properties of the model system.

7.2 Advantages and Disadvantages of Molecular Simulation

Advantages

- Simulations are relatively inexpensive compared to normal lab experiments. Compared with the actual scientific experiments that require expensive instruments, molecular simulations require relatively much cheaper computational power. While most experiments would require months of man-hours to conduct the experiments, a molecular simulation could accomplish the same task in a matter of hours, if not in minutes.
- Using molecular simulations, researchers can conduct simulations on potentially hazardous or toxic chemicals without causing any harm to the environment.
- Molecular simulations provide molecular-level insight into macroscopic phenomena. Such insight is difficult at best to obtain experimentally (e.g., it may require complex experiment such as neutron scattering) and impossible to obtain experimentally in some cases.

Disadvantages

- Since simulations are based on molecular models, it is imperative to use a model accurately describing the system. An inaccurate model representing the system may result in incorrect understanding.

- Depending upon the simulations that are conducted, extremely high levels of computational power may be required to obtain the necessary results. Presently, simulations are limited to systems and phenomena that extend over micron length and nanosecond time scale. However, with the increase in the availability of higher computational power, the gap between the available and required computational power is gradually decreasing.

7.3 Chemical Potential

According to [17], the chemical potential determines phase equilibria and is an important quantity to determine by molecular simulation. Various methods exist for calculating the chemical potential for pure substances. The equation of state can be numerically integrated from a reference state to derive the chemical potential [17]. However, here we are interested in obtaining it from a molecular simulation.

7.3.1 Widom Insertion Technique

Attention has largely been concentrated on the particle insertion method of Widom, as this method can be incorporated into a simulation code with minimal effect on performance. Using the Widom insertion method, the chemical potential for pure fluid is computed by performing a number of test insertions of a similar molecule. A new molecule (of the same species) is inserted at a random position and the potential energy (U) experienced by the inserted particle is measured. The chemical potential (μ) measured over 'n' test insertions is given by:

$$\mu = kT \ln < \exp(-{}^{U}\!/_{kT}) >$$

where, '*k*' denotes the Boltzmann constant and '*T*' denotes the Temperature. The Widom insertion method to compute potential can be extended to mixtures and multi-component systems too. However, for dense systems, direct particle-insertion methods are known to yield poor statistics [15]. In dense systems successful trial insertions of test particles are rare events.

7.3.2 Particle Swap Technique

An alternative way of measuring chemical potential, as suggested by Reference [15] is by using the particle swap method. Instead of sampling the potential energy changes resulting from particle insertion or removal, the potential energy changes associated with the virtual transformation of a particle of species A into a particle of species B is sampled. For a two-component mixture of species A and B, such as Argon-Kryton system, this method directly provides values of the differences between excess chemical potentials (excess values with respect to the perfect gas values). The chemical potential is calculated as follows:

$$\begin{aligned}\Delta\mu &= \mu_A - \mu_B \\ &= -\beta^{-1}\ln\left\langle \exp(-\beta\Delta U^{A+B-})\right\rangle \\ &= \beta^{-1}\ln\left\langle \exp(-\beta\Delta U^{A+B-})\right\rangle\end{aligned}$$

where ΔU^{A+B-} denotes the change in potential energy that results from the transformation of a B particle into an A particle, likewise ΔU^{B+A-} for an A particle. Also, $\beta = kT$ where, '*k*' denotes the Boltzmann constant and '*T*' denotes the Temperature. In this equation, the symbol $\langle ... \rangle$ indicate an average of the quantity over the simulation.

8 Analysis and Implementation

8.1 Analysis

The development of the molecular simulation module for measuring chemical potential required the following:

1. Developing a simulation module using the various classes in Etomica.
2. Implementing a Meter Class, measuring the chemical potential using the Particle Swap technique described earlier.

8.1.1 Simulation Elements

The Table 8.1 describes the various elements of Etomica that were used are described, along with a brief description about the manner in which they were used in the development of the chemical potential module. Refer to Appendix A.3 for a detailed description of the Etomica class files.

8.1.2 Data Flow Diagrams

The Data Flow Diagrams represent an overview of the system that was developed.

8.1.2.1 Context-Level Diagram (CLD)

As shown in Figure 8.1, the Context-Level Diagram represents an overview of the system (applet) that was developed

- Student and Etomica-Molecular Simulation API, represent the external entities of the system.
- Student interacts with the various class files in Etomica, using the graphical web based interface provided by the applet.

The Student specifies the various simulation parameters and runs the simulation. Etomica displays the results in the form of tables, charts etc.

Table 8.1: Etomica Elements and the Corresponding Implementation in the Module

Etomica Element	**Implementation of Chemical Potential Module**
Simulation – Main class holding the various elements of the simulation	An instance of the Simulation class was created
Phase – Container of molecules that interact with each other	A Phase Class was instantiated
Space – physical framework that provides the ability to create vectors, tensors, boundaries (periodic/non-periodic boundary conditions), and coordinates	Since this was a 2D simulation, an instance of Space2D was created. Periodic boundary conditions were used for the simulation.
Integrator – Generates the new configurations of the phase by defining the movement of atoms/molecules.	Etomica doesn't have an NPT integrator. Therefore, the Gear4NPH integrator, which approximates an NPT simulation, was used instead.
Controller – Controller controls all actions performed by the integrators in a smulation. (includes the start/stop button)	The controller class was extended to create a new controller class (ControllerJT). Using the new controller class, the simulation starts off with the NVE integrator, runs for a maximum of 100 steps and transfers the phase to the Gear4NPH integrator. It displays the

Table 8.1 Continued

Etomica Element	**Implementation of Chemical Potential Module**
	start/stop button on the applet.
Species – A Species is a collection of identically formed Molecules. Molecules can consist of one or more Atoms.	SpeciesSpheresMono class, species in which molecules are made of a single spherical atom were used
	▪ Species A $\Rightarrow$ Black colored species ▪ Species B $\Rightarrow$ Blue colored species
Potential – The potential class defines the interactions between Atoms	Lennard-Jones inter-atomic potential, spherically symmetric potential of the form $U(r) = 4 * \varepsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}]$ $\varepsilon \Rightarrow$ Strength of the pair interaction, $\sigma \Rightarrow$ Atom size parameter, specified the interaction between species A and species B molecules. Three interactions are required: A-A, B-B and A-B.
Display – View the results of the	1. An instance of DisplayPhase class

Table 8.1 Continued

Etomica Element	**Implementation of Chemical Potential Module**
simulation in the form tables, plots of data curves, molecular movements etc.	was created to display the molecular movements. A scale slider modified the scale of the display.
	2. A DisplayPlot displaying the rdf values for AA, AB, BB interactions 3. A DisplayTable class displayed the results of the simulation in a table. 4. DisplayBoxes were used to set the following parameters in both Real and Lennard-Jones Unit: ▪ Pressure ▪ Temp ▪ Density ▪ Mass of a species ▪ Sigma (σ) for a species ▪ Epsilon (ε) for a species
Modulator – An interface to make changes to the properties of another object.	An instance of the Modulator class was created to link the “scale” property of

Table 8.1 Continued

Etomica Element	**Implementation of Chemical Potential Module**
	the DisplayPhase with the scale slider.
***Device* – Devices provide graphical interfaces for modifying these fields.**	1. A DeviceSlider was created to change the mole fraction of the species. 2. DeviceButtons were used to ▪ Restart the simulation ▪ Reset the running averages of the simulation.
Meter – Meters measure various physical properties of a Phase.	1. A new meter called MeterParticleSwap, was created for measuring the chemical potential using the particle swap technique. 2. Meters for measuring radial distribution function (rdf) values for A-A, B-B, and A-B interactions between molecules were used. 3. Meters were created for measuring ▪ Pressure ▪ Temperature

Table 8.1 Continued

Etomica Element	Implementation of Chemical Potential Module
	▪ Density ▪ Configurational Energy ▪ Chemical potential using ▪ Widom Insertion technique

