A THEORETICAL STUDY OF CRYSTAL GROWTH IN NANOPOROUS MATERIALS USING THE MONTE CARLO METHOD

A thesis submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2013

James Gebbie

School of Chemistry
# Contents

List of Abbreviations  
List of Symbols  
Abstract  
Declaration  
Copyright Statement  
Acknowledgements  

1 Introduction to Crystal Growth and Nanoporous Materials  
1.1 Justification and Structure of Thesis  
1.2 Crystal Growth Theory  
1.2.1 Phases of Matter  
1.2.2 Supersaturation  
1.2.3 Supersaturation in Solution  
1.2.4 Nucleation  
1.2.4.1 Primary Nucleation  
1.2.4.2 Secondary Nucleation  
1.2.5 Induction Time  
1.2.6 Crystal Growth  
1.2.7 The Growth Surface  
1.2.7.1 Models to Simplify Crystal Growth  
1.2.7.2 Growth on a Smooth Surface  
1.3 Crystallography  
1.3.1 Crystal Systems  
1.3.2 Miller Indices  
1.3.3 Miller-Bravais Indices  
1.4 Nanoporous Materials  
1.4.1 Zeolites  
1.4.1.1 The Properties and Uses of Zeolites  
1.4.1.2 Synthesising Zeolites
1.4.1.3 The Crystal Growth of Zeolites
1.4.2 Zeotypes
1.4.3 Metal Organic Frameworks (MOFs)
1.5 References

2 Computational Methodology

2.1 Critical Literature Review
2.1.1 Gale – Monte Carlo Study on Urea
2.1.2 Boerrigter – MONTY
2.1.3 Modelling Crystal Growth at the Centre for Nanoporous Materials (CNM)
2.2 Introduction to the Programs and Functionality
2.2.1 The Simulation Program
2.2.1.1 Basic Operation
2.2.1.2 Connectivity
2.2.1.3 Energetics
2.2.1.4 Driving Force
2.2.1.5 General Algorithm
2.2.1.6 Performance Enhancements
2.2.2 The Visualiser Program
2.2.2.1 Features
2.2.2.2 Performance
2.2.2.3 Summary of key bindings
2.3 References

3 Results and Discussion (Zeolite L, Offretite & Erionite)

3.1 Aims
3.2 Zeolite L
3.2.1 Introduction to Zeolite L
3.2.2 Problem Breakdown
4.3.1.2 Energetics 169
4.3.1.3 Results and Discussion 171
4.3.2 Model Approach 2 178
  4.3.2.1 Connectivity 178
  4.3.2.2 Energetics 179
  4.3.2.3 Results and Discussion 181
4.3.3 Model Approach 3 193
  4.3.3.1 Connectivity 193
  4.3.3.2 Energetics 194
  4.3.3.3 Results and Discussion 196
4.4 Silica Sodalite 205
4.5 References 208

5 Results and Discussion (Zeolite A) 210
  5.1 Aims 211
  5.2 Introduction to Zeolite A 211
  5.3 Problem Breakdown 214
    5.3.1 Connectivity 214
    5.3.2 Energetics 215
  5.4 Results and Discussion 217
    5.4.1 Crystal Growth of Zeolite A 217
    5.4.2 Nucleation 240
  5.5 References 247

6 Conclusions and Further Work 249
  6.1 Conclusions 250
  6.2 Future Work 255
    6.2.1 Further Work (scientific) 255
      6.2.1.1 Improving the Generality of the Algorithm 255
      6.2.1.2 Introducing Defects 256
      6.2.1.3 Multiphase Systems 257
      6.2.1.4 Nucleation 257
    6.2.2 Further Work (computational) 257

6
6.2.2.1 Large Simulations 257
6.2.2.2 Facet Fractions 258
6.2.2.3 Intelligent Parameter Exploration 260
6.2.2.4 Improvements to Usability 260

6.3 References 261

Appendices 262
Appendix 1 263
Appendix 2 282
Appendix 3 295
Appendix 4 298
Appendix 5 299
Appendix 6 394

Word count: 74,996 words
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two-Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-Dimensional</td>
</tr>
<tr>
<td>4R, D4R</td>
<td>Four-Ring, Double Four-Ring</td>
</tr>
<tr>
<td>5R, D5R</td>
<td>Five-Ring, Double Five-Ring</td>
</tr>
<tr>
<td>6R, D6R</td>
<td>Six-Ring, Double Six-Ring</td>
</tr>
<tr>
<td>AFM</td>
<td>Atomic Force Microscopy</td>
</tr>
<tr>
<td>BCF</td>
<td>Burton, Cabrera and Frank (theory)</td>
</tr>
<tr>
<td>CNM</td>
<td>Centre for Nanoporous Materials (University of Manchester)</td>
</tr>
<tr>
<td>ERI</td>
<td>Erionite</td>
</tr>
<tr>
<td>FSC</td>
<td>First Sphere Coordination</td>
</tr>
<tr>
<td>KDP</td>
<td>Potassium DiHydrogen Phosphate</td>
</tr>
<tr>
<td>KMC</td>
<td>Kinetic Monte Carlo</td>
</tr>
<tr>
<td>LTA</td>
<td>Linde Type A (zeolite A)</td>
</tr>
<tr>
<td>LTL</td>
<td>Linde Type L (zeolite L)</td>
</tr>
<tr>
<td>MC</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>MD</td>
<td>Molecular Dynamics</td>
</tr>
<tr>
<td>MOF</td>
<td>Metal Organic Framework</td>
</tr>
<tr>
<td>OFF</td>
<td>Offretite</td>
</tr>
<tr>
<td>PBC</td>
<td>Periodic Bond Chain (theory)</td>
</tr>
<tr>
<td>PBU</td>
<td>Primary Building Unit</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SBU</td>
<td>Secondary Building Unit</td>
</tr>
<tr>
<td>SEM</td>
<td>Scanning Electron Microscopy</td>
</tr>
<tr>
<td>SOD</td>
<td>Sodalite</td>
</tr>
<tr>
<td>ZIF</td>
<td>Zeolitic Imidazolate Framework</td>
</tr>
</tbody>
</table>
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Concentration</td>
</tr>
<tr>
<td>G</td>
<td>Gibbs Free Energy</td>
</tr>
<tr>
<td>H</td>
<td>Enthalpy</td>
</tr>
<tr>
<td>k</td>
<td>Boltzmann Constant</td>
</tr>
<tr>
<td>P</td>
<td>Pressure</td>
</tr>
<tr>
<td>$P_D$</td>
<td>Probability of Growth</td>
</tr>
<tr>
<td>$P_G$</td>
<td>Probability of Dissolution</td>
</tr>
<tr>
<td>r</td>
<td>Radius</td>
</tr>
<tr>
<td>S</td>
<td>Entropy</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
</tr>
<tr>
<td>T</td>
<td>Temperature</td>
</tr>
<tr>
<td>U</td>
<td>Internal Energy</td>
</tr>
<tr>
<td>V</td>
<td>Volume</td>
</tr>
<tr>
<td>$v_c$</td>
<td>Molecular Volume</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Surface Energy</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Driving Force (supersaturation)</td>
</tr>
</tbody>
</table>
Abstract

This work is aimed at understanding the underlying processes of crystal growth in nanoporous materials at the molecular level utilising computational modelling. The coarse grain Monte Carlo program constructed over a number of works at the CNM has shown success in modelling cubic zeolite systems. The goal of this work is to adapt the program to deal with the complexities of a wide range of different crystal systems.

There have been many studies of crystal growth and many problems solved however in zeolites there are still a lot of questions to answer. Growth rates and activation energies for crystal growth processes in zeolites are some of the things that remain unsolved for zeolites. Coarse grain Monte Carlo modelling simplifies the problem and can provide an insight into the underlying processes that govern crystal growth.

This study focused its energetics around the formation of stable closed cage surface structures deduced from careful study of the dissolution of zeolite L terraces. Two approaches from an energetic point of view were investigated during the course of this study, the first considered the energetics from an energy of attachment point of view whilst the second focused on the energy of destabilisation with respect to crystal bulk.

In this study the crystal growth of the following systems were probed computationally LTA, SOD, LTL, ERI, OFF. Both zeolite and MOF crystal systems were studied over the course of this work.

The algorithm developed in study shows some potential in being able to give insight to experimental crystal growth chemists as to how changing the rates of growth of certain cage structures would affect the overall morphology of the crystal grown. They can then utilise their knowledge of how using certain cations or templates for example can alter the stabilisation of certain cage structures to in effect design crystals of desired properties.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application or another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

I would like to thank Professor Michael Anderson (my supervisor) for his constant support and guidance, for pushing me and encouraging me throughout my PhD, and in particular in giving oral presentations. I would also like to thank Mike not only for his professional support, but also in becoming a good friend over the years.

I would also like to thank my partner Amrita Rayet, who has given me constant support and encouragement when things got tough! She showed me that with enough determination and energy that anything is achievable, she was my inspiration, my rock and my best friend. I love you so much!!

In addition I would like to thank all of the members of the Centre for Nanoporous Materials who provided not only vital scientific discussion but also friendship, in particular I would like to thank Dr Mark Holden, Dr Pablo Cubillas, Dr Amy Holmes, Dr Alex Bushell, Dr Lindsay Lai, Dr Qin Ling, Dr Chompoonoot Nanthamathee and Rachel Smith. A very special thank you to my dear friend Andrew Eberhart, for all those years of sharing lunch and all those amazing times in the pub, thanks man! You are awesome.

Also I would like to thank Mark Whitby (Research Computing Services), Mike Croucher (Research Computing Services), Ian Cottam (Research Computing Services) and Zaiem Bakker (IT Services) for their support with various supercomputing architectures and software.

Finally I would like to thank my family for always being supportive and providing me with a nice quiet place to relax, my parents for always encouraging me and totally not understanding a word I said about my work.
Chapter 1

Introduction to Crystal Growth and Nanoporous Materials
1.1 Justification and Structure of Thesis

The underlying processes in which atoms and molecules form regular ordered crystalline solids has been of great interest to scientists of all scientific fields for over 100 years. The motivation for modern day research comes from the desire to produce materials that have been designed for a particular purpose. This rational design requires detailed understanding of the mechanisms by which the molecules become arranged during growth of a crystalline phase from solution. With this kind of understanding of the fundamental processes of crystal growth, it would be possible to gain control over some of the properties of the resulting material such as crystal size, morphology and defects.

Nanoporous materials are one such class of compounds that are of great industrial interest due to their useful properties. Zeolites, zeotypes and metal-organic frameworks are all members of the nanoporous class of materials. These materials consist of highly ordered frameworks and most range in porosity and in varying numbers of porous dimensions. Their porous nature creates huge surface areas and voids within the crystal where a whole host of chemistry can take place. Since pore properties such as length and surface exposure can vary with the crystals size and morphology, these properties can be an important factor in determining the overall material properties.

The focus of this study is to further develop our current understanding of the crystal growth of nanoporous materials from a theoretical approach. The coarse grain Monte Carlo approach has been promising in explaining some of the topographical and morphological habits in zeolite A. The main goal of this study is to further develop an algorithm that can be applied to different crystal frameworks and also extend to the possibility of different crystal types such as MOF’s.

The structure of this thesis follows the following format; chapter 1 provides an insight into the literature concerning crystal growth and the science surrounding nanoporous materials. Chapter 2 will give a brief overview of the history of this line of modelling at the centre for nanoporous materials and some key literature behind the project. It will then move on to discuss the basic operation, functionality and features of the programs written during this work. Chapter 3 contains the results and discussion as well as brief introductions to zeolite L, Offretite and Erionite systems.
Chapter 4 is focused on the development of this model to its current state, the sodalite system was chosen since it is relatively simple to study experimentally providing a quality test case for development. The chapter briefly introduces the sodalite framework and the ZIF-8 material followed by an introduction to each model approach then the results and discussion. Chapter 5 is concerned with the study surrounding zeolite A, a brief introduction to the zeolite followed by an introduction to the model is presented. This is then followed by the results and discussion of both the crystal growth study and the nucleation study. Chapter 6 is the final chapter containing an ensemble of the conclusions from each study along with the future work arising from this thesis. The appendix at the rear of this thesis contains the source code for a small selection of the programs coded throughout this PhD.

1.2 Crystal Growth Theory

The natural beauty of crystalline materials has fascinated mankind for centuries. Crystals occurring in nature form in a variety of locations and conditions, ranging from the intense inferno of volcanoes to the cool mineral rich waters of the oceans. Crystals are formed by the process of crystallisation from a medium such as a melt, solution, gel, vapour or plasma.

![Figure 1-1 An image of giant natural Gypsum crystals in the caves of Naica, Chihuahua, Mexico. The crystals measure up to 11 metres in length and were discovered in 2000. Copyright Javier Trueba / Madrid Scientific Films.](image)

Crystallisation is behind many technological, laboratory procedures and pharmaceutical processes. Large single crystals grown from solution have been
exploited in high energy optics such as the high energy laser facility at Lawrence Livermore National Laboratory (Figure 1-2), where optical purity is essential at such high beam energies.\textsuperscript{1} The Czochralski process is a method to obtain large single crystals of semiconducting materials for the production of electrical components.\textsuperscript{2}

In chemical production crystallisation is used to isolate products of large scale reactions on the multi ton scale, in the pharmaceuticals industry drugs are produced in crystalline form to ensure slow and sustained rates of activity inside the patient’s body.\textsuperscript{3}

A molecular level understanding of the crystal growth process provides the ability to quantitatively describe crystallisation and ultimately take control to produce the desired outcome in terms of size, morphology and quality.

1.2.1 Phases of Matter

A phase of a pure substance is a distinct form of matter that occurs when the physical properties and state are uniform throughout the system. Melting, vaporisation and conversion between polymorphs (allotropes if the system is elemental) are all examples of a change in phase whilst keeping chemical composition constant. A phase transition is the spontaneous conversion of one phase to another and occurs at a certain temperature for a given pressure. The regions of temperature and pressure at which a given phase is stable are indicated on a phase diagram (Figure 1-3).\textsuperscript{4}
Figure 1.3 A typical phase diagram of a single component system showing the phase boundaries between the solid, liquid and vapour phases.

The lines that separate each phase are known as phase boundaries, they show the pressures and temperatures at which two phases coexist at equilibrium. The crystal phase is thermodynamically stable at low temperatures and high pressure, the liquid phase is thermodynamically stable at high temperatures and high pressures whilst the vapour phase is stable at high temperature and low pressure.

There are two special points on the phase diagram to note, the triple point indicated at O and the critical point indicated by O’. At the critical point the liquid-vapour boundary terminates indicating that the density of the vapour phase is equal to that of the liquid phase. At and above this critical point the surface between the two phases disappears producing a uniform phase termed a supercritical fluid.

The other special point, the triple point indicates the conditions in which the three different phases (crystal, liquid and vapour) can coexist in equilibrium. If one was to move along the lines A-A’ or AA” the vapour phase would become supersaturated with respect to the crystalline phase. Moving along the line B-B” in the negative temperature direction would cause the liquid phase to become supersaturated with respect to the crystalline phase.
1.2.2 Supersaturation

As mentioned earlier the lines that divide each phase on the phase diagram in Figure 1-3 represent the points at which the two phases are in equilibrium, this means that the chemical potentials of the two phases are also in equilibrium. If a change in temperature or pressure occurs which leads to deviation from one of these lines, then one of the phases becomes more stable with respect to the other. This increased stability means that the chemical potential becomes smaller than the potentials of the other phase(s). A change that results in such a change in stability triggers a transition from one phase to the other. The difference between the chemical potentials $\Delta \mu$ of the two phases is called the thermodynamic driving force or supersaturation and can be defined as\(^5\)

$$\Delta \mu = kT \ln \frac{P}{P_0}$$

Equation 1-1

If we consider the case of moving along the A-A’ as mentioned in the previous section, the chemical potentials of the vapour phase $\mu_v$ and the crystal phase $\mu_c$ can be plotted against pressure Figure 1-4. At pressures lower than $P_0$, $\Delta \mu$ is negative therefore the crystalline phase will sublime. Whilst if the pressure is greater than $P_0$ then $\Delta \mu$ is positive and the vapour phase will crystallise.\(^5\)

Figure 1-4 Chemical potential vs pressure at constant temperature for the vapour $\mu_v$ and crystal $\mu_c$ phases when moving along the line A-A’ as in figure 1-3. The pressure $P_0$ represents the equilibrium pressure.\(^5\)
Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials

The case when moving along the line B-B” in Figure 1-3 shows the change in phase at constant pressure. The chemical potentials of the liquid $\mu_l$ and crystal $\mu_c$ phases can be plotted against temperature Figure 1-5. At temperatures lower than $T_m$ the driving force $\Delta \mu$ is positive therefore the liquid phase will solidify into a crystalline phase, whilst at temperatures greater than $T_m$ the crystalline phase will dissolve into the liquid phase.(negative $\Delta \mu$).²

![Chemical potential vs temperature at constant pressure for the liquid $\mu_l$ and crystal $\mu_c$ phases when moving along the line B-B”. $T_m$ represents the melting point.²](image)

1.2.3 Supersaturation in Solution

It’s been discussed how phase transfers from the vapour and liquid phase in given conditions produce uniform crystalline phases. In solution chemistry, a solution is supersaturated if it contains more of a dissolved material than can ordinarily be dissolved in a given amount of the specific solvent, it is defined as:²

$$\Delta \mu = kT \ln \frac{C}{C_0}$$  \hspace{1cm} \text{Equation 1-2}$$

In 1897 Wilhelm Ostwald was the first to introduce the concept that supersaturated solutions occur with different stabilities.⁶

A solution can be thought to consist of three different phases, the stable phase occurs in undersaturated solutions where dissolution of solid material dominates. The metastable phase occurs at low supersaturation, spontaneous nucleation is unfavourable
whilst epitaxial growth may still occur on seeds. The labile phase occurs at relatively high supersaturation and contains the nucleation and amorphous crystal growth pathways, the different zone are summarised in Figure 1-6.