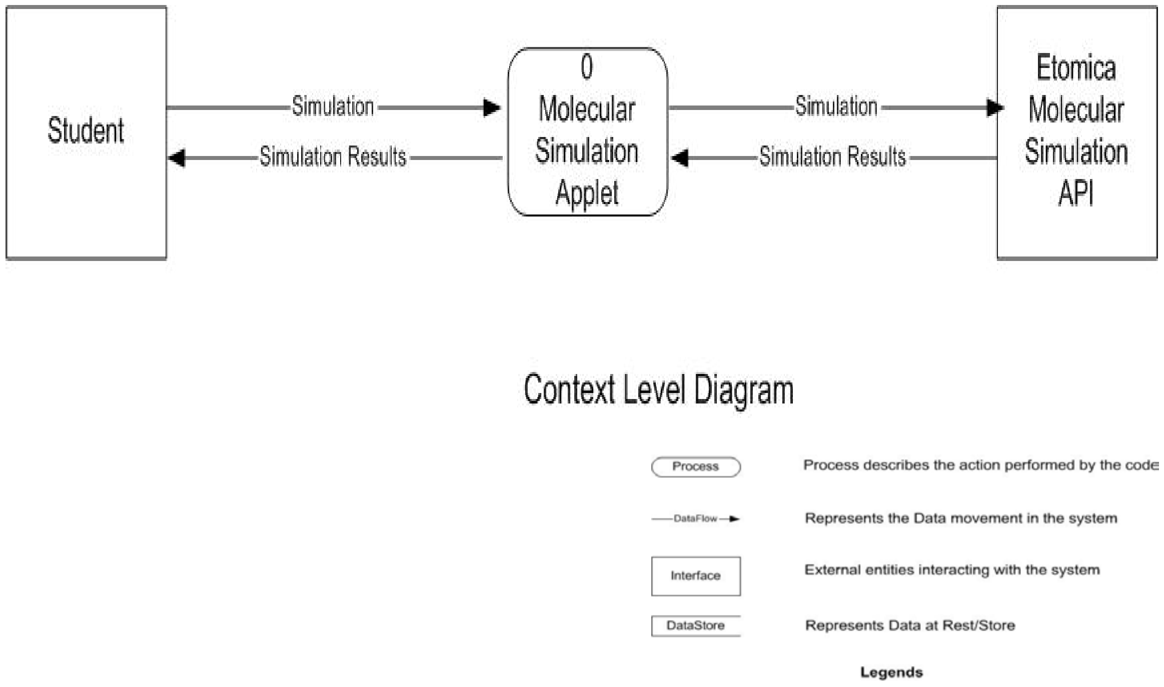

Student
Simulation
Simulation Results
0
Molecular Simulation Applet
Simulation
Simulation Results
Etomica Molecular Simulation API
Context Level Diagram
Process
Process describes the action performed by the code
DataFlow
Represents the Data movement in the system
Interface
External entities interacting with the system
DataStore
Represents Data at Rest/Store
Legends

Figure 8.1 Context-Level Diagram of the Molecular Simulation Module

8.1.2.2 Level-1 Diagram

The Level-1 Diagram, as shown in Figure 8.2, shows the interaction of the Student with the Molecular Simulation Applet (Process 1.0). The student can specify the following:

- Select the mole fraction of species A and species B by using a Jslider (Process 1.1)
- Set the initial conditions (Process 1.2) by setting the Pressure, Temperature and Density values in Real/Lennard-Jones Units.
- Choose the types of species (Process 1.3) by using a drop down list. Specify the Mass, Epsilon and Sigma values for the species.
- Control the progress of the simulation (Process 1.4) by starting, stopping, resetting and restarting the simulation.
- Specify the scale of display (Process 1.5) of the molecular movements.

The Data Flow Diagram, depicted in Figure 8.3, shows the interaction of the Molecular Simulation Applet with the various class files in Etomica.

- An instance of the Simulation, Phase, Space2D, Controller, Integrator, Modulator class are created.
- Since the applet measures the chemical potential for binary mixtures, an instance of species A and species B is created using the Species class.
- Several Meters: MeterParticleSwap, MeterRDF, MeterPressure, MeterDensity, MeterTemperature, MeterConfigurationalEnergy and MeterWidomInsertion are created for measuring various simulation results.
- DeviceSlider and DeviceButton instances are created to manipulate the user input.
- The results are displayed using the DisplayTable, DisplayPhase, DisplayBox and DisplayPlot classes.

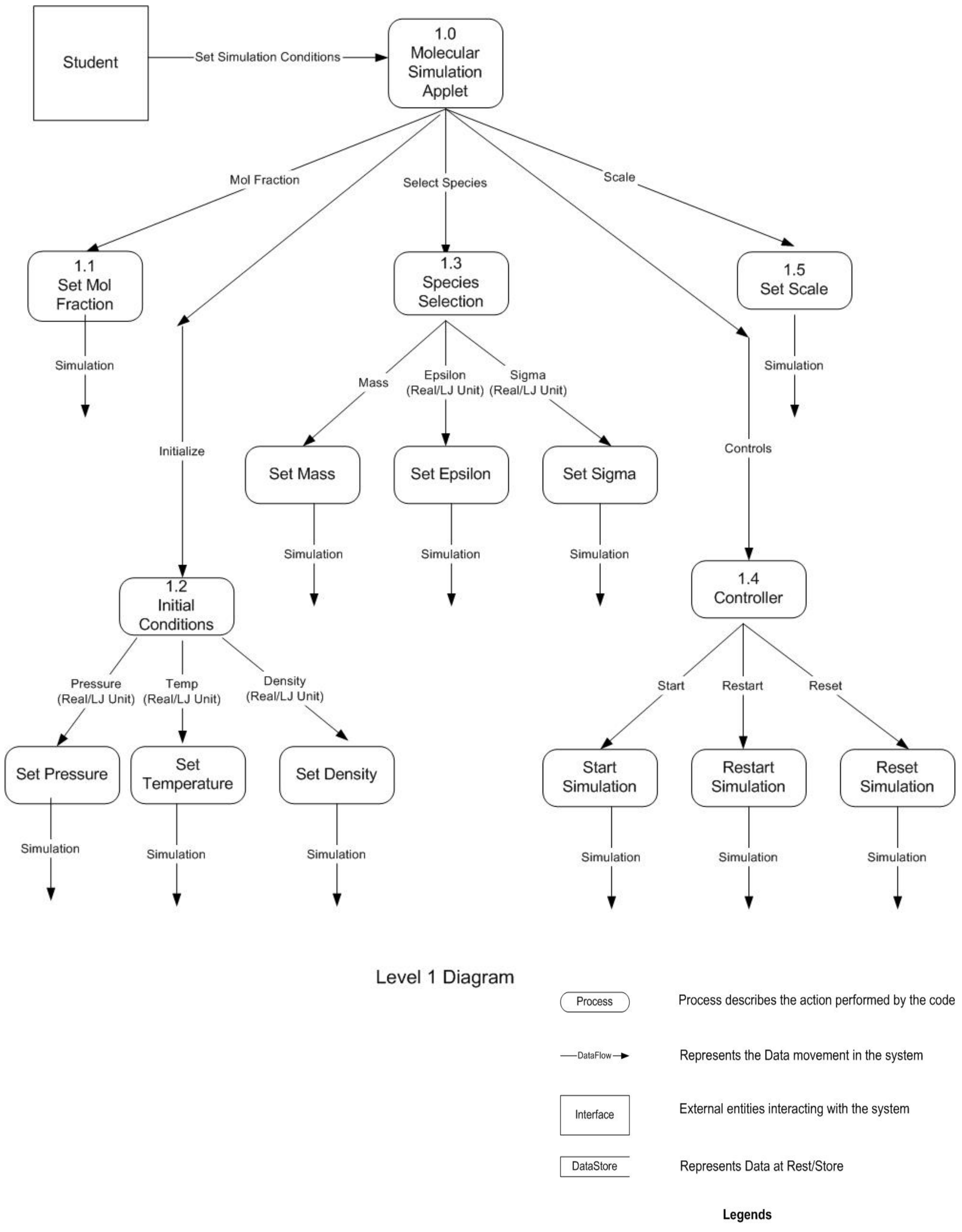


Figure 8.2: Level-1 Diagram of the Molecular Simulation Module (Part 1)

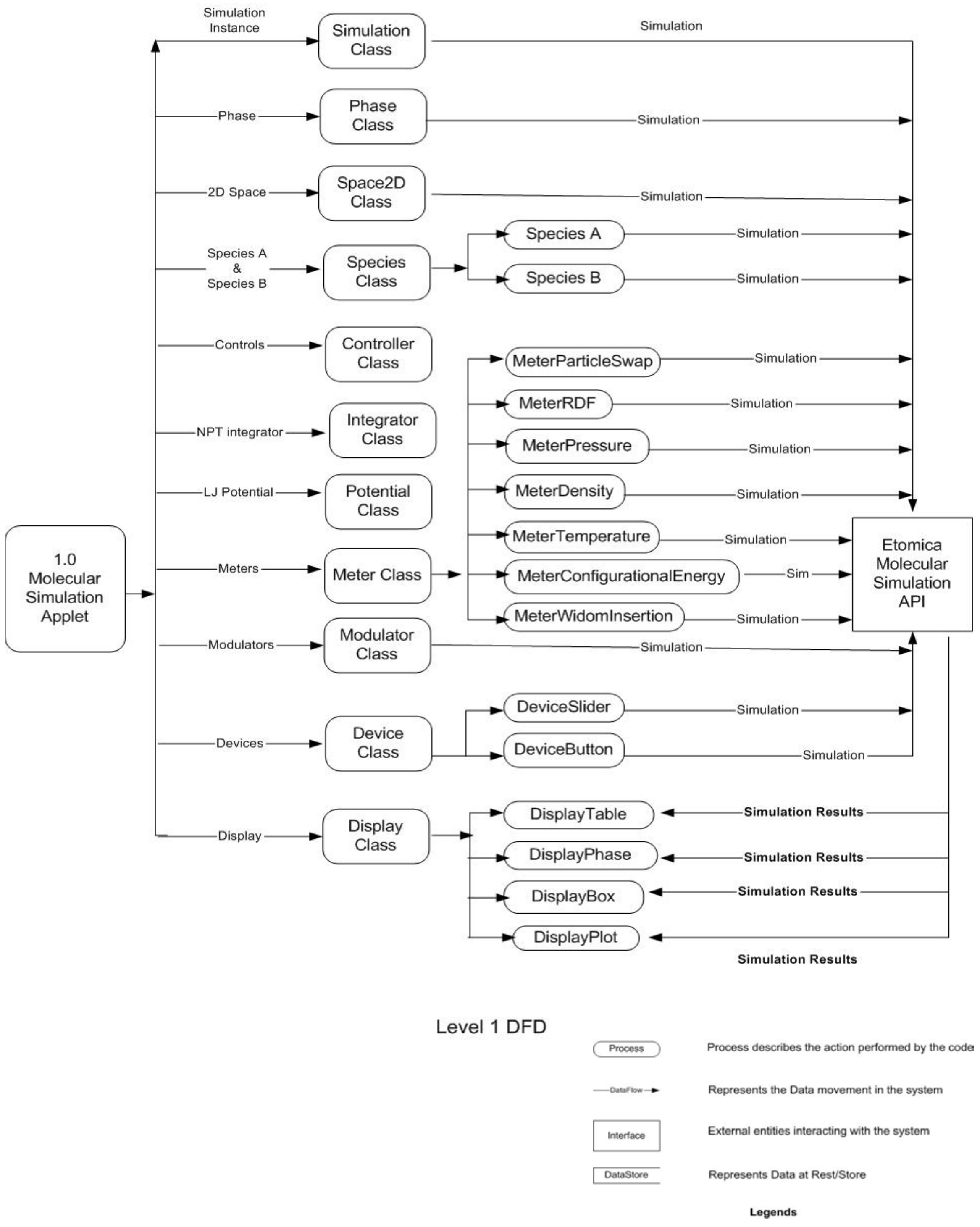


Figure 8.3: Level-1 Diagram of the Molecular Simulation Module (Part 2)

8.2 Implementation – Class Diagram

The class diagram, Figure 8.4, gives an overall picture of the various java class files developed as part of the system. Also, the pseudo-code implementation, described in Appendix A.2, gives an overview of the various methods, constructors etc. used in the class files.

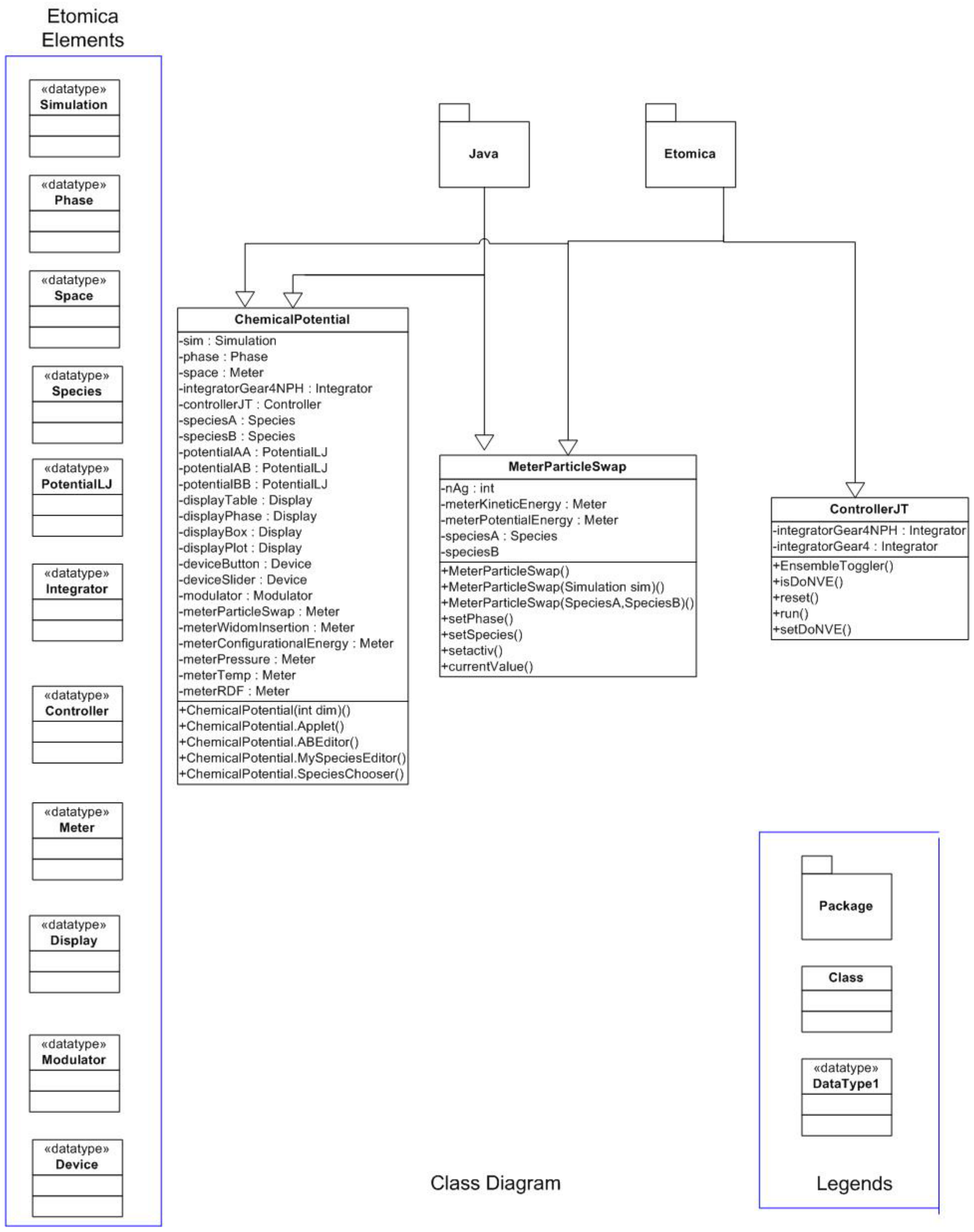
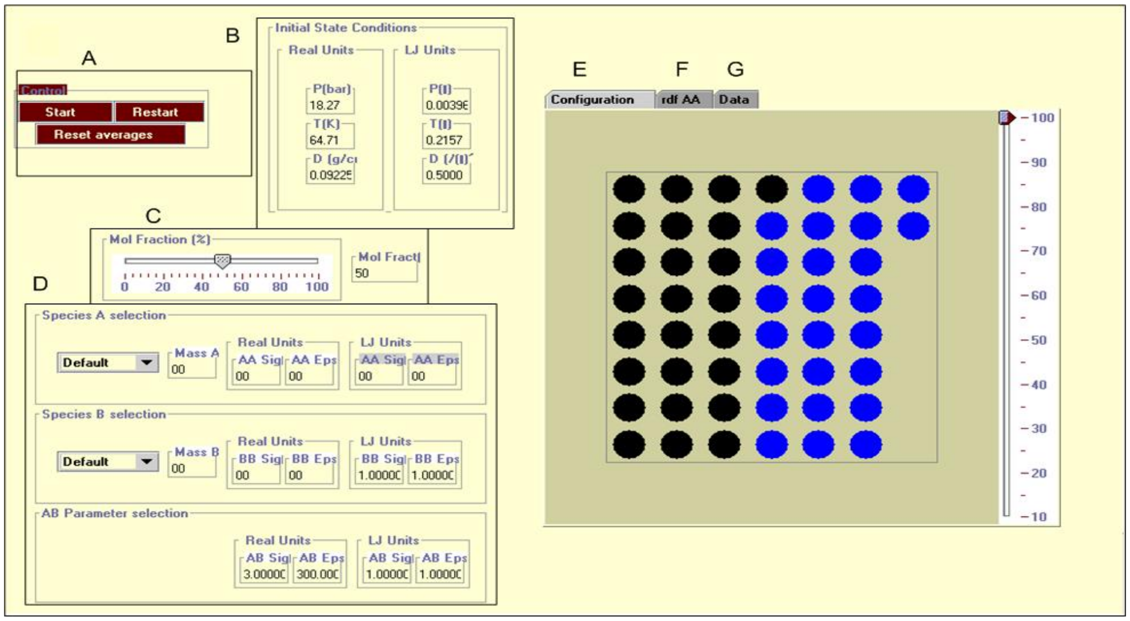