1.2.4 Nucleation

The creation of a new phase requires the occurrence of small clusters of building units (atoms or molecules) within the supersaturated phase, this was initially pointed out by Gibbs. The development of theory for formation of such clusters by Gibbs, Volmer, Farkas, Stranski and Kaishew, Becker and Döring, Frenkel and others, this became known as classical nucleation theory. Nucleation is the first stage of crystal growth, and consists of two main categories. These are primary and secondary nucleation; both processes will be explained in detail in this section.

1.2.4.1 Primary Nucleation

Primary nucleation refers to nucleation in a system that contains no previously formed crystalline material. There are two main mechanisms in which this can take place, homogenous and heterogeneous nucleation.
Homogenous Nucleation

Homogenous nucleation occurs by the spontaneous arrangement of molecules to form small clusters. Clusters that form must reach a critical size in order to continue growth into a large crystal.\(^\text{15}\)

Energetics

The free energy of formation of a nucleus (droplet) can be described in terms of the free energy contribution of the bulk crystal and that of the surface free energy. The Gibbs equation for surface energy of a flat surface is:

\[
\Delta G_s = 4\pi r^2 \sigma
\]

Equation 1-3

Where \(r\) is the radius of the nucleus and \(\sigma\) is the surface tension at the interface with solution. If one is to treat the cluster as a small sphere then the free energy of formation can be expressed as follows:\(^5\)

\[
\Delta G_i = -\frac{4\pi r^3}{3v_c} (\Delta \mu) + 4\pi r^2 \sigma
\]

Equation 1-4

Where \(r\) is the radius of the cluster and \(v_c\) is the molecular volume of the cluster. Figure 1-7 shows a plot of \(\Delta G_i\) against \(r\) with a maximum at \(r^*\) which corresponds to the radius of the critical nucleus.

Figure 1-7 The Gibbs free energy as a function of cluster radius when envisaged as a sphere. The value \(r^*\) represents the radius of a critically sized nuclei.\(^5\)
What this plot is essentially showing is how the free energy $\Delta G_i$ of the formation of a cluster varies with cluster time (purple curve). The light blue curve illustrates the positive free energy contribution as a result of the surface free energy, hence the dependence on $r^2$. The dark red curve illustrates the negative contribution towards the total free energy as contributed by the volume excess term, hence the dependency upon $r^3$. The point at which these terms start to balance forms the critical cluster radius $r^*$. At lower $r < r^*$ the surface free energy dominates the free energy and thus nuclei are energetically unstable. However as $r > r^*$ the volume excess free energy quickly dominates the free energy expression and thus the nuclei is stabilised by its bulk and will proceed on to growth. The critical radius $r^*$ can be expressed as:

$$ r^* = \frac{2\sigma v_c}{\Delta \mu} $$  

Equation 1-5

Differentiating equation 1-4 with respect to $r$ and equating to zero gives:

$$ \frac{d\Delta G_i}{dr} = -\frac{4\pi r^2}{v_c} \Delta \mu + 8\pi r \sigma = 0 $$  

Equation 1-6

Substituting Equation 1-5 into Equation 1-4 gives:

$$ \Delta G_{critical} = \frac{16\pi \sigma^3 v_c^2}{3\Delta \mu^2} $$  

Equation 1-7

Where $\Delta G_{critical}$ is the Gibbs free energy of formation of a critically sized nucleus as a function of supersaturation.$^5$

**Nucleation Rate**

As mentioned earlier it is difficult for crystals to nucleate within the meta-stable zone, however once a critical supersaturation has been reached the rate of nucleation increases rapidly as a function of supersaturation (nucleation zone) Figure 1-6.$^5$ The critical nucleation rate can be expressed as:

$$ J = \Omega \exp \left[ -\frac{\Delta G_{critical}}{kT} \right] $$  

Equation 1-8

Where $\Omega$ consists of pre exponential factors relating to adsorption and geometry of the nucleus.$^5$ It was demonstrated in Equation 1-6 that $\Delta G_{critical}$ can be shown as a
function of supersaturation, substituting Equation 1-6 into Equation 1-8 and plotting gives Figure 1-8.\(^5\)

![Figure 1-8 A plot showing the nucleation rate J against supersaturation Δμ where Δμ_{critical} shows the critical supersaturation where nucleation begins to increase rapidly, hence the end of the meta-stable region.\(^5\)](image)

**Heterogeneous Nucleation**

The second case is heterogeneous nucleation, it occurs when foreign substrates are present that aid nucleation, however not all substrates act to aid nucleation some can inhibit the process. Heterogeneous nucleation can occur at much lower supersaturation than homogenous nucleation.\(^15\) This is due to the fact that when considering the Equation 1-4 and Equation 1-5, both the ΔG\(_i\) and r\(^*\) depend upon the surface free energy γ. It is known that in the presence of a foreign substrate a decrease in γ also reduces ΔG\(_i\) and r\(^*\) at constant supersaturation Δμ.\(^16\) What is more the reduction in γ will also decrease the critical supersaturation for nucleation. This reasoning means that heterogeneous nucleation is more favourable than that of homogenous nucleation, with the lowest γ when a close match between the foreign substrate and the crystallising material is found.\(^16\)
1.2.4.2 Secondary Nucleation

Secondary nucleation is growth of new crystalline material by introducing a seed crystal to the supersaturated solution. The seed crystals act as a substrate on which secondary nucleation and growth can occur. For the same reasons as primary heterogeneous nucleation, secondary nucleation is both more favourable and occurs at a lower supersaturation than both homogeneous and heterogeneous primary nucleation. This is because the substrate being used is identical (closest theoretical) to the material that is being crystallised.\(^{16}\)

1.2.5 Induction Time

The induction time refers to the elapsed time between achieving a supersaturated growth solution and the observation of the first nucleated crystals. The time reported can vary widely and is dependent on many parameters, such as the successful determination of supersaturation, the ability to detect first nuclei etc.\(^{15}\)

1.2.6 Crystal Growth

Crystal growth is the second part of the crystallisation process. The nuclei formed in the first part of crystal growth (nucleation) continue to grow into larger crystals by incorporating growth species into the crystal surface. The processes that lead to incorporation of new material can be thought to consist mainly of the following steps:

1. Transport of growth units in solution.
2. Adsorption of growth units onto the surface.
3. Movement of growth units around the surface.
4. Attachment of growth units to surface site.

These four processes can be classified into two distinct categories. The first process belongs to the transport process category, whilst 2-4 belong to the surface process category. These events usually proceed in a sequential manner, as a result the slowest of the four processes will govern the overall crystal growth. With this in mind it can be seen that crystal growth can be controlled by the transport properties if step 1 is slowest, or surface controlled if any of the other 3 processes are slowest.\(^{17,18}\)
1.2.7 The Growth Surface

Crystal growth is an extremely complicated topic to both study and quantify. A number of models and thus from those models, mechanisms have been developed to explain the way crystals grow. Some of these models will be introduced here followed by some of the surface growth mechanisms commonly observed.

1.2.7.1 Models to Simplify Crystal Growth

There are numerous models that aim to explain the complexities of the growth of crystals. A small selection which constitutes the most commonly referred to will be introduced in this section.

The Kossel, Stransi, Volmer (KSV) Theory

The Kossel crystal is composed of simple cubic building units, each building unit serves as an approximation of the chemical composition represented by each unit. The number of connections to other units that each unit has represents the approximate number of chemical bonds needed to add the unit in its particular position. Kossel\textsuperscript{19}, Stransi\textsuperscript{19} and Volmer\textsuperscript{10} recognised that discontinuities in a crystal surface were important in incorporating new material onto the crystal surface. Figure 1-9 shows the different attachment types.

\begin{center}
\includegraphics[width=\textwidth]{kossel_model.png}
\end{center}

\textit{Figure 1-9 A schematic of the Kossel model (a) represent a 6 coordinate site or crystal bulk (b) shows the different types of surface site type}

Kossel\textsuperscript{20} and Stransi\textsuperscript{19} introduced the idea that growth via kink sites was more favourable than those of surface or edge sites. In Figure 1-10 Stransi showed that it
is possible to differentiate between sites on \{100\}, \{110\} and \{111\} by considering the connectivity of neighbouring sites.\(^{19}\)

\[ \text{Figure 1-10 Stranski applied the Kossel model to the } \{100\}, \{110\} \text{ and } \{111\} \text{ of the NaCl lattice.}\(^{19}\) \]

**The Burton, Cabrera, Frank (BCF) Theory**

Early nucleation theories suffered from shortfalls, one of which was once the growth front of kinks had reached the edge of the crystal it could no longer provide low energy nucleation sites. The creation of a new layer would require 2D nucleation, a process which the calculated supersaturation is approximately 50%.

Frenkel\(^{21}\) showed that a perfect crystal above the temperature absolute zero would have a certain amount of thermal roughening of the surface, which he showed to contain a high number of kinks. Burton Cabrera\(^{22}\) showed the number of kinks was much higher than proposed by Frenkel, an important point in understanding the rate of advance of steps. As a consequence of this theory, they showed that it would take extremely high supersaturation to create a new layer of steps on a perfect crystal surface.\(^{23, 24}\) This lead to the conclusion that the growth of crystals under low supersaturation can only be explained by realising that the growing crystals cannot be perfect.

Burton Cabrera and Frank introduced that the formation of points of dislocations would act as a continual source of steps. Frank\(^{25, 26}\) suggested that the dislocations would lead to spiral growth Figure 1-11.
In spiral growth nucleation of a new layer is not necessary so the crystal can grow at low supersaturation, as what tends to be observed experimentally.

**Periodic Bond Chain (PBC) Theory**

Hartmann showed for perfect crystals the morphology could be roughly predicted by deducing the slowest growing face. Burton had already suggested the growth rate of a given face in the direction normal to its surface was proportional to the kink density on that face. Hartman classified crystal faces into three different groups, F (flat), S (step) and K (Kink) based on how many dense rows of atoms they are parallel to. If the surface is parallel to at least two dense rows of atoms then it is of type F, if the surface us parallel to one row then it is of type S and not parallel to any then it is of type K. Typical examples of each type would be the Kossel faces (100) as type F, (110) as type S and (111) as type K.\(^5\)
K faces have the greatest density of kink sites than the other two faces and as a consequence will grow the fastest. S faces also form kink sites but the density is not as high as for K faces. In perfect crystals, S faces shouldn’t form kink sites at all, it follows that under low enough supersaturation to suppress nucleation the K face should disappear first followed by the S faces. The result should be a crystal consisting of just F faces and cease to grow any further due to nucleation being suppressed.5

1.2.7.2 Growth on a Smooth Surface

The models introduced previously can be used to understand the growth of crystals. For crystal growth to occur on a smooth surface that is defect free, a layer by layer mechanism is required. This mechanism goes under a variety of names in scientific literature such as birth and spread, layer by layer growth or the TLK mechanism.

Figure 1-13 shows the birth and spread mechanism, for layers that are incomplete. Growth units can move around the surface with preference to attach into kink sites (a). The more material that incorporates into edge and kink sites on existing layers the more these layers advance (b). When layers are complete, nutrients can either undergo desorption back into the growth solution or go onto form a 2D nuclei thus forming a new layer (c).5
Figure 1-13 showing birth and spread growth a) a growth unit is moving along the crystal surface and adding preferentially to kink site b) the layer has advanced due to incorporation of growth units at kink sites c) a 2D nucleation event is required to grow a new layer.

The rate at which 2D nucleation can occur is highly supersaturation dependant. If the supersaturation is high enough then several layers may nucleate one on top of the other, even before other layers can complete.

If growth is occurring on a surface that is not defect free then it is likely that it will include screw dislocation defects. If screw dislocations are present then the spiral growth mechanism will occur as has been described in the previous section under models. The screw dislocation is an endless source of new layers, due to the fact as it screws around itself the defect is propagated up through new layers. Due to this fact, crystal growth will still occur at supersaturation levels where nucleation of new layers is not possible. For this reason, often crystals grown to equilibrium conditions can exhibit a lot of spirals on the surface.
1.3 Crystallography

Crystallography is the scientific study of the arrangement of atoms in solid materials. In this section some of the basic crystallographic theory is introduced along with some of the commonly used terms and notations which often contain subtle information.

1.3.1 Crystal Systems

A crystal structure is composed of a pattern of atoms or molecules arranged in a particular way, and a lattice which describes long range order and symmetry. A lattice can be thought of as an array of equivalent points in a given number of dimensions, in all cases in this project will only deal with 3 dimensions. A pattern placed at any one of these lattice points would be identical to a pattern located at any other lattice point. Lattice points can be seen to form small repeating box structures that repeat throughout the lattice structure, these tiny boxes are termed unit cells Figure 1-15.\(^{29,30}\)

![Figure 1-15 Showing the four lattice types that occur in 3 dimensional structures a) The primitive lattice given the symbol P contains a single lattice point b) The body centred lattice given the symbol I contains a lattice point at each corner and at the cell centre c) The face centred lattice given the symbol F contains a lattice point at each corner and the centre of each face d) The side centred lattice given the symbols A, B or C has lattice points at each corner and at two sides. The symbol is chosen to describe the plane the two side lattice points are in (A points in bc plane, B points in ac plane and C points in ab plane).](image)
Unit cells are not limited to perfect cubes, however they must contain parallel sides. These cells can be described by the lengths of the three sides a, b and c and the angles between them α, β and γ. Crystals can be classified as belonging to one of seven crystal systems, the simplest is the cubic system in which all sides are equal length and the angles between them are all 90°. The tetragonal and orthorhombic unit cells are based on rectangular geometry, not all lengths are the same but all angles are 90°. In the remaining systems both the lengths and angles vary from each other and in some cases no angles are 90°.

In some texts only six crystal systems are shown because they combine the rhombohedral and hexagonal into a single hexagonal system. The seven crystal systems are summarised in Table 1-1.
In the mid-19th century the French crystallographer Auguste Bravais showed that there can only be 14 unique different space lattices, these space lattices have become more commonly known as Bravais lattices. The 14 Bravais lattices can be obtained by combining the four lattice types Figure 1-15 with the seven crystal systems Table 1-1. The table in Table 1-2 shows a summary of the 14 unique Bravais lattices.
Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials

<table>
<thead>
<tr>
<th>System</th>
<th>Primitive (P)</th>
<th>Body Centred (I)</th>
<th>Face Centred (F)</th>
<th>Side Centred (A,B or C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubic</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>Tetragonal</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>Orthorhombic</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>Monoclinic</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td>Triclinic</td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td>Rhombohedral</td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
<tr>
<td>Hexagonal</td>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Table 1-2 Illustrating the 14 unique Bravais lattices.\(^{29}\)

1.3.2 Miller Indices

C.S Weiss was the first to give an approximation of a faces angular position to the crystallographic axes. Weiss parameters give the relative number of units in which a face cuts each given crystallographic axis.\(^{29}\)

Miller introduced a method of denoting faces that was much simpler than Weiss parameters and thus became the adopted method for expressing crystal faces. Miller indices are simply the reciprocal of Weiss parameters and cleared of any fractions. For a 3 dimensional unit cell three indices are require, these are denoted h,k and l and
a given face of a crystal is usually written \((hkl)\), whilst a set of related planes are written as \(\{hkl\}\). Figure 1-16 shows some examples of Miller indices.\(^{29}\)

![Figure 1-16 Illustration showing examples of miller indices](image)

1.3.3 Miller-Bravais Indices

In the hexagonal system there are three “a” axes perpendicular to the “c” axis, this requires a modification the way Miller indices to include this fourth axis. This modification is referred to as the Miller-Bravais index and is denoted by \((hkil)\) An important rule is that the first three parameters \(h,k\) and \(i\) must sum to zero.\(^{29}\)

\[h + k + i = 0\]

Consider the two cases in Figure 1-17, in case a) the shaded face intersects the \(a_1\) at +1, the \(a_3\) at -1 and doesn’t intersect the others thus the Miller-Bravais index would be \((10\overline{1}0)\). Whilst in case b) the shaded face cuts three axes, \(a_1\) at +1, \(a_2\) at +1 and \(a_3\) at -0.5 so the Miller-Bravais index would be \((11\overline{2}0)\).
1.4 Nanoporous Materials

Nanoporous materials covers a wide range of materials from different classes of materials, in this thesis only the materials from the zeolite, zeotype and metal organic frame work classes will be discussed.

1.4.1 Zeolites

Zeolites are porous aluminosilicate minerals. First discovered in the mid-18th century when Axel Fredrik Cronstedt observed upon heating the mineral stilbite Figure 1-18(c), large amounts of steam could be seen coming from within the material. Cronstedt coined the term zeolite derived from the Greek words ζεω (zeō) and λιθος (lithos), which together have the meaning “boiling stone”.

Zeolites occur in nature from reacting volcanic materials rich in silica and alumina minerals with saline water Figure 1-18. Natural reactions occur at moderate temperatures and pH requiring an exceptionally long time to complete the reaction (up to tens of thousands of years). Zeolites produced in nature find very little use in industry due to the fact they are rarely if ever found phase pure, some applications have found use for natural zeolites for example as a mixer in construction materials.
Industrial applications such as catalysis and purification usually require a high purity and more recent applications require more control over the properties that the zeolite exhibits. Silica and alumina are the most abundant mineralogical materials on earth, so the ability to supply the raw materials is limitless.

Since Richard M. Barrer succeeded in the first synthesis of the zeolites Mordenite and Chabazite in 1948,\textsuperscript{36-38} the door was opened to the discovery of a whole plethora of different zeolite frameworks some of which have not been discovered in nature. At present the International Zeolite Association Atlas contains 204 unique framework types that have successfully been synthesised and characterised.\textsuperscript{35, 39} There exists a database of hypothetical structures which currently stands at over two million unique frameworks.\textsuperscript{40}

Zeolite frameworks consist of primary building units (PBU) of tetrahedral aluminate and silicate $\text{AlO}_4^{-5}$ and $\text{SiO}_4^{-4}$ (T- sites) Figure 1-19.

![Figure 1-19 Primary building units of zeolitic frameworks a) silicate $\text{SiO}_4^{2-}$ b) aluminate $\text{AlO}_4^{2-}$.