Figure 8.4: Class Diagram of the Molecular Simulation Module

9 Results and Discussions

9.1 Molecular Simulation Applet

The Figure 9.1 shows a screenshot of the molecular simulation applet for measuring chemical potential of binary mixtures. Referring to Figure 9.1, A is used to set the controls (start/stop/restart) for the simulations, B is used to set the initial simulation conditions (pressure, temperature, density in real/Lennard-Jones unit), C is used to set the mole fraction of the mixture, D is used to select the species A/B, set the sigma and epsilon values for the A-A, A-B and B-B interactions, E is used to display the molecular simulation, F is used to display the rdf plots while G is used to display (in the form of a table) the chemical potential values of the mixture, measured by the Widom and particle swap methods.



Initial State Conditions

Real Units: P(bar) 18.27; T(K) 64.71; D (g/cı 0.09225

LJ Units: P(I) 0.00396; T(I) 0.2157; D (/(I)ʼ 0.5000

Mol Fraction (%); Mol Fract 50

Species A selection: Default; Mass A 00; Real Units: AA Sig 00, AA Eps 00; LJ Units: AA Sig 00, AA Eps 00

Species B selection: Default; Mass B 00; Real Units: BB Sig 00, BB Eps 00; LJ Units: BB Sig 1.0000C, BB Eps 1.0000C

AB Parameter selection: Real Units: AB Sig 3.0000C, AB Eps 300.00C; LJ Units: AB Sig 1.0000C, AB Eps 1.0000C

Figure 9.1: Screenshot of the Molecular Simulation Applet for Measuring the Chemical Potential of Binary Mixtures

9.2 Testing

The chemical potential is calculated as follows:

$$\begin{aligned}\Delta\mu &= \mu_A - \mu_B \\ &= -\beta^{-1}\ln\left\langle \exp(-\beta\Delta U^{A+B-})\right\rangle \\ &= \beta^{-1}\ln\left\langle \exp(-\beta\Delta U^{A+B-})\right\rangle\end{aligned}$$

where ΔU^{A+B-} denotes the change in potential energy that results from the transformation of a B particle into an A particle, likewise ΔU^{B+A-} for an A particle. Therefore, theoretically the chemical potential measured by swapping an A molecule with a B molecule, should give the same value (though opposite in sign) as the chemical potential measured by swapping a B molecule with an A molecule. The results were tested with the Fortran code results generated by another member of our research group, Dr. Ariel Chialvo. For an Argon-Argon (ideal case) mixture, the Table 9.1 shows the simulation conditions. The simulation results are shown in Table 9.2. The resulting Radial Distribution Function plots are shown in the Figure 9.2. Results for r >= 4.8 are not meaningful since this is half of the box length of the simulation. Over the meaningful range, the applet and Fortran code are giving essentially the same result.

For an Argon-Krypton mixture, Table 9.3 shows the simulation conditions. The simulation results are shown in Table 9.4. The resulting Radial Distribution Function plots are shown in the Figure 9.3. The applet results are very similar to those from the Fortran code. The latter are smoother and better defined from having been run longer.

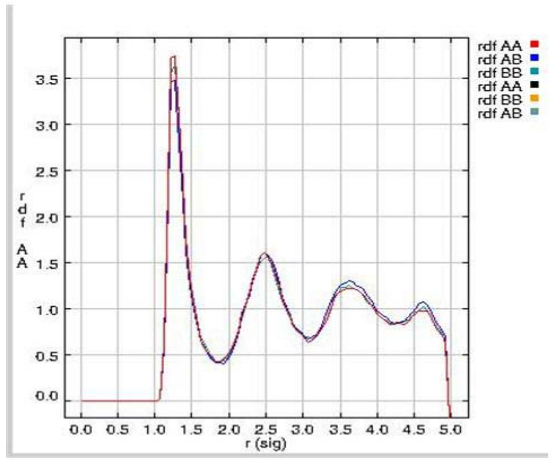
Table 9.1: Argon-Argon Simulation Conditions

Simulation Condition	**Set Value**
Pressure	4.59 Bar
Temperature	53.91 Kelvin

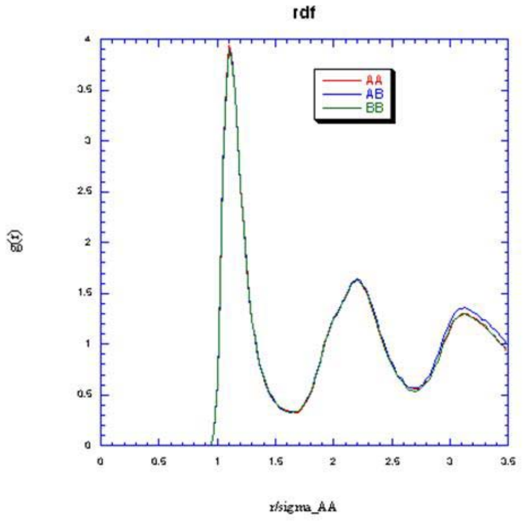
Table 9.2: Argon-Argon Simulation Results

Simulation Condition	**Unit**	**Module Result**	**Fortran Code Result**
Number Density	$(\frac{}{\sigma^2})$	0.725	0.772
Configurational Energy	(ε)	-2.35	-2.55
Chemical Potential- *Widom Insertion*- (Inserting A molecule)	(ε)	-1.65	-1.8
Chemical Potential- *Widom Insertion*- Inserting B molecule	(ε)	-1.75	-1.8
Chemical Potential – *Particle Swap technique* – (A molecule swapped with B)	(ε)	0	0
Chemical Potential – *Particle Swap technique* – (B molecule swapped with A)	(ε)	0	0

Note: The value of the chemical potential measured by the particle swapping technique should be approximately equal to the numerical difference of the chemical potential values measured by inserting A and B molecules.