The primary building units can link together to build up vast 3D structures by corner sharing oxygen atoms with other primary building units Figure 1-20. However Lowenstein’s rule states that Al-O-Al is forbidden so aluminate and silicate must alternate.
To think of the large highly ordered crystal structures in terms of primary building units makes little sense. Meier and Uyttehoeven compiled a catalogue of small structures called secondary building units (SBU) Figure 1-21 that consist of primary building units and can be used to describe most zeolite structures.

![Secondary Building Units](image)

**Figure 1-21** A list of Secondary Building Units (SBU). The tetrahedral atoms are linked together by oxygen atoms, other atoms are ignored. The number in parentheses denotes the frequency of occurrence.
These secondary building units are commonly observed in zeolite structures, this does not have any meaning on what species are present in solution and available for crystal growth. Instead they are simply a means of referring to certain structural features.

*Figure 1-22* Showing that the composite unit SOD can either be thought of as being constructed from four rings 4R or six rings 6R.
Composite building units can be thought of as being created from secondary building units Figure 1-22, this does not imply the linking of both 4R and 6R to form a SOD cage but the SOD cage can be thought of as being constructed from either 4R or 6R units. Most zeolitic frameworks can be described efficiently by composite building blocks Figure 1-23. Figure 1-24 shows three different zeolitic frameworks that all have the SOD composite in common.

By linking only SOD composites the sodalite framework is formed, by linking the sodalite composites with a double four ring (D4R) Zeolite A is formed and by linking the sodalite composites by double six rings (D6R) zeolites X and Y can be formed.
Due to the vast array of structures being reported each unique zeolite framework is given its own three letter code to avoid confusion. This is managed by the IUPAC Commission on Zeolite Nomenclature, founded in 1978. Examples of the three letter codes are LTA (zeolite A), LTL (zeolite L), SOD (sodalite), ERI (erionite) and OFF (offretite).

1.4.1.1 The Properties and Uses of Zeolites

Zeolites exhibit some extremely useful properties that have seen them find wide industrial and commercial applications, some of these properties are described here.

1.4.1.1.1 Porosity and Selectivity

One of the main properties of zeolites is porosity, pores are formed as a result of the ring structures ordered throughout the framework. The size of the pore is determined by the minimum rings size which constricts the pore. Only the tetrahedral (T–sites) atoms such as silicon or aluminium are counted when determining the pore size.
The pore networks in nanoporous materials can be extremely complex, they can be 1D, 2D or 3D depending on the framework. If there are more than a single pore structure in the framework, they may be interconnecting or non-connecting as shown in Figure 1-25.\textsuperscript{45}

The small pore sizes present in zeolites make them ideal for use as molecular sieves. The idea behind molecular sieves is that molecules of a certain size can pass through whilst those larger cannot. Due to the fact that zeolites exhibit a range of pores sizes, an ideal candidate can be selected for the application. Currently this property is unique to zeolites and is one of the properties that make them so valuable as a class of materials. For a zeolite to be effective as a molecular sieve it must have accessible pore structures in which all of the water has been removed to allow guest molecules to move freely within the pores.

Another application which exploits the porosity of zeolites is that of reaction selectivity. The idea behind this process is that the reaction takes place within the pore of the zeolite where steric constraints can act either on the reactants, intermediates or the products to exact reaction selectivity. One industrial example of this is that of the use of ZSM-5 in the production of para-xylene, this product is a major precursor in the production of plastics. The production of xylene from

\textit{Figure 1-25 showing a) 1D b) 2D interconnected c) 2D non-connected d) 3D interconnected e) 3D non-connected pore structures.}
ethylbenzene results in a mixture of the ortho, meta and para isomers of xylene Figure 1-26. Performing the reaction with ZSM-5 ensures that selectivity is well in favour of para-xylene. This is due to the fact that the bulky methyl side chains are aligned and thus the para isomer can leave the pores, whilst the more bulky ortho and meta isomers cannot leave the pores.  

1.4.1.1.2 Surface Area and Catalysis

The porous nature of zeolites means that they have extremely large internal surface areas, in the range of 200-800 m²·g⁻¹. This huge surface area is conducive to their role as catalysts where the active surface area is very important. Aluminosilicate zeolites can be easily transformed into acid catalysts by utilising their ion exchange property. The metal extra-framework cations can be exchanged to ammonium ions by treatment with ammonium nitrate. Once adequate ion exchange has been achieved, the zeolite can then by calcined (heated to extremely high temperature typically > 500 °C) to remove ammonia gas and leave behind the zeolite in its protonated form Figure 1-27. These extra-framework protons act as brønsted acids, and are the active component in catalysis. A common framework used in this manner is that of zeolite Y, it is used to convert long chain hydrocarbons into short chain molecules in the (C₁ – C₆) range which are more useful commercially.  

Figure 1-26 showing the conversion of ethylbenzene into the three isomers of xylene
Figure 1.27 showing the production of a solid catalyst from a zeolite (a) via treatment with aqueous ammonium nitrate (b) followed by calcination to produce the brønsted acid zeolite (c).

1.4.1.1.3 Cation and Ion Exchange in Water

Zeolites are often found in ion exchange applications where they make use of the extra-framework cation property. Each zeolite has a specific order of affinity for extra-framework cations which is highly dependent on the framework charge density. This framework charge density can be tuned to suit the application by changing the Si/Al ratio of the zeolite. Zeolite Y has found applications in treating nuclear waste water, zeolite Y tends to have an affinity for cations with low charge density due to its Si/Al ratio being around 3. Cations that fit this criteria include Cs, Rb, K, Na, and Li with the general trend of affinity going from Cs > Rb > K > Na > Li.\(^{46}\)
Zeolite A is another zeolite widely exploited commercially for its ion exchanging property, it is used on a massive scale in terms of tonnage in washing powder products. Zeolite A has a typical Si/Al ratio of about 1 so has a high affinity for cations with a higher charge density. Zeolite A used in clothing detergents removes Ca\textsuperscript{2+} and Mg\textsuperscript{2+} in exchange for its Na\textsuperscript{+} cations, which has the effect of softening water, increasing the efficiency of the detergents on cleaning clothes and reducing scale build-up on machine parts.\textsuperscript{47}

1.4.1.2 Synthesising Zeolites

Zeolites are synthesised mainly via the hydrothermal method, synthetic zeolites are produced to enable their structure and properties to be controlled as well as their purity. Hydrothermal synthesis is where a silicon, aluminium source are combined in water with a source of cations to form an amorphous gel. The gel is then subjected to high temperatures and pressures ( $> 60 \, ^\circ\text{C}$, $> 1 \, \text{bar}$) where crystallisation will occur. The reason such harsh conditions are used is due to the fact that some of the reactants used are hardly soluble at room temperature.\textsuperscript{48}
Hydrothermal synthesis is the most common synthesis method employed and will be the subject of the science described in this thesis. However other methods of synthesis do exist and will be briefly described here. Solvothermal synthesis is one such method, this method uses other solvents than water. One advantage it has is that generally larger crystals are produced by slowing down mass transfer. Another method is the ionothermal synthesis, this method uses an ionic liquid instead of water and utilises the strong templating effect ionic liquids have. This is one way to substitute the dangers of having high temperatures and pressures associated with hydrothermal and solvothermal syntheses. Finally microwave assisted syntheses are becoming increasingly common due to the much faster synthesis times and greater morphological control than conventional methods.

1.4.1.3 The Crystal Growth of Zeolites

The science surrounding the crystal growth of zeolites is mostly similar to those described in classical crystal growth discussed in detail in section 1.2. The processes that occur at each stage of the growth of zeolites are more complicated than simple salt crystallisations. The processes summarised in Figure 1-30 shows the processes leading from the reactants through to obtaining crystalline zeolite products.
Figure 1-30 showing the key stages of the crystal growth of zeolites. The ordering of aluminosilicate species increase in each successive stage. The structural drawings are totally arbitrary.

The main processes behind the driving force, nucleation and crystal growth will be discussed here with a specific slant towards zeolites.

1.4.1.3.1 Supersaturation

In the synthesis of zeolites, the initial phase is usually a thick amorphous gel which contains sources of the alumina and silica. This gel is then heated to start the crystallisation process. Upon heating this gel the solubility of the alumina and silica species increase which allows more to dissolve, this increases the supersaturation of the gel until it approaches that of $S_{\text{critical}}$ of the gel-solution phases. As the temperature increases still further species dissolve into the solution phase creating a supersaturated solution from which nucleation can occur. The solution-zeolite phase $S_{\text{critical}}$ is much lower than that of the gel-solution phase and thus the solution is supersaturated with respect to the zeolite phase, this means that crystal growth of the nuclei can occur from this supersaturated solution. This process is represented in Figure 1-31.
1.4.1.3.2 Nucleation in Zeolites

Nucleation in zeolites normally occurs by primary nucleation from the secondary amorphous gel phase Figure 1-30. The induction time readily observed in zeolite synthesis is the time it takes to form the precursor species along with the time it takes to form nuclei of a critical size for growth. The reason why nucleation occurs from the secondary amorphous phase is due to the increase ordering of the silica and alumina species. It is more likely that the larger more ordered species would be of the critical nucleus size and proceed on to further growth.\(^{47}\)

\(\text{Figure 1-31 showing a graph depicting the relative solubilities of the different phases of zeolite growth.}\)\(^{49}\)
After an induction time the nucleation rate (red dashed line) rapidly increases as the supersaturation is high, however as the number of nuclei created increases this nucleation rate quickly tails off in favour of growth of existing nuclei Figure 1-32.\textsuperscript{16}

1.4.1.3.3 Crystal Growth of Zeolites

After the nucleation stage of crystal growth, nuclei that reach a critical size will proceed on to further growth, this growth has been shown to proceed in the most part linearly until the stage where all of the nutrients in the gel have been exhausted. This has been highlight by the green line in Figure 1-32.\textsuperscript{16} Zeolite crystal growth is thought to proceed by the inclusion of small growth units only a few atoms in size. This addition of small units which have to be arranged into complex framework structures is a very time consuming process compared to simpler crystallisations.\textsuperscript{47} A multitude of parameters and factors can affect the outcome of zeolite syntheses, some of these will be briefly discussed in the next section.

1.4.1.3.4 Factors that Influence Crystal Growth

The synthesis of zeolites is very complex with many parameters that complicate the crystal growth even further. Some basic parameters that affect the outcome of zeolite crystal growth will be described below.
Reagents

The source of alumina chosen for the synthesis can have a large impact on the outcome of the synthesis. The generally used sources are usually a metal aluminate (sodium aluminate), aluminium salts or hydroxide. Elemental aluminium is avoided due to its extremely poor solubility. The use of aluminium salts can cause problems in the syntheses in alkali conditions as the formation of alkali salts can interfere with the phase produced. In some cases the wrong zeolite can be produced if not careful, the use of sodium aluminate can minimise this.\textsuperscript{50}

The source of silica chosen for the synthesis can also have a dramatic effect upon the outcome of the synthesis. Silica can be sourced in the form of soluble silicates (aqueous sodium silicate), colloidal silica solutions (30\% silica by weight), precipitated or fumed silica. The main effect the silica source has is that of the Si/Al ratio. Soluble silicates and precipitated silicas usually contain alumina impurities and are as a result used in the production of zeolites with low Si/Al ratio. Higher Si/Al ratio can be achieved by using fumed or colloidal silica, however other complications can arise from their use.\textsuperscript{50} Fumed and colloidal silica has to undergo depolymerisation before zeolite synthesis can occur, which mean that generally a higher temperature and longer time is needed for a successful synthesis.\textsuperscript{51}

Water is a very important parameter in the synthesis of zeolites, distilled water is the major source but some can be contained within the reagents. For example, colloidal silica can be comprised of large amounts of water.\textsuperscript{50} Water is an important component of any zeolite synthesis, but plays a very important role within the synthesis of crystals of high alumina content as they are more hydrophilic than their high silica counterparts. This means the framework in the high alumina case has a high affinity for water, which generally fills the pores and cavities. This extra-framework water can be removed post synthesis readily without damaging the framework. Increasing water in a synthesis has the effect of lowering supersaturation, which causes the crystal growth phase of crystallisation to be favoured over nucleation. As a results syntheses with larger amounts of water generally produce fewer but larger crystals.\textsuperscript{48}

Cations used in zeolite syntheses can be classified into two broad groups of inorganic and organic cations/templates. The main inorganic cations that are used are sodium
and potassium, however barium and calcium are often used. The most common reagent forms of cations are the alkali metal hydroxides, they serve as both the source of the metal cation and also to increase the pH of the synthesis gel which is generally a positive property. Cations are generally required to balance framework charges, however they are sometimes required to form certain structural units. This is apparent for the formation of cancrinite cages which is impossible without potassium ions present in the synthesis.\(^{52}\) The often used hydroxides also serve to increase the solubility of the alumina and silica species.\(^{48}\)

Organic cations or templates as they are often referred have several uses in the formation of zeolite frameworks. They generally affect the nucleation processes, lower the chemical potential of certain formation processes or even play a stabilising role in the zeolite framework. Tetramethylammonium (TMA\(^+\)) is a commonly used organic template in zeolite syntheses, it is found to reside within the β cage within the sodalite framework. This lends to the fact that the β cage must have formed around the template and thus trapping it within.\(^{46}\)

**Time**

In zeolite syntheses the general trend is that crystallinity increases over time. However to complicate matters most zeolites are metastable phases and as a result behave as dictated by Ostwald’s law of successive phase transformation. This is that the initial metastable phase is the least stable thermodynamically and over time will transform into progressively more thermodynamically stable phases.\(^{53}\) One synthesis where this is apparent is that of producing zeolite X, if left for too long then zeolite P will be present in the product. To circumvent this transformation, the synthesis is quenched at the optimal time for high purity of zeolite X.\(^{46}\)

Complicating things still further, the time in which the synthesis gel is allowed to age before mixing the reactants and heating has an effect upon the outcome of the synthesis.\(^{51}\) A number of important things occur during ageing of the synthesis gel which include the dissolution of the silica and the formation of nuclei which grow upon heating.\(^{53,54}\) Ageing has the following effects upon the crystallisation; increased nucleation rate, a reduced crystal size, an increase in the number of crystals produced, a decrease in induction period and a faster rate of crystallisation.\(^{48}\)
Temperature

An increase in synthesis temperature will generally increase the nucleation rate and to a much greater degree the crystal growth rate. This in turn leads to fewer and larger crystals and also a faster crystallisation time. Temperature also has an impact upon the properties of the zeolite produced, zeolites that are synthesised at higher temperatures are generally more dense as the amount of water would be lower due to more of it being in the vapour phase. This fact would appear to set limits upon how high the temperature of the synthesis should be due to the fact that eventually too great an amount of the water will be in the vapour phase.

pH

Zeolites are predominantly produced from basic solutions mostly in the range of pH 9 – 13. Changing the pH of the solution causes quite complex effects upon the crystal growth and is generally poorly understood, as the effect differs greatly across systems. As a general rule, increasing the pH leads to shorter induction times, nucleation periods and an increase in the rate of crystallisation, this means smaller crystals are produced. This is due to the fact that at higher pH the solubility of the silica and alumina sources is increased thus increasing the supersaturation, and as a result the aluminosilicate polymerisation rate.

Further complicating matters, pH can affect the zeolite framework and morphology of resultant crystals. The Si/Al ratio is seen to decrease with increasing pH, this is due to the rate that silica tetrahedra can be incorporated into the framework. The silica species are heavily deprotonated at high pH which affects the silica species ability to form links. Another effect observed in some syntheses is at higher pH, a decrease in aspect ratio is observed for reasons that are poorly understood.

Seeding

Seeding a synthesis, where a small amount of the desired crystalline zeolite product is added to the synthesis gel in order to influence the synthetic outcome more in favour of the desired product. The outcome of the synthesis can be further controlled by keeping the supersaturation low enough such that primary nucleation does not occur. This results in crystal growth being primarily of the seed crystals resulting in much larger crystals. At higher levels of supersaturation, increasing the number of
seeds added can yield reduced crystallisation periods and shorter if not eliminated induction periods.\textsuperscript{48}

**Agitation**

Most zeolite syntheses, especially those reported in the intentional zeolite association book of verified syntheses do not suggest carrying out agitation or stirring. Agitating leads to smaller and in most cases more uniform distribution in size of crystals. This is due to the increased mass transport and as a result increased supersaturation that agitation has, increasing the level of nucleation. There are reports of upon agitation of known syntheses that previously were performed without agitation of formation of totally different phases. This is known to occur in the zeolite X and MFI syntheses where zeolite A and TON are produced respectively.\textsuperscript{54}

1.4.2 **Zeotypes**

Zeotypes are porous crystalline materials that exhibit “zeolitic” properties but consist of other elements than silicon and aluminium.\textsuperscript{46} These materials became of great interest since the discovery of the aluminophosphate (AlPO\textsubscript{4}) zeotype in 1982.\textsuperscript{55} Aluminophosphate materials exhibit no net framework charge, this charge neutrality comes about from the fact that Al and P exist within a very strict 1:1 ratio. For every aluminate \(\text{[AlO}_4\text{]}^5^-\) which contributes a net charge of -1, there is a phosphate \(\text{[PO}_4\text{]}^3^-\) which contributes a net charge of +1 thus creating a charge neutral framework.

Another interesting class of zeotype materials is the zincophosphates, originally synthesised by Gier and Stucky in the early 1990’s.\textsuperscript{56} In the first cases framework analogues of the aluminosilicate materials such as faujasite (FAU), sodalite (SOD) and LiA (ABW) frameworks were reported.\textsuperscript{56-58} It was not long before frameworks that did not have aluminosilicate analogues were reported. Among these new frameworks were systems with extremely interesting properties, such as the chiral zincophosphate (CZP),\textsuperscript{59, 60} framework materials that contained huge pore structures up to 24 T- sites in circumference,\textsuperscript{61} and frameworks with helical pores.\textsuperscript{62}

Zincophosphate frameworks, unlike the aluminophosphates have an overall negative charge of -1. This is due to the fact that the \(\text{[ZnO}_4\text{]}^6^-\) provides an extra -1 unit of charge per tetrahedra than that of the \(\text{[AlO}_4\text{]}^5^-\). This negative charge is balanced by the inclusion of cations such as sodium, lithium and templating organic molecules.\textsuperscript{56}. 

\textsuperscript{53}
In most cases zincophosphates exhibit a strict 1:1 ratio of zinc and phosphorus. Interestingly zincophosphates have been synthesised that violate this strict 1:1 ratio, materials that contain ratios that aren’t 1:1 zinc to phosphorus exhibit interesting properties and features such as $\text{OZn}_4$ tetrahedra.

For most cases, the crystal growth of zeotypes is thought to occur upon the same lines as that of zeolites.

1.4.3 Metal Organic Frameworks (MOFs)

Metal Organic Frameworks (MOFs) are a class of compounds that have gained a lot of popularity over recent years. MOFs are nanoporous solids that are composed of metal ions/clusters that are linked together by organic linker molecules.

![Figure 1-33 showing the basic steps behind the formation metal organic frameworks.](image)

The organic linkers in MOFs are of a much larger size than the oxygen linkers of the inorganic porous solids like zeolites and zeotypes. This means that MOFs have much larger pore sizes (0.3 – 3.5 nm), larger surface areas (Langmuir, 100 – 6000 m$^2$.g$^{-1}$) and lower densities (0.20 - 1.00 g.cm$^{-3}$) than those reported for the inorganic zeolite and zeotype materials.
One of the interesting properties surrounding the study of MOFs is the almost limitless scope for the creation of new materials. This can be done by altering the metal ion/cluster or changing the linker molecules to form new materials. One doesn’t have to stop at simply changing the framework components. The organic linkers themselves can be functionalised with various different useful functional properties, such as those in the IRMOF examples shown.

**Figure 1-34** showing a catalogue of acid linkers used in the synthesis of various IRMOFs, the number corresponds to the structure prefix number.

**Figure 1-35** an example of different MOF materials that can be produced by changing the organic linker molecules. This particular example is that of IRMOFs, the number by each structure is the prefix for each IRMOF structure.
Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials

The potential applications for MOFs are wide ranging, a lot of research focus has been applied to the storage of strategic gases such as H₂, CH₄, CO₂, O₂ and NOₓ etc. The vision the researchers within this field have in mind is to on the one hand store gases seen as the future of the fuels and energy sector in a safe and in sufficient quantities. Whilst on the other be used as the active capture and storage substrate in green applications such as carbon capture, where gases such as CO₂ and CH₄ are captured and stored long term.

Due to the porous nature and the fact MOFs can be readily functionalised means that they show good promised in catalytic applications. However, only a small number have been found to be promising. The major issue with catalysis is MOFs tend to be thermally unstable or unstable in certain solvent or atmospheres.

The instability of MOFs has been exploited in some applications where they degradations is desirable. One such application area is that of medical applications, MOFs can be made of the metals that exist within the body such as iron, copper, manganese, nickel and zinc along with non-toxic organic linkers. One of the potential applications makes use of the ability to make nano-sized particles of MOFs (NMOFS) such that they can freely circulate within the body. These MOFs can be loaded with therapeutic or biomedical agents and delivered in a targeted and controlled way within the body. Other interesting studies in the field have seen the MOF functionalised with imaging agents and cis-platin smart-drug into nano-sized particles of MIL-101(Fe). This can be used in medical imaging to follow in real-time the therapeutic response to anticancer treatments.

Other applications of MOFs are in smart sensing and imaging devices. Some MOFs have interesting optical, electrical and magnetic properties that are detectable when the framework is interacting with analytes.

1.5 References


Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials


Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials


Chapter 1 – Introduction to Crystal Growth and Nanoporous Materials


Chapter 2

Computational Methodology
Chapter 2 – Computational Methodology

2.1 Critical Literature Review

The studies reviewed here form an overview of the literature critical to the development of the thinking throughout this project.

2.1.1 Gale – Monte Carlo Study on Urea

Gale et al.1-3 studied the growth and dissolution of urea crystals from solution. The simulations performed were based on the Kossel crystal model Figure 2-1. Molecular dynamics (MD) simulations using an atomistic approach were performed to calculate the activation energy of various types of growth and dissolution.

By considering all of the processes that govern the incorporation of material into a crystal surface Gale was able to build up a complete energetic map of the processes involved. For species to become incorporated into the crystal C1-5 in Figure 2-1, solution molecules S must first be adsorbed onto the surface. This requires a given activation energy to be overcome. Species can adsorb at a variety of different site types in Gale’s model denoted by A1-5, each site will have different energies associated with adsorption, this relates to the energy of de-solvation and other processes. Once adsorbed species can either leave the surface or become attached to the crystal, all of these processes have energies associated with them. Gale derives a set of energy levels governing the attachment of material at the various site types C1-5 whereby the energetic intermediates of adsorption and attachment are considered.
This energetic information can then be utilised in Monte Carlo simulations to investigate the properties of the energetic outcomes pre-calculated using molecular dynamics. The results of these simulations can thus be rendered as shown in Figure 2-2.

Figure 2-2 showing one of the result sets from Gale’s Monte Carlo simulation (top) shows surface from above (bottom) shows surface from side. Renderings of the above form can be extremely useful for comparison with experimental observations (AFM for example.

2.1.2 Boerrigter – MONTY

Boerrigter et al. developed a Monte Carlo program called MONTY. In the MONTY energetic model it considers entropic and kinetic aspects by taking the driving force into account. MONTY deals with a two phase system, where molecules are either in the crystal phase or in the solution phase (motherphase). Growth units in the crystal phase can only exist in discreet positions; this means that imperfections are not considered. Growth units in the motherphase are treated by their thermodynamic bulk properties. Monty was constructed to deal with more than the simple Kossel crystal model; it is also able to deal with growth in any crystallographic direction.

The theory of implementation of MONTY is described by starting with the Gibbs free energy of a growth unit in either phase, \( G_{\text{phase}} \)

\[
G_{\text{phase}} = H_{\text{phase}} - TS_{\text{phase}} = U_{\text{phase}} + PV_{\text{phase}} - TS_{\text{phase}}
\]  

Equation 2-1
Chapter 2 – Computational Methodology

Where $H$ denotes the enthalpy, $S$ is the entropy, $V$ is the volume. The pressure $P$ and temperature $T$ are assumed to be constant throughout the system. The entropic part of the Gibbs free energy difference between the two phases is fully described by the difference in entropy of the bulk phases regardless of surface position. The Gibbs free energy for a particular site, $i$, was the total interaction energy of the growth unit with its neighbours $\Delta U_i$ becomes,

$$\Delta G_i = \Delta U_i - T\Delta S + P\Delta V$$

Equation 2-2

Where,

$$\Delta S = S_{\text{mother phase}} - S_{\text{crystal phase}}$$

Equation 2-3

And,

$$\Delta V = V_{\text{mother phase}} - V_{\text{crystal phase}}$$

Equation 2-4

A growth unit that attaches to the crystal surface loses some of its motherphase interaction energy, which is proportional to the energy gained at the crystal surface. This effective change in motherphase energy can be expressed as,

$$U_{i}^{\text{moph eff}} = \frac{U_i}{U_{\text{crystal phase}}} U_{\text{mother phase}}$$

Equation 2-5

Where $U_i$ is the internal energy change going from the motherphase to the crystal phase for a given site $i$, $U_{\text{crystal phase}}$ and $U_{\text{mother phase}}$ are the bulk interaction energies of the crystal phase and motherphase respectively. Re-expressing,

$$\Delta U_i = U_{i}^{\text{moph eff}} - U_i = \left(\frac{U_{\text{mother phase}}}{U_{\text{crystal phase}}} - 1\right)U_i$$

Equation 2-6

The probabilities of growth and dissolution of a given site $i$ can be related to the Gibbs free energy of that site type and the Boltzmann factor as follows

$$\frac{P_G}{P_D} = \exp\left(\frac{\Delta G_i}{k_B T}\right) = \exp\left(\frac{\Delta U_i - T\Delta S + P\Delta V}{k_B T}\right)$$

Equation 2-7

Where $P_G$ and $P_D$ are the probabilities of growth and dissolution of a given site types respectively. To express the probabilities separately, Boerrigter et al. introduced two new parameters $\lambda_1 \lambda_2$, where $\lambda_1$ controls the probabilities based on site dependence. A value of $\lambda_1=0$ describes probability of attachment whilst a value of $\lambda_1=1$ describes
probabilities of detachment. In a similar way $\lambda_2$ describes the probabilities with respect to driving force $\Delta \mu$. Written separately the probabilities are expressed as

$$P_i^G = \exp\left(-\beta \lambda_1 (\Delta U - \Delta U_i) - \beta (\lambda_2 - 1)(\Delta U - T\Delta S + P\Delta V)\right)$$  \hspace{1cm} \text{Equation 2-8}

And,

$$P_i^D = \exp\left(-\beta (\lambda_1 - 1)(\Delta U - \Delta U_i) - \beta \lambda_2 (\Delta U - T\Delta S + P\Delta V)\right)$$  \hspace{1cm} \text{Equation 2-9}

Where

$$\beta = \frac{1}{k_B T}$$  \hspace{1cm} \text{Equation 2-10}

$$\Delta U = U_{\text{mother phase}} - U_{\text{crystal phase}}$$  \hspace{1cm} \text{Equation 2-11}

$$\Delta \mu = \Delta U - T\Delta S + P\Delta V$$  \hspace{1cm} \text{Equation 2-12}

2.1.3 Modelling Crystal Growth at the Centre for Nanoporous Materials (CNM)

Atomic Force microscopy (AFM) is a powerful tool to observe and understand crystal growth, AFM forms a central part of this research field at the CNM. Coarse grain modelling was introduced to assist in the understanding of the observations made via AFM. Early studies between modelling and experiment in the group started with zeolite A. Zeolite A was chosen because of its industrial importance and its structural simplicity.

2.1.3.1 Two Dimensional Coarse Grain Modelling of Zeolite A

Anderson et al.\textsuperscript{5,6} confirmed the surface spreading mechanism at low supersaturation on the (100) face of zeolite A by using AFM. The terraces each of 1.2 nm step height exhibited layer by layer growth. The step height was shown to represent half of the unit cell height for zeolite A. During these studies to understand the crystal growth of zeolite A, interesting surface features were observed in the way of curved terrace fronts, particularly at the intersection of two terraces. This being a well understood phenomenon in simple molecular crystals is a different problem entirely in systems as complex as zeolites.
To investigate this Anderson et al.\textsuperscript{5, 6} constructed a 2D Monte Carlo program where a 2D grid represented the surface of the (100) face of zeolite A. Each grid element was chosen to represent a 1.2 nm square of crystal surface which consisted of a sodalite cage and two double four rings (D4R). This simplification is known as coarse graining, the logic used to choose the coarse grain building unit was to correlate the terrace height observed via AFM with the zeolite structure Figure 2-4. This simplification was crucial because an attempt to understand crystal growth by atomic level simulations would have been extremely complicated.

The model invoked the use of the Kossel model to define five simple site types each different by their connectivity to neighbouring sites. The site types could then be assigned probabilities for simulations and these could be changed to investigate the properties of crystal growth. Anderson proposed that the curved features came about as a result of kink sites which are known in classical crystal growth to cause such phenomena.
These kink sites come about when two growing terrace fronts coalesce Figure 2-5, once the terraces coalesce it was suggested kink sites would quickly start to grow out from the point of merger and thus create a curved terrace front.

Not only did Anderson et al. manage to recreate the experimental observations, but they managed to deduce information about the relative rates of growth of each site type and the effect these had upon the outcome of the crystal growth.

2.1.3.2 Three Dimensional Model of Zeolite A
Chin Chong\textsuperscript{8} created the original 3D Monte Carlo simulation program for the zeolite A crystal in the line of studies at the centre for nanoporous materials at Manchester.
To achieve this, Chong built upon the work done previously by Anderson in the 2D simulations. To extrapolate the model to 3D Chong initially added in a sixth site type in terms of those prescribed by the Kossel model, and also allocated 3D memory space for the crystal to grow on.

It proved difficult to access the range of morphologies desired with only single sphere neighbour connectivity as prescribed by the Kossel model. So Chong incorporated second sphere neighbour connectivity into the model. This enabled the site types that are associated with each of the three main facets (100), (110) and (111) to be given probabilities that can be varied as a set of site types, such that the rates of growth of each facet can be controlled separately.

Chong also introduced the idea of mimicking driving force by altering the probabilities of the nucleation site types with respect to the site types of higher connectivity. This study was to understand the crystal growth process starting with a single nucleus. This program managed to successfully simulate the zeolite A morphology via layer by layer growth in 3D whilst showing good surface correlation with AFM. The relative growth rates were calculated via the attachment energy of the building blocks, however this model did not consider dissolution.

2.1.3.3 Modelling Crystal Growth in Zeolite A

Ayako Umemura\textsuperscript{9} constructed a new program to simulate crystal growth in cubic systems due to technical limitations in earlier programs. The new program took a more modular approach which made the overall code much simpler and shorter, overcoming some of the technical limitations of the old programs. In addition to the 3D program, 2D programs were constructed to investigate the surface spreading of cubic crystals on the (100), (110) and (111) faces. The site type definitions were based on the ones used in Chong’s program however detachment site types were added for each attachment site type to consider both growth and dissolution.
Figure 2-7 showing a) activation energies b) internal energies for FSC sites used for calculation in Ayako’ model. S, E, K, U, Su and B denote Surface, Edge, Kink, U, Surround sites and Bulk of Kossel model, respectively. (−a+b) denotes the gap between the activation energy for growth, a, and dissolution, b. ΔU is the difference of the internal energies in the mother liquor phase, Umoth, and in crystal phase, Ucryst. 1/2ΔU is an ideal equilibrium state for this model.9

Umemura showed that with reference to studies by Piana and Gale2 and Boerrigter4, an appropriate energy correlation diagram in terms of both first and second sphere coordination can be created Figure 2-7. A temperature term was also included in this program to investigate the influence of temperature upon crystal habit Figure 2-8.
Umemura showed that the formulae derived by Meekes et al. Equation 2-8 and Equation 2-9 can be expressed in a much simpler form:

\[ P_i^G = \exp \left( -0.5 \frac{U_i}{kT} + 0.5 \frac{\Delta \mu}{kT} \right) \quad \text{Equation 2-13} \]

And

\[ P_i^D = \exp \left( +0.5 \frac{U_i}{kT} - 0.5 \frac{\Delta \mu}{kT} \right) \quad \text{Equation 2-14} \]

The only parameters that are variable in these expressions are the internal energy \( U_i \) in which each site type \( i \) has a unique internal energy, and the driving force \( \Delta \mu \). This was the first time that the driving force was incorporated in an organic way into the models at the CNM in Manchester.

This program was shown to be successful at simulating the crystal growth habits of zeolite A with respect to driving force and temperature extremely well. This program and the studies described above provide the foundations in which the entirety of the work presented in this thesis is based.

### 2.2 Introduction to the Programs and Functionality

In this project two separate program lines were developed, the simulation program and the visualisation program. The operation and features will be discussed separately.

#### 2.2.1 The Simulation Program

The program developed as part of this study is a continuation of the long running research done at the CNM on this topic. A number of improvements both scientific
and computational have been made to this program. The basic operation of this program will be discussed in this section along with some detailed information as to what improvements have been made over the course of this study.

2.2.1.1 Basic Operation
This program consists of a single Fortran 90 source file that can be compiled by most commercial and open source Fortran compilers, the source can be found in appendix 1. For this study the program was mainly compiled using the NAG Fortran IDE (locally) and using the Intel Fortran compiler (supercomputer). The program was mainly run on Epsilon and The University of Manchester Condor pool. Epsilon was a Beowulf cluster consisting of 12 nodes, 4 of which had 12 Gb RAM, 4 had 8 Gb of RAM, 2 with 6 Gb and 2 with 4 Gb of RAM however this cluster was phased out in January 2013. Due to the nature of the memory requirements of this program generally only the 6 Gb and larger nodes could be utilized. The condor pool consists of 38 dedicated high performance and high memory cores accessible 24 hrs per day, and overnight/weekends a further 2000 cores are brought online consisting largely of machines in the student computer clusters situated around the campus. Similarly this program could only be run on the larger memory nodes at this current time due to its large memory requirements.
The simulation code can be classified into the major sections illustrated in Figure 2-9, the basic functionality common to all versions of the program created in this study is explained below.

**Input**

In this section of the program all of the input information is entered into the program. This can be either done manually in the console window if done locally, or alternatively these parameters can be stored in a text file and then supplied on the command line. The advantage that having the parameters stored in separate files is that it suites the architecture of condor, whereby many instances of the same program can be run at the same time on different nodes by supplying the correct input file.
The input file contents vary from version to version of the program depending on what system it targets but this can be ascertained by reading the source file. Typical information that the input files contain are the names of any output files, size of simulation field, number of iterations, driving force mode and starting value, and information regarding how many frames of output and the frequency that these should be taken.

Initialisation

This part of the program is where the starting configuration for the simulation is set up. The arrays for the simulation field and the site types are allocated and then populated with the starting configuration. This starting configuration is usually the first grown cage that is positioned in the centre of the simulation field, thus by doing this the program has to be started with the correct neighbouring site type information. All counters are set and any pre-calculation is done in this part of the program.

Monte Carlo Random Selection

During the main loop of the application the Monte Carlo method is applied to select which site is going to be either grown or dissolved. This is done by firstly generating a random number (R), this random number is always in the range \(0 \leq R < 1\). The total probability \(P_t\) of all of the sites present is calculated by taking the number of each site type \(C_i\) and multiply it against the probability of each site type \(P_i\) and summing over all site types \(n\) Equation 2.15.

\[
P_t = \sum_{i=1}^{n} C_i \times \sum_{i=1}^{n} P_i
\]

Equation 2.15

The probability for each site type is then found by dividing the total probability for each site type by the total probability Equation 2.16.

\[
P_i = \frac{C_i \times P_i}{P_t}
\]

Equation 2.16

The random number R is then compared against \(P_i\) to select a site type for growth/dissolution. Once a site type has been determined then a second random number is generated and used to then choose which specific site within the selected site type should be used for growth/dissolution.
Updates and Neighbour Checking

After performing a growth/dissolution event, a significant amount of updating needs to be performed. Firstly the site that was involved in the event is updated so that all of the information stored in the arrays for this particular site is correct. This coordinate is then passed to the neighbour checking routine which then sequentially works through each of the sites that surround this site and updates their information to reflect any change in site types.