Applet Results



Fortran Code Results

Figure 9.2: Comparison of the Applet RDF Plots for the Argon-Argon Mixture and the Fortran Code RDF Plots

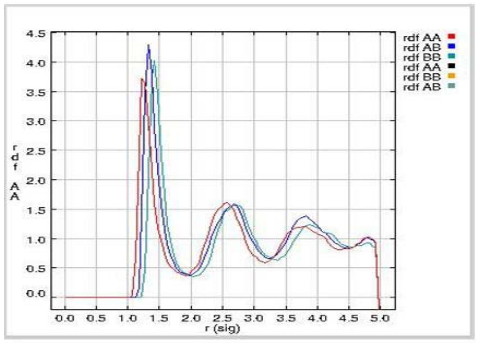
Table 9.3: Argon-Krypton Simulation Conditions

Simulation Condition	**Set Value**
Pressure	21.282 Bar
Temperature	53.91 Kelvin

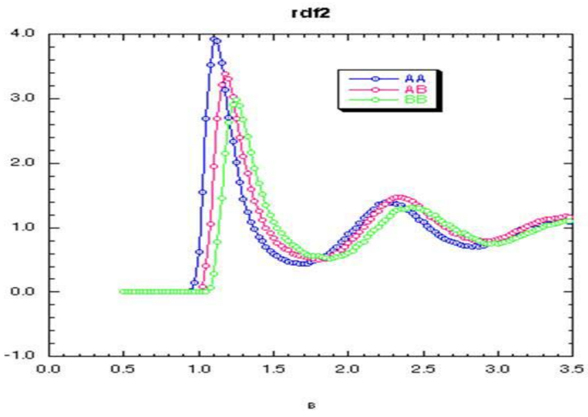
Table 9.4: Argon-Krypton Simulation Results

Simulation Condition	**Unit**	**Module Result**	**Fortran Code Result**
Number Density	$(\diagup_{\sigma^2})$	0.6	0.587
Configurational Energy	(ε)	-2.82	-2.557701
Chemical Potential- *Widom Insertion*- (Inserting Ar molecule)	(ε)	0.95	Not Available
Chemical Potential- *Widom Insertion*- Inserting Kr molecule	(ε)	-1.15	Not Available
Chemical Potential – *Particle Swap technique* – (Ar molecule swapped with Kr)	(ε)	-1.302	-1.1083
Chemical Potential – *Particle Swap technique* – (Kr molecule swapped with Ar)	(ε)	-2.317	-2.275

Note: The value of the chemical potential measured by the particle swapping technique should be approximately equal to the numerical difference of the chemical potential values measured by inserting A and B molecules. The differences between the Fortran code and the applet can be attributed to differences in the numbers of molecules and in the way the potentials are cut off.



Applet results



Fortran code results

Figure 9.3: Comparison of the Applet RDF Plots for the Argon-Krypton Mixture and the Fortran Code RDF Plots

10 Conclusions

The four educational modules have been successfully developed and tested under the Windows, UNIX and Mac platforms. The modules have been uploaded to the website: http://www.cs.utk.edu/~gupta/ . The website provides a link to the necessary plug-ins required for viewing the phase diagram applets on different platforms.

There are six types of phase diagrams, of which we have developed phase diagrams for types I and II. As part of the future work, applets for types III-VI need to be developed. However, the computer programs developed as part of this thesis can be used for the future development process. Also, the programs can be extended further to incorporate visualization of any kind of 3D data.

Last but not the least, the applets need to be incorporated into educational modules. These modules would require adequate documentation, relevant background information, examples, problem questions, etc. Once these modules are adequately documented, professors can use them for assigning homework problems to students. On the other hand, students can use these simple and interactive modules in order to gain a better understanding of complex phenomena.

The graphics capabilities of OpenGL and the platform independence of Java indeed provide a promising future for the development of web based education modules. It is hoped that these web-based educational modules would make this package a useful instructional aid.

LIST OF REFERENCES

1. C.McCabe, Colorado School of Mines, 2003.

2. T.Naps, J.B., R.Jimenez-Peris, M.McNally, *Using the www as the delivery mechanism for interactive, visualization-based instructional modules.* ITiCSE97 Working group on visualization, 97.

3. K.Engel, D.I., *Visualizing chemical data in the internet - data driven and interactive graphics.* Computer & Graphics, 1998. **22**(6).

4. Kamthan, P., *Java Applets in Education*, in *Internet Related Technologies*. 1999.

5. K. Tian, K.J., W.Chapman, *3D visualization of phase diagrams.* 2001.

6. Angel, E., *Interactive Computer Graphics: A Top-Down Approach with OpenGL.* 1999.

7. M.Jern, *3D Data Visualization on the Web.* CODATA Euro-American Workshop Visualization of Information and Data: Where We Are and Where Do We Go From Here?, 1997. **24-25**.

8. K.Engel, R.W., T.Ertl, *Isosurface extraction techniques for Web-based Volume Visualization.* IEEE Visualization conference, 1999(October): p. 25-28.

9. Sun, http://sun.java.com 2001.

10. Day, B., *3D graphics programming in Java*. 1998.

11. Goethel, S., *GL4Java*. http://www.jausoft.com/gl4java, 2001.

12. A.Dix, G.E., *By chance - enhancing interaction with large data sets through statistical sampling.* Advanced Visual Interfaces, 2002: p. 167-176.

13. Graphics, S., http://www.opengl.org, *OpenGL Programming Guide.*

14. M.Sarfraz, *A rational cubic spline for the visualization of monotonic data.* Computers & Graphics, 2000. **24**(4): p. 509-516.

15. P.Sindzingre, C.M., G.Ciccotti, *Calculation of partial enthalpies of an argon-krypton mixture by NPT molecular dynamics.* Chemical Physics, 1989. **129**: p. 213-224.

16. Mihalick, B., *Development of a standalone Java-based molecular simulation environment*, in *Chemical Engineering*. 2001, University of New York: Buffalo.

17. D.M.Heyes, *Chemical Potential, Partial Enthalpy and partial volume of mixtures by NPT molecular dynamics.* Chemical Physics, 1992. **159**: p. 149-167.

18. Clare McCabe, A. Galindo, G, Jackson**,** *Predicting the High-Pressure Phase Equilibria of Binary Mixtures of Perfluoro-n-alkanes n-Alkanes Using the SAFT-VR Approach.* Chemical Physics, 1998. **102**

APPENDIX

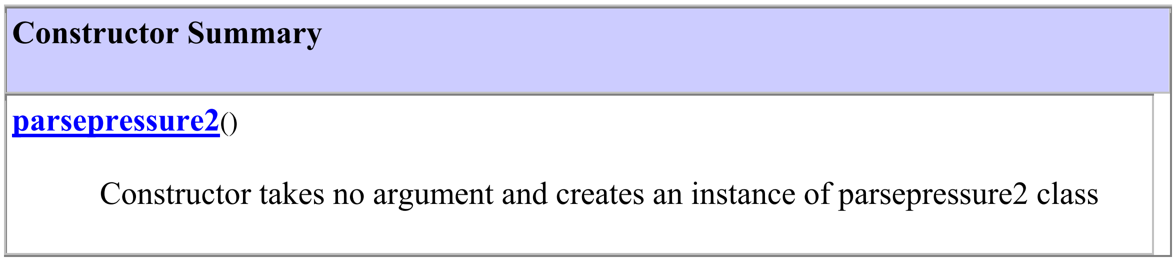
A.1 Phase Diagram - Pseudo-code

The parsepressure2 class (Figure A.1.1 and Figure A.1.2) reads the data files and samples the data points at regular intervals. The interval, type of sampling, number of samples etc, can be specified. The surface2 class (Figure A.1.3 and Figure A.1.4) creates an instance of the surfaceCanvas2 class (Figure A.1.5 and Figure A.1.6), which is the Java-OpenGL phase diagram applet). A thread is created to read the pressure, temperature values of the volume slice. It implements the Runnable interface. The main class-surfaceCanvas2 initializes all the graphic functions. It extends the GLAnimCanvas class and implements the MouseListener, MouseMotionListener, KeyListener, ActionListener interfaces.It parses the text files and constructs the NURBS surfaces. It performs interactive functions such as volume slice, *p-x, p-T, T-x* projections, light switch, log-scale, zoom, etc.