Output

At the output step of the program, the current iteration number is checked to see if it corresponds to a frame that should be stored as an output. A snapshot of the current simulation is generated from the site type array, the bulk sites are excluded since they do not yield much information for visualisation purposes and just make the output file unnecessarily large. The output file consists of a list of the 3D coordinates for each site that forms the surface layer of the crystal at its current stage of growth/dissolution. This file is then used with the visualiser program to render the results; this will be explained later.

2.2.1.2 Connectivity

The connectivity is one of two pieces of information that is required about a given system in order to perform simulations. There are two main lines of program, hard coded and generalised. For the generalised code the connectivity will be discussed later in this section under the general algorithm. In the hard coded versions of the program the connectivity is hard coded on a per system basis, this means a new program has to be made each time a new system investigated.

The model used for the connectivity in different systems stem from the dissolution studies performed by Brent on the dissolution of zeolite L crystals. In that particular study, Rhea performed repeated scans of a single line across a single width terrace under very mild dissolution conditions. It was then possible to record the terrace height as a function of time Figure 2-10. It can be observed that not all the step heights are equally as stable as they have different lifetimes under the same conditions.
These heights could then be correlated against the framework structure of zeolite L to discover what the surface termination might consist of for each step height. Figure 2-11 shows the terminations and their respective heights, they are numbered to coincide with the numbering used in Figure 2-10. It can be seen that the longest lived and thus more stable surface terminations are 1, 2 and 3. These three structures contain closed cages; in this case the closed cages are cancrinite cages. Surfaces terminating in closed cages are entirely made up of Q₃ and Q₄ connected T-sites. These are a lot more stable under dissolution conditions than those with connectivities of Q₁ and Q₂.

It was these observations that led to the current thinking about the connectivity used in the simulations throughout this study. In previous work Umemura had used both first and second sphere connectivity to define site types, however in this work the second sphere would be dropped since it is mainly used in studies of ionic crystals where second sphere neighbouring atoms are directly adjacent to the first sphere neighbouring atoms.
Using this logic of closed cages being the most structurally stable unit on the crystal surface, it is then possible to argue from the point of view of growth and dissolution that the formation of the closed cage unit would be the rate determining step. That is, in reality many addition and dissolution of simpler units may take place up until the formation of a closed cage which becomes very stable and long lived at the surface of the crystal.

In this study closed cages formed the coarse-grain building block, thus it is now possible to separate the connectivity of multiple types of building blocks within the same structure whereas in previous studies portions of the structure that correlate to AFM step heights were coarse grained into a cubic building block making this difficult, specifics of each system will be discussed in the results and discussion chapters.

Following on from previous studies, the Kossel model will be invoked to describe site types based on neighbour connectivity. Figure 2-12 shows the simple site types represented in the Kossel model. These are often used in simple cubic systems to describe different site types. However in systems that are more complex than cubic it
requires a geometric modification, this study used such modifications to describe the site types for more complex systems.

In addition, two energetic approaches were investigated; the energy of attachment and energy of destabilisation. These will be described in detail in the following section. However, the energy of attachment follows the logic of the Kossel model closely, where a given site attachment energy is directly related to the number of neighbour connections. The destabilisation approach relies on working out how destabilised a given site is from the crystal bulk and is directly related to the number of T-sites that need to be added to complete the cage. For this approach the Kossel model was modified so that instead of representing the coarse-grain unit with cubes it used the actual geometry of the cage, so each vertex of the shape represents a physical T-site atom. Thus each site type can be represented by how many vertices are present due to the number of neighbours being present.

![Site Types](image)

**Figure 2-12 Showing some simple site types from the Kossel model.**

This approach allows much higher complexity in the way systems are dealt with programmatically, it allows the introduction of multiple types of cages that are linked together within the same simulation, they can be assigned their own set of site types and thus their own energy level system.
2.2.1.3 Energetics

The energetic information is the second of the required pieces of information about a given system in order to perform simulations. The energetics are what allows probabilities and therefore rates to be assigned to each site type. Over the course of this study two different approaches with respect to energetics were investigated, the first considered energy of attachment and the second considered energy of destabilisation. Both approaches will be discussed in this section, however most of the later work in this study was performed using the energy of destabilisation approach.

Energy of Attachment

The energy of attachment model is an extension of the energetics model developed over previous studies conducted in the group, particularly those by Ayako Umemura. The basic idea behind this model is the attachment of a coarse-grained unit of structure via the formation of linkages or bonds. The coarse-grained building unit has a given number of attachments to neighbouring units. Energies can then be assigned to these bonds and thus it is then possible to construct an energy level diagram of all of the permutations in neighbour connectivity.

For example if we consider the LTL structure, it can be seen that the cancrinite cages are linked together in the a and b directions by 2 bonds which in this study will be referred to as bidentate connections, but in the c direction they are linked by 6 bonds through D6R’s which will be referred to as hexadentate, Figure 2-13 illustrates this.

Figure 2-13 on the left the structure of zeolite L, on the right the two types of linkages are displayed along with the crystallographic axes.
By considering that each cancrinite cage can be surrounded by 3 other cancrinite cages in the plane of the (a,b) directions and 2 other cancrinite cages along the c direction. This then opens to the idea that one can permute all of the different combinations of connectivities of bidentate and hexadentate linkages based upon the occupations of all the neighbouring sites. Then if an internal energy $U_B$ and $U_H$ is assigned to each bidentate and hexadentate connection respectively then the permutations can be plotted on an energy diagram Figure 2-14.

Figure 2-14 the energy level diagram for the zeolite L system as derived using the energy of attachment model. The internal energies of bidentate $U_B$ and hexadentate $U_H$ have been labelled. The arbitrary zero line has also been indicated. Indicated in brackets are the number of bidentate connections (left parameter) number of hexadentate connections (right parameter).

The probability formulae used in this model for growth and dissolution are:

$$P_i^G = \exp \left(-0.5 \frac{U_i}{kT} + 0.5 \frac{\Delta \mu}{kT}\right)$$  \hspace{1cm} Equations 2-17

And

$$P_i^D = \exp \left(+0.5 \frac{U_i}{kT} - 0.5 \frac{\Delta \mu}{kT}\right)$$  \hspace{1cm} Equations 2-18

Destabilisation Energy

The energy of destabilisation model is an attempt to approach the problem from the opposite direction. In the attachment energy approach each site type is stabilised by the number of connections/bonds to neighbouring sites. This approach however
makes it very difficult to include effects from templates and cations as they may not influence the bonds that are being considered as connecting, but they may influence the formation of closed cages. The destabilisation energy approach was developed for just this purpose.

By considering the LTA system as an example, it is made up of three different kinds of cage: the $\alpha$-cage, $\beta$-cage and D4R. For each of these cages it is possible to envisage two extremes for a given cage, these are fully isolated and fully surrounded (bulk), Figure 2-15 shows this for the $\alpha$-cage. For the cage that is in the bulk this can be thought of as fully stabilised as it is fully surrounded by crystal, this site type would be placed at zero units of destabilisation. For the isolated cage, this is the most destabilised configuration and would be placed at the highest destabilisation energy. Any configuration at the surface of the crystal would fall somewhere between the two extremes.

![LTA structural make up by cage](image)

*Figure 2-15 showing the LTA structural make up by cage, a) shows a portion of LTA crystal b) shows hypothetical extremes in connectivity for the $\alpha$-cage, top is a totally isolated cage, bottom is an $\alpha$-cage that is considered bulk and middle is a surface cage.*

Where exactly each site falls is related to the number of $Q_3$ connected T-sites that need to be removed into solution (converted to $Q_0$), this number is related to the number of T-sites that belong to neighbouring cages (only the T-sites belonging solely to the cage being considered can be considered). So for the isolated $\alpha$-cage 48 T-sites can be considered whilst for the bulk site 0 T-sites can be considered. There
are then 48 different site types from an energetic point of view for the LTA cage, this same process can be repeated for the other cages, a β-cage has 24 T-sites so 24 distinct site types can be distinguished and the D4R has 8 T-sites so 8 distinct site types can be distinguished from an energetic point of view. This can be plotted on an energy level diagram Figure 2-16.

![Energy Level Diagram](image)

*Figure 2-16 showing an energy level diagram for the LTA system using the destabilisation energy approach. The internal energy separation $U$ for each cage has been marked by red arrows. The thick black line show the positions of the cage configurations depicted.*

The probability formulae for growth and dissolution have to be modified by change of signs for this approach as showing in Equation 2-19 and Equation 2-20. This is to ensure that for a higher rate of growth a smaller destabilisation energy is required.

$$P_i^G = \exp \left( -0.5 \frac{U_i}{kT} + 0.5 \frac{\Delta \mu}{kT} \right)$$  \hfill Equation 2-19

And
Chapter 2 – Computational Methodology

\[
P_i^D = \exp\left( +0.5 \frac{U_i}{kT} - 0.5 \frac{\Delta \mu}{kT} \right) \tag{Equation 2-20}
\]

2.2.1.4 Driving Force

To model what happens to crystals under high, equilibrium and low supersaturation the idea of driving force was introduced. This driving force is the \( \Delta \mu \) term in Equation 2-21 and Equation 2-22. The driving force can be specified at the start of a simulation, during the simulation the driving force value can be changed by the program using Equation 2-23 and Equation 2-24. The driving force at a given iteration \( i \) is given by \( \Delta \mu_i \), is controlled by the starting driving force \( \Delta \mu_s \) and the parameter \( x_s \) which dictates how rapidly equilibrium should be reached. Typically a growth event would decrease the driving force, whilst a dissolution event would increase the driving force much in the same way these events would affect supersaturation in reality. There are some exceptions to this depending on what driving force mode the program is operated under.

\[
P_i^G = \exp\left( -0.5 \frac{U_i}{kT} + 0.5 \frac{\Delta \mu}{kT} \right) \tag{Equation 2-21}
\]

\[
P_i^D = \exp\left( +0.5 \frac{U_i}{kT} - 0.5 \frac{\Delta \mu}{kT} \right) \tag{Equation 2-22}
\]

\[
x_h = \frac{x_s - 1}{1 + \Delta \mu_s} \tag{Equation 2-23}
\]

\[
\Delta \mu_i = \frac{x_i - x_s}{x_h} \tag{Equation 2-24}
\]

The driving force may be operated under four different modes Figure 2-17, these different modes were designed to mimic as closely as possible typical experiments performed in the lab by experimentalists within the group. Mode 1 is the more natural driving force profile, where the driving force starts off at some value and is allowed to approach equilibrium. This is an incredibly important feature as the program can find the equilibrium driving force, utilising simple mathematics. Mode 2 is for investigating the effects upon a crystal of applying a constant driving force. Mode 3 is a combination of the first two modes, it is a two stage scenario starting with constant driving force of specified value and held for a specified time after
which the driving force is allowed to equilibrate. Mode 4 is a 3 stage experiment, the first two stages being the same as mode 3 and the third stage incorporating an extra constant driving force stage. These are very useful modes for simulating similar scenarios to those that are performed in the lab and via in-situ AFM, where a crystal is typically synthesised hydrothermally and then placed in a growth cell on the AFM and exposed to growth solutions of various concentrations.

![Diagram showing the four different modes of driving force that may be applied.](image)

**Figure 2-17** showing the four different modes of driving force that may be applied. a) starts off high and achieves equilibrium in a specified time b) constant driving force of a specified value c) two stage, stage 1 constant driving force of specified value, stage 2 equilibration stage d) three stage, stage one constant driving force of specified value, stage 2 equilibration stage, stage 3 constant driving force of specified value.

The driving force affects the rate of growth/dissolution of all of the site types in the simulation. There are three possible scenarios that the driving force may cause for a given site type, if the driving force is higher than a given energy level then the probability and therefore rate of growth is high and the rate of dissolution low for that site type. If the driving force is lower than that given energy level then the rate of growth is low and the rate of dissolution of that site type is high. If the driving force is close or equal to the energy level for that site type then the growth and dissolution rates are in equilibrium Figure 2-18.
2.2.1.5 General Algorithm

The general algorithm was developed with the aim of providing a quick and easy way to describe generic systems in terms of connectivity and energetics without the need for reprogramming/recompiling specific hard-coded source code. This version of the code incorporates all of the features described so far with the exception of the connectivity and the energetics. This program needs a connectivity file describing the system to be studied, within the file information about how many different cage types and how they connect to neighbouring cages is specified. In addition to this a description of how each T-site’s growth status is calculated. The source code for this version can be found in appendix 2, connectivity files for the zeolite A and sodalite systems can be found in appendix 3 and 4 respectively.

The basic structure of the connectivity input file can be explained as follows with reference to the sodalite input file in appendix 4.

The first line of the file should always be the number of different cage types, even if the same cage is present but in different orientations then these should be included as a different type so that their individual connectivity can be described. In the case of sodalite this first line is 1.

The next line should then contain the first line of information for the first cage, this typically contains three numbers that represent the number of neighbours, the number of T-sites and the number of cages each T-site is fused to in that order. At the current time this program does not support a per T-site fusion number so systems that have T-sites of differing fusion number are currently not supported. For sodalite this

\[
\begin{align*}
P^G &= \text{Lowest} & P^D &= \text{Highest} \\
P^G &= \text{Low} & P^D &= \text{High} \\
P^G &= \text{Low} & P^D &= \text{Low} \\
P^G &= \text{High} & P^D &= \text{Low} \\
P^G &= \text{High} & P^D &= \text{Highest}
\end{align*}
\]
would be 14 (number of neighbours), 24 (number of T-sites), 3 (neighbouring cages that share a vertex of the cage or T-site).

Following this is a list of information about neighbouring cages consisting of 4 columns; the first three columns contain relative coordinates of each neighbour whilst the fourth is an id number that identifies the cage type of each neighbour. This id value should correspond to the order that each cage type is specified within the connectivity file. For sodalite there is only one cage type so this column is all 1. There are 14 neighbours so there are 14 rows of coordinates.

Following these is then a list of T-site information, this list is composed of as many columns specified by the T-site fusion parameter on line two mentioned earlier. In the case of sodalite this is 3 because each T-site is fused to 3 other neighbouring cages not including the one being counted. Each row in this list specifies, by number, which 3 neighbours each T-site is connected to. These three numbers correspond to the positions of these neighbours coordinate information in the previous list.

If the system requires more cages to be described, these are then simply appended onto the end in the same fashion by providing information about the cage followed by a list of neighbour coordinates then by a list of T-site connectivity information.

See Appendix 3 for the input file for LTA which describes 5 different cage connectivities. These input files could be easily generated automatically possibly from a crystallographic information file (.cif) in future versions.

The energetics used in this program is the destabilisation energetic approach which was described earlier, this allows the algorithm to build its energy level diagram from the connectivity information, due to the fact that each energy level is related to how many T-sites are being added, this means for each cage the number of energy levels corresponds to the number of T-sites within that given cage. Two extra energy levels that have no probability are utilised per cage to make output of the results file much simpler.

2.2.1.6 Performance Enhancements
A number of performance enhancements have resulted in the ability to run much larger simulations without requiring more memory, also the average time in which a simulation can be performed has also been drastically improved. Some of these enhancements are described in this section.
Chapter 2 – Computational Methodology

Memory

By replacing the 2D static array used to store the site type information with a 1D allocatable array of allocatable arrays it is possible to save a lot of memory space. This is because enough space had to be allocated at the beginning of an experiment such that all of the sites could be stored. If the experiment was a large one then this would quickly limit the size of the array that can be allocated. In addition to this some of the site types are highly populated such as the bulk site types and some are sparsely populated such as the nucleation site types Figure 2-19. This is a complete waste of memory.

Figure 2-19 illustrating memory utilisation a) shows an ordinary 2D array with its size being dictated by the largest population, this creates wasted space (white) b) shows how the allocatable arrays can be used to create a storage structure of similar shape as the data to save space.

In addition to this, the fact the array of arrays is now fully allocatable allows for dynamic resizing. When a particular row is full then it is passed to a resizing routine which adds on a number of elements. Currently this routine can add chunks of 1 million elements to the resized row, this is so that the program is not always busy doing memory management and thus utilises most of its time for the simulation.

Speed

A number of speed increases have been obtained across various versions of this program, some of the most notable results have been obtained by eliminating large blocks of logical ‘if’ statements to assign site types. These were replaced by a lookup table resulting in a modest speed increase. Loop unrolling also obtained significant performance increase. Some of the loops in the program were successfully replaced with numerical based methods resulting in a faster running simulation. The fastest of all speed increases came from a mathematical trick involving the exponential part of
the site type probabilities. The formulae for a single site type probability are of the following general forms for growth and dissolution respectively.

\[ P_i^G = \exp \left( -0.5 \frac{U_i}{kT} + 0.5 \frac{\Delta \mu}{kT} \right) \]  \hspace{1cm} \text{Equation 2-25}

And

\[ P_i^D = \exp \left( +0.5 \frac{U_i}{kT} - 0.5 \frac{\Delta \mu}{kT} \right) \]  \hspace{1cm} \text{Equation 2-26}

The only part of those two equations that change iteratively is the driving force term to the right hand side, due to the fact that the driving force is globally the same to all site type calculations it is possible to split this calculation up using the power rules. The left hand side can be evaluated once at the initialisation step of the program with each value stored in a 1D array of size corresponding to the number of site types. Whilst the right hand side can be computed once per iteration, then the corresponding parts of the equation can be multiplied together rather than computing many exponential calculations. This results in a very large decrease in the time taken to perform a single simulation.

With the improvements described the following performance increases were possible based on comparable hardware.

<table>
<thead>
<tr>
<th>Improvement</th>
<th>Old Program</th>
<th>New Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of simulation 2D</td>
<td>1-2 µm²</td>
<td>15-20 µm²</td>
</tr>
<tr>
<td>Size of simulation 3D</td>
<td>0.3-0.5 µm³</td>
<td>0.8 – 1.5µm³</td>
</tr>
<tr>
<td>Runtime 100,000,000</td>
<td>27 min</td>
<td>7 min</td>
</tr>
</tbody>
</table>

\textit{Table 2-1 some of the increases in performance achieved through optimisation}

2.2.2 The Visualiser Program

Previously to this study rendering of the simulation output was performed using Surface Evolver,\textsuperscript{12} the use of this program for this particular purpose had its limitations. The colour palette was very limited with only very basic and unattractive colours available, due to this fact images required further processing in an image editing software to achieve the desired results.
The overall image in Surface Evolver was constructed by placing a small cube at each grid coordinate representing a growth unit. To do this in Surface Evolver one must compile an input file with a list of all of the vertices, edges and faces of each cube. There are 8 vertices, 12 edges and 6 faces to each cube. However, Surface Evolver requires the faces be split into triangles so there are 12 triangles for each cube. This means that for each cube there are 22 lines of data in the dataset. To save space and speed up load time 7 vertices were included thus leaving out the rear facing edges and faces obtaining a 10 fold speed up over rendering with Wolfram Mathematica. However, the downside was that upon rotation the rear faces appeared incomplete and therefore could not be imaged for appearance reasons.

Even with that simplification a relatively small experiment producing 47,881 data points at the surface would require 1,197,028 lines and a file size of 44 Mb to render. A much larger data file would require a very long time to parse into the program for rendering. If more complex geometries than cubes are used, for example a β-cage, the file size to describe the surface will increase drastically.

All of these issues, both the manual nature of post processing and the slow nature of the rendering are not conducive to high throughput scale computation. A bespoke program was created in C/C++ and using the OpenGL API to solve these issues with the sole aim of making the post processing of data on a high throughput scale a realistic one. In this section the functionality and features of this program will be discussed.

The visualiser is constructed in C/C++ and compiled using Microsoft visual C++ Express 2010, the source can be found in appendix 5, it utilises the rendering power of OpenGL to achieve the desired results. In the current versions the user interface is constructed based on calls to the Windows API making this currently unavailable on Mac OS and Linux platforms, however aside to this the rest of the code is portable to other platforms. The input file is now simply the raw output from the simulation code containing only the growth coordinates, this drastically increases time based performance and disk storage requirements as will be discussed later in this section.

The main mode of operation of this visualiser is to place at each coordinate the correct cage type in accordance with the crystal structure, it does this by storing within display lists the draw commands for each cage type and then simply
translating to the correct coordinates followed by a call to draw the display list contents. After every single coordinate has been processed in this fashion the crystal is then rendered to the screen.

2.2.2.1 Features
The main features of this program and their respective use are described below:

Real time 3D rotation

The ability to rotate the crystal about any of the 3 major axes in real time allows features on any face of the crystal to be explored Figure 2-20 shows this feature. The same rotation matrix is stored between frames of the experiment such that a particular feature can be easily monitored without having to reorient the scene each frame manually. To use the rotation functionality, simply hold the left mouse button and move the mouse to rotate the scene.

Real time zoom

The ability to zoom in on features of the simulation is useful in particular on very large simulations where the surface structures are not apparent without a closer look Figure 2-21 shows this feature. The zooming is performed in real time and also the level of zoom at a given time memorised between frames of the experiment so that a specific feature can be tracked. To use the zoom function, on the keyboard ‘z’ zooms in whilst ‘x’ zooms out.
Figure 2.21 Image from visualiser software demonstrating the zooming capability a) shows a fully zoomed out sodalite crystal b) shows the contents of the black box in (a) and c) shows the zoomed contents of the black box in (b).

Translation

The ability to translate the crystal along the x and y axes is useful when zoomed in to a surface, this gives the ability to move and focus upon any area at any zoom level. The translation position is also memorised between frames of an experiment to allow progressive features to be tracked. Figure 2.22 shows an example. The translation functions have been mapped to the w, a, s and d keys. The x axis (left and right) is mapped to ‘a’ (left) and ‘d’ (right) whilst the y axis (up and down) is mapped to ‘w’ (up) and ‘s’ (down).

Figure 2.22 showing a crystal of zeolite A a) shows a view centred on the (111) facet b) shows a view with the crystal translated downwards providing a view across the (100) facet

Processing modes

Single file and single frame, this processing mode allows for simple manipulation of a single file containing a single frame, all of the functions are available to use in this
mode in real time. The resulting crystal can be saved as a bitmap file, multiple image files can be saved exhibiting specific features if required.

Single file and multiple frames, experiments that capture multiple frames of a crystal’s evolution over time have these frames all stored within a single output file with a special marker at the end of each frame, the visualiser detects these markers ensuring each frames data is correctly stored. It is possible to switch between frames using the left and right arrow keys on the keyboard. If there are a large number of frames then these can be automatically all saved as image files with a single click, each frame will have its frame number appended to the file name, each frame will be drawn with the same rotation, zoom and translation settings so that specific features can be tracked over a large number of frames.

Multiple file (batch), this mode can handle both of the previous situations but over a very large number of input files. In a single click the folder containing all of the experiment files can be processed, the visualiser will load each file in turn and create a bitmap file for all frames present before moving on to the next. This mode also uses the same rotation, position and translation settings across all files and frames. To set these settings up before a large batch process, it is recommended to load up one of the files to check if the settings are adequate and performing alterations before starting the batch process going. This is by far the most useful feature for large scale results post processing.

**Colouring**

The colouring works in two main modes, layer by layer and cage type each will be discussed below.

Layer-by-layer colouring, this mode of colouring is particularly useful for tracking individual terraces or layers as they grow Figure 2-25. This mode of colouring is achieved by finding the 3 points that lie at the centre of each of the three facets (100), (110) and (111) Figure 2-23.
Figure 2-23 illustrating (in red) the three points that lie in the centre of each of the three facet types.

These points are then used in conjunction with the equations of plane for each of the facet types to establish layers, at the current time there are four colours used in the program so every four layers the colours repeat Figure 2-24. Each point in the data set is checked against these plane equations to elucidate which plane it belongs to and also as to which plane layer and then a colour is assigned accordingly.

Figure 2-24 illustrating how layers may be coloured on the (100) face with respect to the point at the centre of the face (highlighted in red).

This mode of colouring has some extra functionality, there are two menu entries associated with this mode under the colouring menu (layers M1 and layers M2). These are the same mode but with a shift by half a layer of structure; sometimes if the surface is populated with small cages such as D4R and they are coloured
differently to the layer below they cause bad aliasing to occur, these modes can be interchanged to alleviate this. This mode also has some keys on the keyboard associated with it, they allow the number of layers coloured on each of the three facet types handled by these colouring routines to be controlled. The keys ‘h’ and ‘b’ increase and decrease respectively the number of layers coloured on the (100) face, the keys ‘j’ and ‘n’ increase and decrease respectively the number of layers coloured on the (110) face and the keys ‘k’ and ‘m’ increase and decrease respectively the number of layers coloured on the (100) face.

![Figure 2-25 showing layer by layer colouring on a zeolite A crystal on a) (100) face b) (110) face and c) (111) face.](image)

Cage type colouring, being able to colour code each cage type on the crystal surface with a different colour allows very quick and efficient determination and comparison of what a crystal surface is composed of Figure 2-26. It can be enabled by selecting cages under the colouring menu.

![Figure 2-26 showing colouring by cage type on a zeolite A crystal on a) (100) face b) (110) face and c) (111) face.](image)

**Polylines**

Polylines can be added to the cages being drawn by selecting it under the view menu, for some of the cages at some orientations the simplicity of their drawing (for performance reasons) can make them a little difficult to distinguish, so to alleviate
Chapter 2 – Computational Methodology

this problem lines can be drawn along the edges of each face in a darker colour such that they become more distinct.

2.2.2.2 Performance

This newly developed program has gone a long way to facilitating simulations on a high-throughput scale. This is mainly due to the amount of time saved when post processing such a high volume of data, also the physical storage requirements have been drastically reduced when compared to storing Surface Evolver files. These improvements have largely come as a result of the following major improvements.

Reduction in input file size, since the input files for rendering only contain the raw output coordinates from the simulation the file size is much smaller than the files required for Surface Evolver. In fact the file size is 10% of that required to render the crystal in Surface Evolver. The reason this is so much faster is because in the old method of using Surface Evolver, every triangle used to render the crystal and its corresponding colouring and lighting information was stored in the file. This would be very slow to parse into memory, this information has to then be passed down the rendering pipeline to the graphics processor for rendering. The new approach has all of the geometric information for rendering stored inside a precompiled display list, then only the coordinates need to be loaded into memory and used to place a display list call at each coordinate. This approach has the advantage that it can take advantage of the ever increasing performance of modern graphics processing hardware.

Another major improvement came from the removal of the manual iterative step required to accurately and seamlessly colour the crystal. The old method involved performing a ‘guess’ as to what parameters would colour the crystal the best and then generating the Surface Evolver file, this would be followed by a manual rendering check and then repeating this until the desired results were achieved. This could typically take more than 10 minutes per iteration and could take 10 or more iterations to get the desired results. Often the problem seemed to the arise due to fact that some crystals can grow faster in some directions than others due to the complete randomness of the technique, resulting in a crystal that has its central point offset from the starting point. This was alleviated by moving the colouring into the visualiser where more information about the crystal can be obtained easily. During
load time of the crystal file, the offset is calculated and a correction factor is applied during the colouring routines resulting in seamless colouring first time automatically.

In this case it was much faster to perform these tasks at load time on the fly from the raw simulation file, rather than to generate the rendering information in a separate script and store them to file and then have to load this huge file into another program for rendering. This is due to the fact that modern processors and graphics processors can calculate the required information many times faster than the same information can be loaded from a file into memory. Some typical figures relating to these improvements are summarised in Table 2-2

<table>
<thead>
<tr>
<th></th>
<th>Surface evolver</th>
<th>Visualiser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical file size</td>
<td>180 Mb</td>
<td>16.8 Mb</td>
</tr>
<tr>
<td>Rendering a set of 1000 simulation results</td>
<td>160 hours</td>
<td>1 hour</td>
</tr>
</tbody>
</table>

Table 2-2 A tabulated summary of the improvements made between the old method of using Surface Evolver and the new visualiser program.

2.2.2.3 Summary of key bindings

Table 2-3 shows a summary of functions that are bound to the keyboard and mouse. These functions have been bound to keys that are familiar to users that frequently use keyboard and mouse shortcuts. It is anticipated that the key bindings will be surpassed by a fully capable GUI (Graphical User Interface) in future projects as is described in the future work section of this thesis.
Chapter 2 – Computational Methodology

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMB</td>
<td>Hold and move mouse for rotation</td>
</tr>
<tr>
<td>Z</td>
<td>Zoom in</td>
</tr>
<tr>
<td>X</td>
<td>Zoom out</td>
</tr>
<tr>
<td>W</td>
<td>Translate scene up</td>
</tr>
<tr>
<td>S</td>
<td>Translate scene down</td>
</tr>
<tr>
<td>A</td>
<td>Translate scene left</td>
</tr>
<tr>
<td>D</td>
<td>Translate scene right</td>
</tr>
<tr>
<td>H</td>
<td>Increase number of layers coloured on the (100) face</td>
</tr>
<tr>
<td>B</td>
<td>Decrease number of layers coloured on the (100) face</td>
</tr>
<tr>
<td>J</td>
<td>Increase number of layers coloured on the (110) face</td>
</tr>
<tr>
<td>N</td>
<td>Decrease number of layers coloured on the (110) face</td>
</tr>
<tr>
<td>K</td>
<td>Increase number of layers coloured on the (111) face</td>
</tr>
<tr>
<td>M</td>
<td>Decrease number of layers coloured on the (111) face</td>
</tr>
</tbody>
</table>

Table 2-3 A summary of each key binding and respective functionality.

2.3 References

Chapter 3

Results and Discussion (Zeolite L, Offretite & Erionite)
This section deals with the specifics of the study relating to the zeolites Zeolite L (LTL), Offretite (OFF) and Erionite (ERI), the chapter will be split into two main sections, one dealing with LTL and the other dealing with OFF and ERI. Each section will introduce the system, introduce how the problem was approached from a computational perspective then followed by the results and relevant discussions arising from them.

3.1 Aims

- To construct a program that can handle more complex connectivity than simple cubic frameworks.
- To study the crystal growth of zeolite L and potentially elucidate some of the mechanistic details as to how the morphological and topographical features come about.
- To apply the resulting program to other systems within the ABC-6 family of zeolites such as Erionite and Offretite and compare with experimental crystal growth observations.

3.2 Zeolite L

In this subsection some brief information about the LTL structure and its properties will be presented and will serve to put the results obtained into perspective. A discussion on how the problem was approached from a computational perspective will be presented in this subsection. Dealing with a non-cubic structure at the time was something new in this line of studies conducted at the CNM. Conclusions and future studies arising from this subsection will be included in chapter 6 under the relevant subcategories.

3.2.1 Introduction to Zeolite L

Zeolite L is not a naturally occurring zeolite, it was first synthesised by Breck and Acara in 1958.\(^1\) The crystal structure was not elucidated until 1969 by Barrer and Villiger.\(^2\) Zeolite L has a hexagonal unit cell with lattice parameters, a = 18.16 Å, b = 18.16 Å and c = 7.57 Å. It belongs to the P6/mmm space group.\(^3\)

The main commercial use of zeolite L is in the petroleum industry, it is used as a catalyst in the conversion of naphtha based compounds into aromatic compounds.
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)

One such process that utilises zeolite L is the AROMAX developed by Chevron Corporation.\textsuperscript{4}

In zeolite L Figure 3-1 the structural framework consists of stacks of cancrinite cages that are connected in rotational alignment through D6Rs. Six of these stacks are connected together via bridging bonds to form a larger central LTL cage. These large central cages are stacked in alignment and form the large pores that run perpendicular to the \{001\} face of the crystal. This large LTL cage has an opening ring size consisting of 12 T-sites with a diameter of 7.1 Å\textsuperscript{3}.

\begin{center}
\includegraphics[width=0.5\textwidth]{zeolite_L_structure.png}
\end{center}

\textit{Figure 3-1 the structural features of zeolite L as viewed perpendicular to the \{001\} direction.}\textsuperscript{3}

Synthetic zeolite L crystals exhibit hexagonal rod morphology, with individual crystals being in the size range of 2 - 5 µm Figure 3-2. The properties of the crystal such as their size, the aspect ratio, and also whether they are sharp hexagonal or more cylindrical can be altered by modifying synthetic conditions. These modifications could include the addition of more H\textsubscript{2}O, altering the Si/Al ratio or adding different cations/templates.\textsuperscript{4-6}
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)

It is such alterations of synthetic parameters and their respective observations upon the crystal habit that motivates this study to form a simple and clear model that aims to develop understanding of what processes may govern such changes.

3.2.2 Problem Breakdown

In this section details about the experimental approach will be discussed.

3.2.2.1 Coarse Graining

The logic for the coarse graining performed on the LTL system stem from an experimental study performed by Brent et al. at the University of Manchester. In this study Brent et al. found mild conditions in which dissolution was just about possible at a rate measurable by the AFM. By scanning across a single line of a single terrace repeatedly it was possible to track the dissolution. It was thus possible to record the terrace height as a function of time Figure 3-3. It can be observed that not all the step heights are equally as stable as they have different lifetimes under the same conditions.
This height data could then be correlated against the structure of LTL and the heights matched the most likely surface terminations. With the aid of some calculations performed by Jelfs et al. at UCL these terminations were verified. Figure 3-4 shows these structure terminations, the numbering used match the numbering used within Figure 3-3. It can be seen that the stable (the most long lived) surface terminations involve closed cage structures, which is in itself not a surprise since closed cages contain T-sites at the surface that are all Q\textsubscript{3} connected which makes them very stable species.

It was these observations that led to the current thinking about the connectivity used in the simulations throughout this study. In previous work Umemura had used both first and second sphere connectivity to define site types, however in this work the second sphere would be dropped since it is mainly used in studies of ionic crystals where second sphere neighbouring atoms are directly adjacent to the first sphere neighbouring atoms.
Using this logic of closed cages being the most structurally stable unit on the crystal surface, it is then possible to argue from the point of view of growth and dissolution that the formation of the closed-cage unit would be the rate-determining step. That is, in reality many addition and dissolution of simpler units may take place up until the formation of a closed cage that becomes very stable and long lived at the surface of the crystal. To compute these micro additions and dissolutions would be computationally very expensive both in the respect of time and also most likely memory.

In the study of LTL the coarse-grain building block was chosen to be the cancrinite cage, this means that connections are cleaved at the two bond linkages in the (a,b) directions and at the D6R in the c direction. This then leads on to a discussion of how the connectivity is dealt with computationally.

3.2.2.2 Connectivity
To tackle the LTL problem in terms of dealing with its more complex connectivity it was necessary to map its framework on to a Cartesian coordinate system. This requirement is different from the simpler cubic systems that have been done in past
studies. Figure 3-5 illustrates the thought process involved in going from the actual crystal structure to one that is represented within the array structures of the program.

Starting at a) it can be seen that the structure shown is a slice of the structure through the (a,b) plane each layer is then connected together via the cancrinite cages in the c direction via a D6R linkage. Image b) shows an overlay of the schematic representation of the structure and the structure, this schematic representation captures all the needed connectivity information such as the orientation of the cancrinite cages and their respective connections. Going from c) to d) it can be seen that the structure can be compressed in a hypothetical way such that it would easily map onto Cartesian coordinates whilst preserving all of the information regarding connectivity.
Figure 3-5 showing the process of breaking the structure down into a convenient way to handle inside arrays based on Cartesian coordinates. Going from a) to e) illustrates the steps going from the structure to a representation of the structure in computer memory.

It naturally follows that all of the site types using the principles of the Kossel model can be illustrated in a schematic way. Figure 3-6 shows a representation of all of the site types used in this study of the LTL system, in this model there are two different types of attachment. The first is via 2 bonds to a neighbouring cage in the (a,b) plane which will be termed a bidentate linkage, the second being via 6 bonds to neighbouring cages in the c axis. It follows that for a given cancrinite cage to be
crystal bulk it must be completely surrounded and linked to 5 other cancrinite cages, three via bidentate bonds and two via hexadentate bonds. It is then possible to permute all of the possible combinations of neighbours to arrive at all of the possible site types that can occur. The cancrinite cages are represented by triangles and each site type given a bracketed notation that refers to the number of bidentate connections (left) and hexadentate connections (right).