The MatrixFuncs class (Figure A.1.7 and Figure A.1.8) performs rotation of the phase diagrams depending upon mouse movements

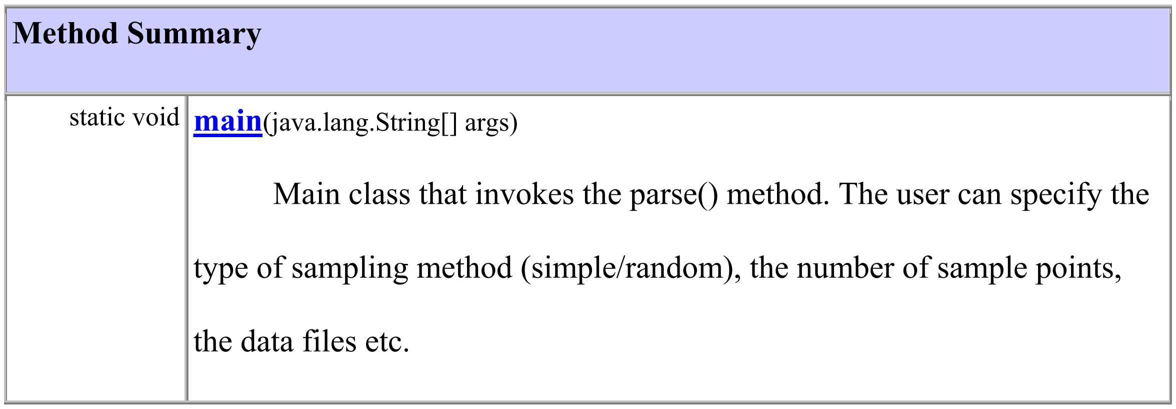
A.2 Molecular Simulation Module - Pseudo-code

The ChemicalPotential class (Figure A.2.1 – Figure A.2.3) is the main applet class that holds all the simulation elements. Using the new Controller class, (Figure A.2.3 – Figure A.2.6) the simulation starts off with the NVE integrator, runs for a maximum of 100 steps and transfers the phase to the Gear4NPH integrator. It displays the start/stop button on the applet. The MeterParticleSwap class (Figure A.2.7 and Figure A.2.8) is the meter class to measure the change in chemical potential via the Particle-Swap Method (species1 -> species2)



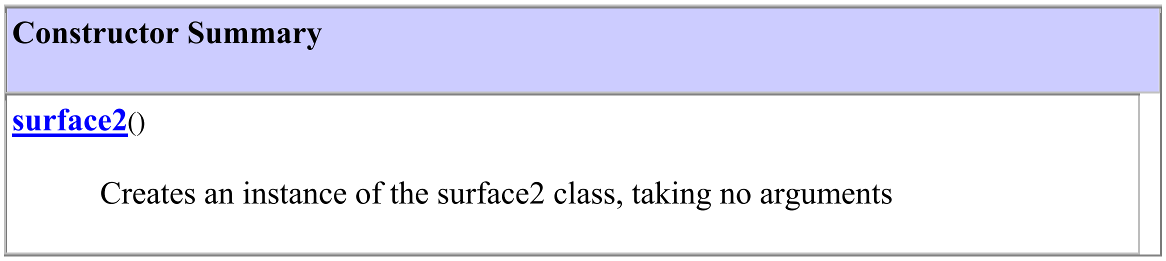
Constructor Summary
parsepressure2()   Constructor takes no argument and creates an instance of parsepressure2 class

Figure A.1.1: Class Parsepressure2 – Constructor Summary



Method Summary	
static void	**main**(java.lang.String[] args) Main class that invokes the parse() method. The user can specify the type of sampling method (simple/random), the number of sample points, the data files etc.

Figure A.1.2: Class Parsepressure2 – Method Summary



Constructor Summary
surface2()   Creates an instance of the surface2 class, taking no arguments

Figure A.1.3: Class Surface2 – Constructor Summary

Method Summary	
void	**destroy**() Destroy the applet.
void	**init**() Initialize the applet
void	**run**() The thread displays the Pressure, Temperature values after sleeping for every 1000ms .
void	**start**() Start the applet.
void	**stop**() Stop the applet.

Figure A.1.4: Class Surface2 – Method Summary

Constructor Summary
surfaceCanvas2(int w, int h) Constructor that takes width and height of the applet as its argument.

Figure A.1.5: Class SurfaceCanvas2 – Constructor Summary

Method Summary	
void	**actionPerformed**(java.awt.event.ActionEvent evt) Checks the type of event such as volume slice, log/normal scale, lighting, PT/PX/TX projection etc.
void	**display**() Displays the nurb surfaces(phase diagram), axes etc.
void	**doCleanup**() Removes the MousListener, keyListener etc
double	**getPressure**() Returns the Pressure value at the slice
double	**getTemp**() Returns the Temperature value at the slice
boolean	**getVol**() Returns the Volume value at the slice
void	**init**() Initializes fonts, menu options, knot values for the NURBS surfaces and adds the Listener objects etc.
void	**initt**() Reads the data points and sets the material properties for the surfaces

Figure A.1.6: Class SurfaceCanvas2 – Method Summary

void	**keyPressed**(java.awt.event.KeyEvent e) Invoked when a key has been released.
void	**keyReleased**(java.awt.event.KeyEvent evt) Invoked when a key has been released.
void	**keyTyped**(java.awt.event.KeyEvent e) Invoked when a key has been typed.
void	**mouseClicked**(java.awt.event.MouseEvent evt) Invoked when a mouse is clicked
void	**mouseDragged**(java.awt.event.MouseEvent evt) Invoked there is a mouse drag event.
void	**mouseEntered**(java.awt.event.MouseEvent evt) Invoked when a mouse enters the applet
void	**mouseExited**(java.awt.event.MouseEvent evt) Invoked when a mouse exits
void	**mouseMoved**(java.awt.event.MouseEvent evt) Invoked when a mouse is moved
void	**mousePressed**(java.awt.event.MouseEvent evt) Invoked when a mouse has been pressed