Figure 3-6 Showing all of the permutations of neighbour connectivity, this yields all of the site types in this model as described using the Kossel model. There are two different connection types in this model, bidentate (2 bond linkage) and hexadentate (6 bond linkage) connections these have been distinguished in the bracketed expression with the number of bidentate linkages on the left and the number of hexadentate linkages on the right. The cancrinite cages have been represented as 3D triangles for ease of visualisation.

It is then possible to consider the problem from an energetics perspective.
3.2.2.3 Energetics

This particular study used the energy of attachment model that has been used in previous studies in the group. The energetics are discussed in detail in chapter 2. Now that all of the site types and their given numbers of connections are known it is possible to then make an energy level diagram. If the number of bonds are considered in each type 2 (bidentate) and 6 (hexadentate) then it is not unreasonable to assume that the energy of making 6 bonds is 3 times greater than that to form 2 bonds. With this simple logic it is then possible to assign internal energies that act as multipliers to the number of each connection within a given site type, these internal energies will be denoted $U_B$ (bidentate) and $U_H$ (hexadentate).

If a simple energy relationship of 3:1 is assumed then the energy level diagram can be constructed. Figure 3-7 shows the energy level diagram for a relationship between $U_B$ and $U_H$ of 1:3 respectively.

![Energy Level Diagram](image)

*Figure 3-7 the energy level diagram for the zeolite L system as derived using the energy of attachment model. The internal energies of bidentate $U_B$ and hexadentate $U_H$ have been labelled. The arbitrary zero line has also been indicated. Indicated in brackets are the number of bidentate connections (left parameter) number of hexadentate connections (right parameter).*

The probabilities for growth and dissolution can thus be expressed in the following equations. The notation $L_{m,n}$ refers to the specific site type as defined by the number of bidentate linkages ($m$) and the number of hexadentate linkages ($n$). The parameters available to change during the simulations are the internal energies $U_B$ and $U_H$, these can be changed so that the spacing between each rung on the energy level diagram stays the same (i.e. they are scaled together) or they can be changed
independently (i.e. they are changed separately) and the effect upon the crystal morphology and topography be observed.

\[ P_{G}^{L_{m,n}} = \exp \left( \frac{0.5(mU_B + nU_H)E}{kT} + \frac{0.5\Delta\mu}{kT} \right) \]  \hspace{1cm} \text{Equation 3-1} \\

And

\[ P_{D}^{L_{m,n}} = \exp \left( \frac{-0.5(mU_B + nU_H)E}{kT} - \frac{0.5\Delta\mu}{kT} \right) \]  \hspace{1cm} \text{Equation 3-2} \\

3.2.3 Results and Discussion

The experiments performed during the course of this study fit into two major categories and will be discussed separately for clarity. The first of which involves changing the separation of the energy levels by scaling the attachment energies, thus the relative positions of each rung remain the same but change in terms of value. The second type of experiment will look at the morphology of zeolite L if the ratio of bidentate and hexadentate energies are changed slightly from the 1:3 that was proposed initial during the derivation of the model.

3.2.3.1 Fixed Energy Level Regime

Simulations can be performed allowing the crystal to grow in full 3D allowing the morphology and topography to be probed, or they can be performed in 2D of a given surface/crystallographic direction to investigate what a given surface may look like. The advantage of the 2D simulation is that a much larger surface/face can be generated due to lower memory requirements, although the crystal habit is unavailable in this mode.

In these experiments the parameters \( U_B \) and \( U_H \) are set to 1 kcal mol\(^{-1} \) and 3 kcal mol\(^{-1} \) respectively. The parameter \( E \) in equations 3-1 and 3-2 can be changed to investigate the effect upon the resulting simulated crystal morphology and topology. The effect of changing \( E \) has the effect upon the energy level schematic of changing the separation of the energy levels as shown in Figure 3-8.
A number of 3D simulations were performed varying the energy scaling parameter $E$ between 1-7 kcal mol$^{-1}$. Figure 3-9 and Table 3-1 show the resulting output from these simulations and the resulting aspect ratio for each energy $E$. The images shown in Figure 3-9 are the final frame taken for each experiment during the equilibrium phase of the experiment.

When the energy scalar is small $E = 1$ kcal mol$^{-1}$ the resulting crystal is almost spherical, from the side it has a slight elongation in the c direction, this is because growth in the c direction is not much more energetically preferred than that in the a/b directions. Conversely at the other extreme, when $E = 7$ kcal mol$^{-1}$ growth in the c direction is very much preferential to that in the a/b direction, this preference is what causes the aspect ratio to become so quickly enlarged at energies above 4 kcal mol$^{-1}$. 

*Figure 3-8 showing two separate energy schemes with different E parameters. Blue shows $E = 1$ kcal mol$^{-1}$. Red shows $E = 3$ kcal mol$^{-1}$.\*
Figure 3-9 showing results from a set of 3D simulations in which the parameter $E$ was varied. a) $E = 1.0$ kcal mol$^{-1}$ b) $E = 1.5$ kcal mol$^{-1}$ c) $E = 2.0$ kcal mol$^{-1}$ d) $E = 2.5$ kcal mol$^{-1}$ e) $E = 3.0$ kcal mol$^{-1}$ f) $E = 3.5$ kcal mol$^{-1}$ g) $E = 4.0$ kcal mol$^{-1}$ h) $E = 4.5$ kcal mol$^{-1}$ i) $E = 7.0$ kcal mol$^{-1}$.
This preference comes about from the energy level schematic, any site type involving the bidentate connection has a $U_B$ of 1 kcal mol$^{-1}$ before being multiplied by $E$, whilst any site involving hexadentate connections has a $U_H$ of 3 kcal mol$^{-1}$ before being multiplied by $E$. Due to the fact that the probability for each site type is derived from an exponential expression, the probabilities involving hexadentate connections are going to become much larger than those involving bidentate connections when the parameter $E$ is scaled up.

<table>
<thead>
<tr>
<th>Energy $E$ (kcal mol$^{-1}$)</th>
<th>Aspect Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1:1.6</td>
</tr>
<tr>
<td>1.5</td>
<td>1:2.2</td>
</tr>
<tr>
<td>2</td>
<td>1:3.1</td>
</tr>
<tr>
<td>2.5</td>
<td>1:4.5</td>
</tr>
<tr>
<td>3</td>
<td>1:6.4</td>
</tr>
<tr>
<td>3.5</td>
<td>1:8.4</td>
</tr>
<tr>
<td>4</td>
<td>1:10.8</td>
</tr>
<tr>
<td>7</td>
<td>1:65</td>
</tr>
</tbody>
</table>

*Table 3-1 showing the aspect ratio of each simulation vs energy scaling parameter $E$.

The 3D dimensional simulations show the overall crystal morphology quite successfully, however the topographical detail is difficult to discern due to the fact that the crystal faces are fairly small, this is a constraint that is rooted in the amount of memory available and also the way the crystal is stored. To get a better view of the topographical detail 2D simulations can be performed on a given face of the crystal.

A similar range of energies were used in the 2D simulations as those used in the 3D simulations for comparison with experimental results. The results can be plotted using a simple listplot in Wolfram Mathematica, a selection of the resulting images from this set of experiments is shown in Figure 3-10.
Figure 3-10 showing 2D simulations of the \{100\} face, the upper image is taken at high driving force and the lower image is taken at equilibrium driving force a) $E = 1$ kcal mol$^{-1}$ b) $E = 2$ kcal mol$^{-1}$ c) $E = 3$ kcal mol$^{-1}$ d) $E = 4$ kcal mol$^{-1}$ e) $E = 7$ kcal mol$^{-1}$. Each panel represents 0.39 μm by 0.49 μm.

Similar to the 3D experiments, when the energy scalar $E$ is small there is little preference for growth in either the a-b/c directions. However as $E$ is increased the preference of growth direction tends towards the c direction ever increasingly as was the case in the 3D experiments.

When comparing the 3D simulations to the experimental data on zeolite L Figure 3-11 good correlation with the surface details occur in the range $E = 3-4$ kcal mol$^{-1}$ with the best surface detail correlations towards the 4 kcal mol$^{-1}$ end of the range. However the aspect ratio of the 3D crystals deviates from those observed experimentally at 4 kcal mol$^{-1}$ with experimental data showing an average aspect ratio of approximately 4, however the aspect ratio for the simulation at 4 kcal mol$^{-1}$ comes in at around 10.
This difference in comparison between the 3D and 2D simulations against experimental data could be for a number of reasons. The simulations are performed based on the assumption that crystal growth is “perfect” that the crystal is defect free. In reality zeolite L crystals exhibit large numbers of defects running along the c direction of the crystal framework, these defects are more pronounced in shorter crystals possibly owing to the fact that defects are stunting the growth of crystals in the c direction. However terraces at the surface of the crystal that appear defect free in the c direction will probably be unaffected by such stunting. Figure 3-12 shows how the size varies widely within a single sample, this probably owes to the fact that different crystals grew under varied conditions and also possess vastly different densities of defects. This could possibly also compound the determination of the average aspect ratio in the sample if a large enough sample is not sampled.
Figure 3-12 an SEM image performed by Brent et al., as can be seen the size of crystals in the same sample widely vary.

Another possibility is that the simulations were performed in an incorrect driving force regime, before this is investigated further it is essential to understand how the driving force affects growing crystals within the simulation.

Figure 3-13 shows the driving force profiles of the 2D experiments shown in Figure 3-10, the actual energy levels all reside at energies below zero on the y axis. The experiments performed started off with a very high driving force at a value very much higher than the top rung on the energy level diagram. The driving force was then allowed to equilibrate to its natural position on the energy level diagram, as can be seen this value varies with different $E$ values. This is due to the fact that the equilibrium driving force always settles at approximately the energy level $L_{(2,1)}$ independently of what value $E$ takes. The site represented by $L_{(2,1)}$ is a type of kink site in this model.

Another interesting property of the driving force is that of Ostwald ripening, this is the process in which smaller objects or features preferentially dissolve providing nutrients for larger features. This effect can be seen in both Figure 3-13 and Figure 3-14, in the first instance it is observable as an elongated hump on the driving force curve between 100,000 and 200,000 iterations, the panels f-j in Figure 3-14 coincide with this range and the rapid disappearance of small features and expansion of larger features can be observed in this subsequence of frames. This process occurs around equilibrium where sites with energies that are above the driving force are prone to dissolution whilst sites below the driving force are still in growth mode.
Figure 3-13 showing the driving force profiles of the 2D experiments shown in Figure 3-10.
Figure 3.14 shows a successive sequence of frames going from a-p, each frame is 0.29 μm by 0.49 μm and represents 20,000 iterations where $E = 4 \text{ kcal mol}^{-1}$. 
The value that the driving force starts at has such a large effect upon the final outcome of the simulation. As a result the possibility that the driving force being too high at the start of the simulation should be investigated. It is also possible that the driving force in reality is never too far above equilibrium (this will be discussed further in chapter 5). Since it is very difficult to measure what real supersaturation values are, and the difficulty in then correlating those values with the driving force in this model, it is important to be able to use other metrics for comparison.

One such metric is that experimental data has shown that lateral spread is quite difficult and generally only occurs via a bridging mechanism where lateral spread of a given terrace only occurs via the bridging to two adjacent cancrinite columns Figure 3-15 c). From an energetic point of view the model being presented here does not differentiate energetically between a single cancrinite cage at the surface or a single cancrinite cage that is forming a bridge shown in a) as they would both consist of a single bidentate connection this is denoted by energy level L<sub>(1,0)</sub> on the energy level diagram.

It can be seen that energetically the bridging cannot be deterred in this model. However, going back to the fact that the driving force may well have been set too high in previous simulations; leads to the possibility that the starting driving force is much lower and possibly even not higher than the highest rung on the energy level diagram. This can be rationalised by the long induction times commonly observed in zeolite syntheses, which would imply site types involved in nucleation are not in all-out growth mode. A series of experiments can be performed to investigate the effects of decreased driving force upon the crystal habit.

![Figure 3-15 showing the bridging mechanism, that lateral growth occurs by bridging two adjacent cancrinite columns.](image)
Figure 3-16 showing how the varied driving force ($\Delta \mu$) experiments compare to the energy levels expt 1) $\Delta \mu = 0$ kcal mol$^{-1}$ expt 2) $\Delta \mu = -1.8$ kcal mol$^{-1}$ expt 3) $\Delta \mu = -2.7$ kcal mol$^{-1}$ expt 4) $\Delta \mu = -3.6$ kcal mol$^{-1}$ expt 5) $\Delta \mu = -7.2$ kcal mol$^{-1}$ expt 6) $\Delta \mu = -10.8$ kcal mol$^{-1}$ expt 7) $\Delta \mu = -14.4$ kcal mol$^{-1}$. 

Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)
By performing experiments for an $E = 4$ kcal mol$^{-1}$ crystal in which the driving force was successively lowered starting at 0 kcal mol$^{-1}$ Figure 3-16 it was possible to observe the effects upon the growth habit of the crystal Figure 3-17. The random number generator used in these experiments was seeded with the same number for each experiment such that the same sequence of random numbers was played out, this allows one to compare only the effect of the changing driving force had the move choices remained the same.

The first five boards in Figure 3-17 (a-e) show identical results. This means that because the move choices are strictly limited by the same random number sequence that changing the driving force thus far has had little to no effect on either the crystal habit, the density of nucleation or the size/aspect ratio of the crystals. This is a surprising observation since the frames represented by Figure 3-17 e) correspond to the driving force line marked as experiment 5 on Figure 3-16. This line is clearly well below the energy level $L_{(1,0)}$ the bidentate nucleation site for the {100} face, however nucleation appears unsuppressed.

However reducing the driving force still further to below $L_{(2,0)}$ represented by frames f) and experiment 6 it can be seen that this reduction has reduced the nucleation density slightly, but the overall habit remains relatively the same. Coming down further to the driving force represented by experiment 7, the results in frame g), we can see that the nucleation density has been modified significantly and also the crystal habit. Comparing the length of the terraces to those in previous experiments where $E = 4$ kcal mol$^{-1}$ for example those in Figure 3-14 it can be seen that the terraces are a lot longer in the c-direction and there are more single columns present, terraces that have spread laterally exhibit shoulder like features that are more in line with AFM observations such as the one shown in Figure 3-11 b).

The reason the terraces spread laterally in this manner under lowered driving force can be interpreted from the energy level diagram in Figure 3-16. For experiment 7 the driving force quite a long way below the $L_{(1,0)}$ energy level needed for the lateral growth to occur without bridging two other cancrinite columns. This means that as soon as one of these $L_{(1,0)}$ sites occur it is rapidly dissolved, this is especially compounded if there are no neighbouring sites available to attach $L_{(1,1)}$ sites in the c direction to stabilize the bridge.
Figure 3-17 showing the results of the driving force experiment, the top image shows the first frame at 2,000 iterations, the lower image shows the final frame at 40,000 iterations. Each frame represents an area of 0.29 µm by 0.49 µm a) \( \Delta \mu = 0 \) kcal mol\(^{-1} \) b) \( \Delta \mu = -1.8 \) kcal mol\(^{-1} \) c) \( \Delta \mu = -2.7 \) kcal mol\(^{-1} \) d) \( \Delta \mu = -3.6 \) kcal mol\(^{-1} \) e) \( \Delta \mu = -7.2 \) kcal mol\(^{-1} \) f) \( \Delta \mu = -10.8 \) kcal mol\(^{-1} \) g) \( \Delta \mu = -14.4 \) kcal mol\(^{-1} \).
This rapid dissolution of $L_{(1,0)}$ is evidenced by the reduction in nucleation as the site type for nucleation on the {100} face is exactly the same as that required to spread laterally without bridging Figure 3-15 a). This lends evidence to the fact that lateral spreading must be due to the bridging of two adjacent cancrinite columns, although the bridge site $L_{(2,0)}$ is also in dissolution mode (above the driving force) it is to a much lesser extent than $L_{(1,0)}$. Also once the bridge has formed the sites created either side of the bridge are of the type $L_{(2,1)}$ which are very much still in growth mode, these sites will quickly stabilise the bridge and successive $L_{(2,1)}$ sites will be created in the c direction, these will grow relatively rapidly catching up with the two initial columns that were bridged, this is what owes to the shoulder like features observed via AFM. The events described here can be shown by rendering the result with the visualiser program described in chapter 2, this allows the bridging event to be captured shown centre of Figure 3-18 a). Image b) shows a view down one of the large pores running the length of the crystal in the c direction, showing the bridging layer catching up with the edge of the pillaring columns.

These experiments lend credence to the validity of the model, and also that the energy scalar $E$ is within the range of 4 kcal mol$^{-1}$. When compared with experimental literature values in which the energy of formation of a T-O-T bond is obtained by NMR (-17.6 kJ mol$^{-1}$) the value obtained during this study is extremely close to the reported range.$^{10}$

Another scenario in which studying the effects of applied driving force is important is in that of dissolution. Whilst studying the dissolution of zeolite L Brent et al. discovered a mysterious effect of “lighting up” of areas undergoing dissolution. This
effect comes about in the lateral force scanning mode of the AFM where the friction of the surface has an effect upon the tip. Areas undergoing dissolution report a different friction to areas not dissolving, this effect was investigated by Brent et al. who reports that terraces tend to dissolve from the tips inwards along the c direction Figure 3-19 a) – e).\(^8\)

An experiment was designed that could closely mimic those that are performed in the lab, where a crystal is first grown and then exposed to a solution that causes it to dissolve and then monitored using the AFM. In the simulation the crystal was first grown using a high supersaturation stage and then allowed to equilibrate, much like synthesising a crystal in the lab and then extracting it via filtration. Then the simulated crystal was exposed to harsh dissolution conditions where the driving force was decreased below the equilibrium driving force by a significant amount Figure 3-20.

\begin{figure}
\begin{center}
\includegraphics[width=\textwidth]{AFM_images}
\end{center}
\caption{Figure 3-19 showing a sequence of AFM images showing the “lighting up” of dissolving terraces (a-e) and a set of simulations in which a crystal is firstly grown followed by a dissolution step (f-j) where dissolved sites are marked in red. Simulations are 0.29 \(\mu\)m by 0.49 \(\mu\)m and each of the panels f) – g) is 100,000 iterations apart and j) is 5000 iterations after i).}
\end{figure}

The images shown in Figure 3-19 f) – j) show the results of such an experiment, the frame f) shows the end of the high supersaturation stage, the frame g) – i) show the
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite

The equilibration stage of the experiment and frame j) shows the dissolution stage of the experiment. As can be seen in this frame dissolution occurs primarily at the tips of terraces and works inwards much in the same way as those observed in reality. This is because the sites at the ends of the terraces are of the lowest stability (higher on the energy level diagram) and thus have the highest dissolution rate compared to other site types.