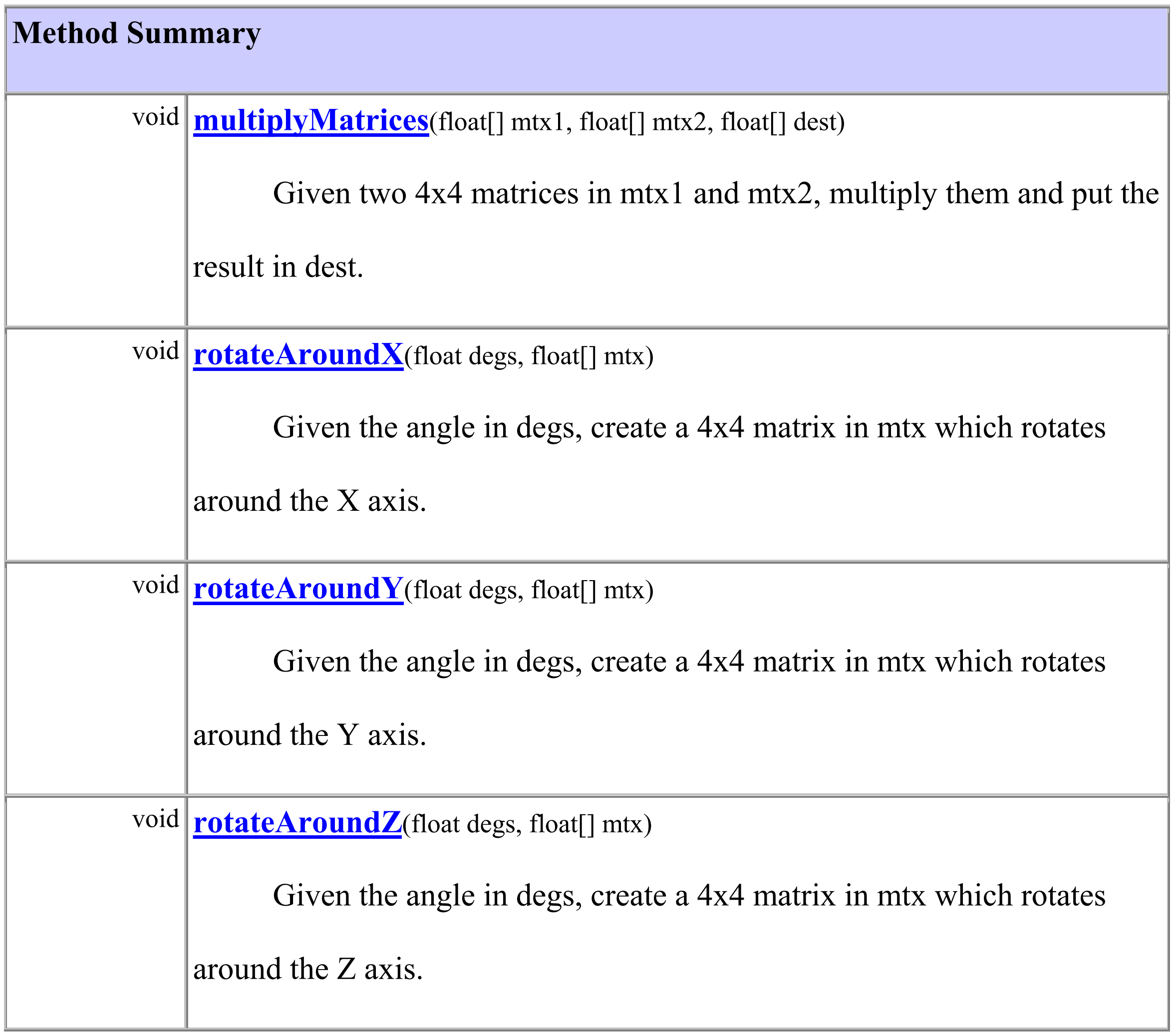
Figure A.1.6 Continued

void	**mouseReleased**(java.awt.event.MouseEvent evt) Invoked when a mouse has been released.
void	**preInit**() Sets doublebuffer to true.
void	**reshape**(int width, int height) Resizes the view.

Figure A.1.6 Continued

Constructor Summary
MatrixFuncs() Constructor that creates an instance of the MatrixFuncs class.

Figure A.1.7: Class MatrixFuncs – Constructor Summary



Method Summary	
void	**multiplyMatrices**(float[] mtx1, float[] mtx2, float[] dest) Given two 4x4 matrices in mtx1 and mtx2, multiply them and put the result in dest.
void	**rotateAroundX**(float degs, float[] mtx) Given the angle in degs, create a 4x4 matrix in mtx which rotates around the X axis.
void	**rotateAroundY**(float degs, float[] mtx) Given the angle in degs, create a 4x4 matrix in mtx which rotates around the Y axis.
void	**rotateAroundZ**(float degs, float[] mtx) Given the angle in degs, create a 4x4 matrix in mtx which rotates around the Z axis.

Figure A.1.8: Class MatrixFuncs – Method Summary

Constructor Summary	
Chemical Potential (int dim)	Constructor takes the space dimensions of the simulation. The constructor class creates instances of the Simulation, Phase, Space, Integrator, Controlle, Species, Potential, Display & all the Meter Classes.

Figure A.2.1: Class ChemicalPotential – Constructor Summary

Method Summary	
static void	**main**(java.lang.String[] args) main class that creates an instance of the ChemicalPotential Class

Figure A.2.2: Class ChemicalPotential – Method Summary

Inner Class Summary	
static class	**ChemicalPotential.ABEditor** Inner class that provides Text boxes for entering Sigma, Epsilon values in Real/LJ Units to specify the AB interactions
static class	**ChemicalPotential.Applet** Creates an instance of the applet and adds the ChemicalPotential Class to the panel
static class	**ChemicalPotential.MySpeciesEditor** Inner class that provides Text Boxes for entering Mass (Mol.Wt), Sigma and Epsilon (Real/LJ Units) to specify the AA/BB interactions
static class	**ChemicalPotential.SpeciesChooser** Inner class that defines a drop-down menu to select LJ parameters to mimic several real substances

Figure A.2.3 Class ChemicalPotential – Inner Class Summary

Inner Class Summary	
class	**ControllerJT.EnsembleToggler** Inner class used to toggle between NVE and NPH ensembles

Figure A.2.4: Class ControllerJT – Inner Class Summary

Constructor Summary
ControllerJT(chemical.Simulation sim, chemical.IntegratorGear4NPH nph, chemical.IntegratorGear4 nve) Constructor uses the Simulation, integratorgear4NPH and integratorGear4 as arguments

Figure A.2.5: Class ControllerJT – Constructor Summary

Method Summary	
boolean	**isDoNVE**() returns doNVE
void	**reset**() Resets the integrators
void	**run**() Thread call to perform 100 steps using NVE integrator and transferring control to the Gear4NPH integrator
void	**setDoNVE**(boolean b, boolean doStart) Transferring control from NVE integrator to the Gear4NPH integrator

Figure A.2.6: Class ControllerJT – Method Summary

Constructor Summary
MeterParticleSwap()   Constructor with no arguments
MeterParticleSwap(chemical.Simulation sim)   Constructor with Simulation class argument
MeterParticleSwap(chemical.Species s1, chemical.Species s2)   Constructor used to specify the test species of the molecule that would be swapped in

Figure A.2.7: Class MeterParticleSwap – Constructor Summary

Method Summary	
Double	**currentValue**() Performs Particle Swapping, doing nAvg attempts.
etomica.units.Dimension	**getDimension**() Etomica method
static chemical.EtomicaInfo	**getEtomicaInfo**() Etomica method
Void	**setactiv**(boolean temp) Sets the Meter Active.
Void	**setPhase**(chemical.Phase p) Sets the phase and gets handle to appropriate species agent
Void	**setSpecies**(chemical.Species s1, chemical.Species s2) Sets the species, takes a prototype molecule, and gets handle to appropriate species agent in phase

Figure A.2.8: Class MeterParticleSwap – Method Summary

A.3 Molecular Simulation API - *Etomica*

Etomica, the molecular simulation API written in Java, provides classes that represent species of molecules, the space that they occupy, the phase that they are in (gas, liquid, solid), and the potentials or molecular interactions between them. This API also has classes that perform MD or MC sampling, control the beginning and the end of the simulation, and meter the thermo-physical properties of the system. The API effectively provides a complete framework for developing and running an experiment on a molecular model of a chemical system [16].

The following lists all of the major components of the Molecular Simulation API along with their implementation and functionality.

- **Simulation**

  The Simulation class (Figure A.3.1) of the Molecular Simulation API is the main class that organizes the elements of a molecular simulation. It holds a handle to every component object in the simulation and also contains methods to register and deregister components. It also holds a Space object that is referenced by all the simulation components whenever spatial values are needed, a Graphics object that can be used to display the simulation in a GUI, as well as an object that specifies the unit system used as a default for all I/O to displays and meters. Finally, the simulation instance contains the element coordinator object, which is responsible for linking all the components in the simulation together just before runtime [16].

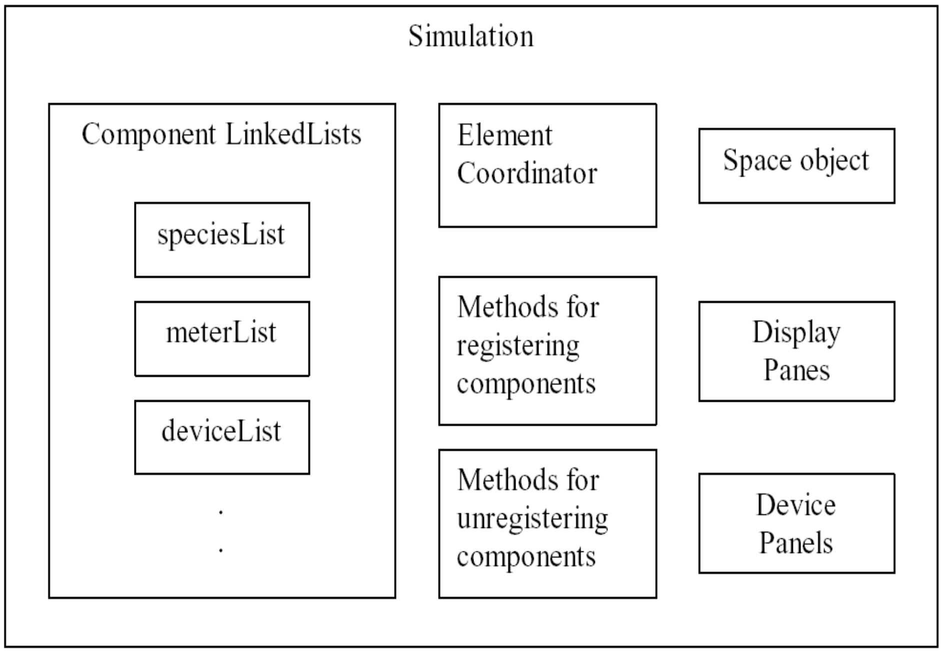


Figure A.3.1: ***Etomica*** **- Simulation Class. From [16].**

- **Phase**

  The Phase class (Figure A.3.2) holds the atoms that interact with each other. It also holds a configuration object that lays out the arrangement of molecules. The boundary object defines the interaction of the molecules at the edges of the phase. The Iterator Factory object helps in looping through the molecules. The Meter object measures the physical properties of a given Phase. The Species Agents objects aids in adding, counting and deleting molecules in a Phase [16].

- **Space**

  Space provides some general physical framework of a simulation by providing the ability to create vectors, tensors, boundaries (periodic/non-periodic boundary conditions), and coordinates (position and momentum of a point in space) [16].

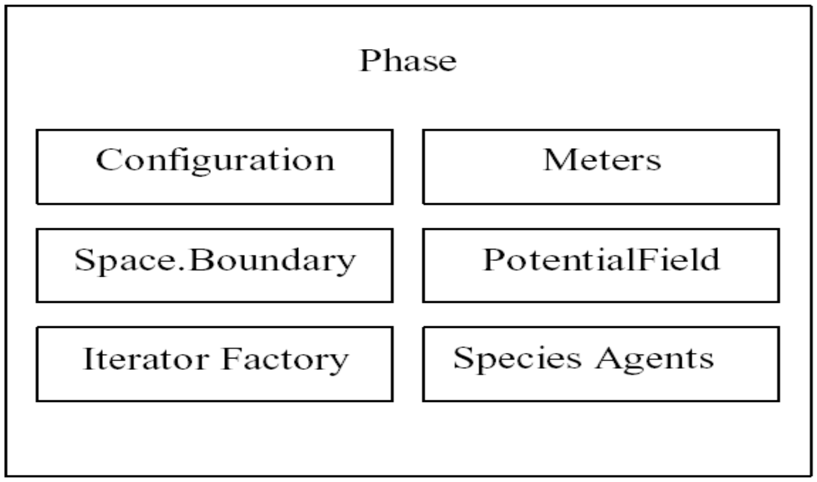


Figure A.3.2: ***Etomica*** **– Phase Class. From [16].**

- **Integrator**

  The Integrator class generates the new configurations of the phase by defining the movement of atoms/molecules. For example, moves can consist of adding or deleting molecules, translating the position of a molecule, or moving molecules based on their present velocity and according to the equations of motion. Etomica provides both the types of integrators (molecular dynamics and Monte Carlo simulations) [16].

- **Controller**

  Controller controls all actions performed by the integrators in a simulation. It is capable of starting and stopping any and all integrators whenever necessary. A simple representation of a controller is a Start/Stop button for a simulation [16].

- **Species**

  A Species is a collection of identically formed Molecules. Molecules can consist of one or more Atoms. Species holds the AtomType of the atoms in the Species'

Molecules, as well as the configuration of the atoms in each of these molecules [16].

- **Potential**

  The Potential class (Figure A.3.3) defines the interactions between Atoms. A Potential can have the property of being hard, soft, or both. Hard potentials describe impulsive interactions between Atoms. Energy curves of this type of interaction have discontinuities. Atoms engaging in this type of interaction have definite collision times, which occur when the collisionDiameter of the two Atoms is equal to their distance apart from each other.

  Soft potentials are non-impulsive interactions between Atoms. Energy curves of this type of interaction are smooth with no discontinuities. Atoms engaging in this type of interaction do not have a definite collision time, but rather are constantly subjected to the force caused by the potential. These potentials provide a force method to acquire the effect of this force [16].

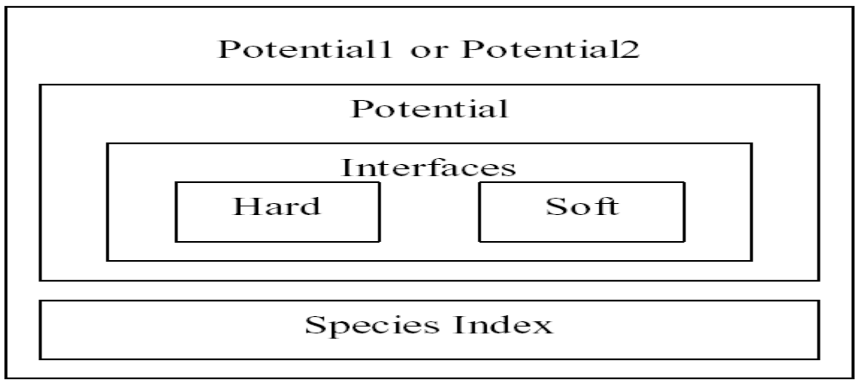


Figure A.3.3: ***Etomica*** **– Potential Class. From [16].**

- **Display**

  Using the Display class, one can view the results of the simulation in the form of tables, plots of data curves, molecular movements etc. For example, DisplayPhase shows the molecular movements of atoms/molecules within a Phase [16].

- **Modulator**

  Modulator provides an interface to make changes to the properties of another object. For example, using a modulator, a Device such as a DeviceSlider, can be linked to the temperature of an Integrator. The Modulator will simply read the current value of the DeviceSlider and modify the temperature of the Integrator accordingly [16].

- **Device**

  Devices such as DeviceSlider, DeviceButton etc are objects that provide access to and methods for changing or modifying fields of other objects. Devices provide graphical interfaces for modifying these fields, as well as a Unit object for determining the corresponding Units [16].

- **Meter**

  A Meter is responsible for measuring and averaging properties during a simulation. Meters measure various physical properties of a Phase. However, they are not capable of displaying the results. Instead, Meters depend on Displays, such as the DisplayBox or DisplayTable, to display the measured results [16].

VITA

Sharad Gupta was born in 1977 and is from India. He attended elementary and high school at Bombay. He received his B.S degree in Computer Engineering in fall 2000, from the University of Bombay. He worked as a Software Engineer in a software consultancy firm - Infosys Technologies Ltd. He began his graduate study at University of Tennessee from Spring 2001 and worked as a Graduate Research Assistant in the Dept of Computer Science.

After completing his MS degree in Computer Science, he decided to pursue an MBA degree at the University of Tennessee. He plans to work as a technology consultant after graduation.