![Driving force profile for dissolution experiment](image)

*Figure 3-20 showing the driving force profile for growing a crystal sample surface and then subjecting it to dissolution.*

The excellent agreement between the experimental and theoretical observations presented so far lends itself to showing that the model presented and discussed thus far is of sound logical and scientific grounding.

3.2.3.2 Varied Energy Level Regime

So far the model has been discussed when operated in the fixed energy level regime, by which the internal energy parameters $U_B$ and $U_H$ are held in the fixed relationship of 1:3 kcal mol$^{-1}$ respectively. However it has been shown that simply changing the separation of the energy levels by scaling $E$ does not yield much morphological variety. The only other property in the fixed energy level mode that could be experimented with was the driving force, and whilst this yielded good results for the
cross section of experimental data it was correlated with, there is still more to understand.

Studies have shown that changing the nature of the synthesis causes big changes in both the aspect ratio and also the shapes of the crystals formed. Different conditions can affect the crystal growth differently, for example, changing the type of cation, changing the amount of cation, changing the amount of Si/Al or both, changing the amount of water or introducing a templating or structure directing agent into the synthesis all have an effect on the resulting crystals.\textsuperscript{11, 12} For example Lee et al. showed that by changing the molar concentration of water in the synthesis the crystal morphology was altered Figure 3-21.\textsuperscript{6}

![Figure 3-21 showing SEM images that show the change in crystal habit of zeolite L when the concentration of water was increased. From a gel composition of 10.9K2O : 1Al2O3 : 20SiO2 : xH2O where a) x = 400, b) x = 600, c) x = 800, d) x = 1000, e) x = 1200 and f) x = 1400.\textsuperscript{6}](image)

The fixed energy regime was unable to produce the effect shown by Lee et al. however one can imagine that if certain structural units were stabilised or site types running in certain directions were poisoned then the rates and thus the energies would be altered. This is the premise for the following set of experiments; the scaling parameter under the varied energy regime is not changed and thus simply set to 1 kcal mol\textsuperscript{-1} such that it can be neglected. The internal energies associated with the bidentate ($U_B$) and hexadentate ($U_H$) can be varied separately. This will completely change the energy landscape that we have seen thus far in the previous study. Figure 3-22 shows the full array of resultant crystal habits that permuting these two parameters through a range of 1-12 kcal mol\textsuperscript{-1}. Experiments from the fixed regime
with $E = 1,2,3$ and 4 kcal mol$^{-1}$ have been marked with red borders to put into perspective how little morphological range the fixed energy regime has.
Figure 3-22 showing an array of resultant morphologies when the internal energies $U_B$ (top) and $U_H$ (left) are given various energy values. The experiments from the previous section on the fixed energy level regime with $E = 1, 2, 3$ and 4 kcal mol$^{-1}$ are marked with red borders.
As can be seen from Figure 3-22 a vast array of morphologies can be produced compared to what has been seen from this model up until this point. These can be better represented schematically where the zones in which the variety of morphologies can be found in reference to energy coordinates are marked Figure 3-23.

In the top left of the schematic are spherical objects, this as at energies where the difference between each energy level is small (almost degenerate) this is what leads to the crystal in these cases being spherical, it is unlikely that this would happen in reality. Towards the center-top, the crystals are disc shaped. On the top right hand side hexagonal shaped plates are observed. Short and long cylinders (rounded rods) form around the central region of phase space. Whilst hexagonal rods are seen in the bottom left of phase space.
Figure 3-24 showing the energy levels of three different morphologies (purple) hexagonal plates $U_B = 2 \text{ kcal mol}^{-1}$ $U_H = 12 \text{ kcal mol}^{-1}$ (red) long cylinder $U_B = 6 \text{ kcal mol}^{-1}$ $U_H = 6 \text{ kcal mol}^{-1}$ and (blue) hexagonal rods $U_B = 12 \text{ kcal mol}^{-1}$ $U_H = 4 \text{ kcal mol}^{-1}$.

To understand why the morphologies occur in those specific zones it is important to consider the energetics involved. Figure 3-24 shows how the energy levels are changing for three of the morphologies. By understanding why these three morphologies come about it is possible to understand the rest of the zones. Probably the most intriguing property about the schematic of morphologies Figure 3-23 is the fact that going in a straight line from bottom left to top right the morphology changes from long hexagonal rods to rounded rods then back to hexagonal plates. Looking at the energy levels it can be seen that the rungs are grouped into five vertical groups that classify them via the total number of connections. It is interesting to see that the further apart the rungs in each particular group the more hexagonal the nature of the
crystal. The difference in length in the c direction, that creates rods versus plates is to do with the stabilisation of growth of the hexadentate connections, the more stable (higher $U_H$) the longer the crystal, this is evident when looking at the positions of the energy levels with hexadentate contributions between the rod and plate like morphologies. The rounded cylinders can be explained by considering what happens between the two cases, as $U_B$ and $U_H$ approach each other a number of the energy levels collapse to just 5 groups of degenerate energy levels, the one shown in Figure 3-24 in red is the extreme where $U_B$ and $U_H$ are equal, but it also occurs at energies approaching this equality until such a point as there is enough differentiation to produce hexagonal rods or plates again. Figure 3-25 shows an enlarged selection of morphologies available from this model.

![Figure 3-25 showing some larger images of the main morphology groups a) spherical crystal b) circular disc c) hexagonal plate d) short cylinder e) long cylinder f) hexagonal rod](image)

The overall goal of creating such a model is that in future studies it should be possible to utilise as a predictive tool to guide synthetic studies/production. For this to be possible the results obtained from the simulations have to be correlated with known chemical/physical parameters. A number of studies have been conducted by Brent et al. in which various parameters within the synthesis of zeolite L were carefully altered and the effect upon morphology observed using SEM.

Brent et al. first varied the molar ratio of water in the following preparation:

$$10.2K_2O : 1Al_2O_3 : 20SiO_2 : xH_2O$$

where $x = 200, 300, 400, 500, 600, 700, 800, 1030, 1200, 1300$
Figure 3-26 Showing SEM micrographs of the hexagonal face \{001\} of zeolite L where the moles of water used are a) 500 b) 600 c) 700 d) 800 e) 1030 f) 1200 performed by Brent et al.®

Figure 3-26 and Figure 3-27 show the SEM images obtained for the \{001\} (hexagonal) face and the \{100\} side wall face respectively, the sequences of SEM shows how the morphology of zeolite L changes as the water concentration is increased in the synthesis.

Figure 3-27 Showing SEM micrographs of the side face \{100\} of zeolite L where the moles of water used are a) 500 b) 600 c) 700 d) 800 e) 1030 f) 1200 performed by Brent et al.®
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite

At high molar concentrations of water the crystals are of higher aspect ratio and exhibit very defined hexagonal rod like morphology. At lower molar concentrations crystals suffer frustrated growth in the c direction thus reducing the aspect ratio significantly. Studies have shown that preferential growth in the c direction in silica rich systems, Table 3-1 shows how the Si/Al ratio increases in syntheses at high molar water concentrations, it is suggested that the higher water content allows the usually difficult silica species to more readily dissolve, possibly by altering the viscosity/pH, thus leading to greater incorporation of silica units onto the crystal.\textsuperscript{13} It has been shown that Si rich systems result in preferred growth in the c direction in zeolite L.

<table>
<thead>
<tr>
<th>Moles of water</th>
<th>Average Length (µm)</th>
<th>Average Diameter (µm)</th>
<th>Aspect ratio (average)</th>
<th>Si/Al</th>
<th>K/Si</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>0.5</td>
<td>1.6</td>
<td>0.3</td>
<td>3.20</td>
<td>0.34</td>
</tr>
<tr>
<td>700</td>
<td>0.7</td>
<td>1.4</td>
<td>0.5</td>
<td>3.27</td>
<td>0.35</td>
</tr>
<tr>
<td>800</td>
<td>2.8</td>
<td>1.9</td>
<td>1.5</td>
<td>3.54</td>
<td>0.36</td>
</tr>
<tr>
<td>1030</td>
<td>3.8</td>
<td>1.5</td>
<td>2.5</td>
<td>3.47</td>
<td>0.35</td>
</tr>
<tr>
<td>1200</td>
<td>7.2</td>
<td>1.5</td>
<td>5.1</td>
<td>3.87</td>
<td>0.33</td>
</tr>
</tbody>
</table>

\textit{Table 3-2 showing the dimension measurements and chemical composition versus water concentration performed by Brent et al.}\textsuperscript{7}

This is an example of how changing a single chemical parameter can alter the nature of the growth of a crystal. In our model a greater extent of growth in the c direction is directly associated with changing the stability of the hexadentate connection. Figure 3-28 shows how the changes in morphology observed map onto the morphology map obtained as a result of performing a mass screen of the attachment energy phase space.
Bidentate energy $U_B$ kcal mol$^{-1}$

Figure 3-28 showing how altering the water content of a synthesis might relate to the attachment energy phase space in the simulations.

Brent et al. secondly varied the molar ratio of potassium hydroxide in the same preparation:

$$xK_2O : 1Al_2O_3 : 20SiO_2 : 1030H_2O$$

where $x = 9.0, 9.5, 10.0, 10.5, 11.0$
Figure 3-29 Showing SEM of the hexagonal face {001} of zeolite L were the potassium oxide concentrations were a) 9.0 b) 9.5 c) 10.0 d) 10.5 e) 11.0. Measurements performed by Brent et al.\(^7\)

Figure 3-29 and Figure 3-30 show the SEM images of the hexagonal \{001\} and side wall \{100\} faces respectively, each sequence of images show how the morphology is affected by changing the molar concentration of potassium oxide in the synthesis. Increasing the molar concentration of potassium oxide causes shortening of the crystals in the \(c\) direction, it is thought that the potassium ions poison the growth sites in the \(c\) direction,\(^7\) thus the greater amount of potassium in the synthesis the greater the extent of the poisoning.
Figure 3-30 Showing SEM of the side face {100} of zeolite L were the potassium oxide concentrations were a) 9.0 b) 9.5 c) 10.0 d) 10.5 e) 11.0. Measurements performed by Brent et al.\textsuperscript{7}

Table 3-3 shows how the Si/Al and also the K/Si change as a function of molar concentration of potassium oxide. This could possibly indicate that an increased amount of potassium is taken up as a result of a lower Si/Al ratio, meaning it is used for charge balancing Al in the framework. It has already been seen that Si is preferentially incorporated along the c direction.

<table>
<thead>
<tr>
<th>Moles of K\textsubscript{2}/O</th>
<th>Average Length (µm)</th>
<th>Average Diameter (µm)</th>
<th>Aspect ratio (average)</th>
<th>Si/Al</th>
<th>K/Si</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0</td>
<td>4.14</td>
<td>1.42</td>
<td>2.80</td>
<td>3.83</td>
<td>0.30</td>
</tr>
<tr>
<td>9.5</td>
<td>4.07</td>
<td>1.72</td>
<td>2.40</td>
<td>3.63</td>
<td>0.33</td>
</tr>
<tr>
<td>10.0</td>
<td>2.00</td>
<td>1.85</td>
<td>1.30</td>
<td>3.35</td>
<td>0.35</td>
</tr>
<tr>
<td>10.5</td>
<td>1.19</td>
<td>1.96</td>
<td>0.61</td>
<td>3.49</td>
<td>0.38</td>
</tr>
<tr>
<td>11.0</td>
<td>0.61</td>
<td>1.58</td>
<td>0.39</td>
<td>3.27</td>
<td>0.37</td>
</tr>
</tbody>
</table>

\textit{Table 3-3 showing the dimension measurements and chemical composition versus potassium oxide concentration performed by Brent et al.\textsuperscript{7}}
Figure 3-31 displays how changing the molar ratio of potassium oxide in the synthesis maps onto the morphology map obtained from mass simulations across the attachment energy phase space. This is another example of how chemical considerations can be mapped against the simulation results, this particular case was an example of how mapping the effects of reducing the growth rate in a particular direction compares between the morphological effects of simulation and real experiments.

Brent et al. also performed experiments using organic templating agents, they predicted and showed that the crown ether 21-crown-7 was the best performing crown ether at templating zeolite L. Brent et al. showed via experimental evidence and in collaboration with Lobo et al. at UCL in the form of theoretical calculations that the crown ether sat within the LTL cage.
Figure 3-32 showing SEM images of zeolite L crystals of progressively increasing molar amounts of organic template, the left image shows the hexagonal face {001} the right image shows the side wall face {100}, as performed by Brent et al.

Figure 3-32 and Table 3-4 show the morphological and chemical properties respectively of the zeolite L crystals resulting from increasing the molar amount of the organic template 21-crown-7. As the amount of the crown ether is increased the aspect ratio is shortened. However, unlike the previous two cases the Si/Al ratio and the K/Si ratio have remained unchanged. This indicates that the template is not reducing the aspect ratio via the same mechanism as the previous two cases. In addition to decreasing the length of the crystal in the c direction, it causes an increase in the diameter of the crystals. Brent et al. discovered that the crown ether is incorporated into the crystal via study using NMR and was proposed to sit inside the LTL cage. It is proposed that the crown ether is stabilising growth in the a/b directions and simultaneously inhibiting growth in the c direction, experimental observations appear to add weight to this theory. The exact mechanism as to how this comes about it still unknown.
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)

<table>
<thead>
<tr>
<th>Moles of 21-crown-7</th>
<th>Crystal length (μm)</th>
<th>Crystal diameter (μm)</th>
<th>Aspect ratio average</th>
<th>Si/Al</th>
<th>K/Si</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.8</td>
<td>1.5</td>
<td>2.5</td>
<td>3.47</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.6</td>
<td>1.3</td>
<td>3.50</td>
<td>0.35</td>
</tr>
<tr>
<td>2</td>
<td>1.8</td>
<td>1.5</td>
<td>1.2</td>
<td>3.49</td>
<td>0.34</td>
</tr>
<tr>
<td>4</td>
<td>1.8</td>
<td>2.3</td>
<td>0.8</td>
<td>3.50</td>
<td>0.33</td>
</tr>
</tbody>
</table>

*Table 3-4 showing the dimensional and chemical properties vs molar concentration of crown ether as performed by Brent et al.*

Mapping the findings made by Brent et al. onto the morphological map obtained via mass simulating the energetic phase space for the attachment energies, it is possible to show how changing the molar amount of crown ether compares to results from simulations Figure 3-33. This is an example how the observations from experiments can be compared with those from theoretical approaches. If more parameters and their influence upon crystal growth habit are well understood, then it is not unreasonable to make the claim that such models can be used to aid synthetic zeolite chemists in their quest to design crystals.

It is possible to simulate the outcomes of stabilising/destabilising various types of attachments and their effects upon crystal morphology. Thus for an experienced synthetic zeolite chemist it should narrow down or point towards the kinds of conditions needed to produce the desired effects. This could potentially serve to reduce the serendipity of discovering new morphologies. Zeolite chemists have long been trying to create hexagonal plate morphologies, the benefits of obtaining such crystals for catalysis would be great since the pores running in the c direction would be of very short length, this owes to crystals of higher performance in terms of product being able to readily diffuse out of the short pores. Other industries have shown interest in the hexagonal plate like morphology. One of which is the electronic display industry, where having flat plate like crystals with regular pores in which molecules with optical properties can be held are of high interest. To date, crystals with hexagonal plate like morphology have not been synthesised.

It would be of significant interest to see this model could predict the conditions to obtain such crystals, theoretically they appear to be producible by increasing the
concentration of crown ether even higher than that used by Brent et al., however this model was not available to test this at the time of performing those experiments. It also appears is a combination of the crown ether and changing the water/K$_2$O could be used to access the hexagonal crystals towards the top right corner of the morphology map.

![Figure 3-33 showing how changing the molar amount of crown ether maps onto the morphological map generated via simulation.](image)

### 3.3 Offretite

In this subsection some brief information about the OFF structure and its properties will be presented and will serve to put the results obtained into perspective. A discussion on how the problem was approached from a computational perspective will be presented in this subsection. Conclusions and future studies arising from this subsection will be included in chapter 6 under the relevant subcategories.
3.3.1 Introduction to Offretite

Natural offretite was first discovered near Mont Semionse, Loire, France in 1890 by Gonnard, and was named in honor of Prof. A. J. J Offrét of Lyon University. Its structure was originally differentiated from that of erionite by Hey and Fejer in 1962. Offretite exhibits a hexagonal unit cell of lattice parameters $a = 13.30 \, \text{Å}$, $b = 13.30 \, \text{Å}$, $c = 7.59 \, \text{Å}$ and belongs to the $P\bar{6}m2$ space group.\(^3\)

There is no major commercial use of offretite, however its use has been explored as a de-waxing catalyst in the petroleum industry. In this de-waxing process long straight chain alkanes can enter the pores in a preferential manner and thus cracked into smaller more useful compounds.\(^{14}\) It has also been investigated for use in naphthalene cracking applications.\(^{15}\) Potassium exchanged forms have been investigated for use in the cracking of iso-octane, they showed selectivity towards producing hydrocarbons of $C_3$ chain length.\(^{16}\)

![Figure 3-34: The structural features of offretite viewed perpendicular to the [001] direction.\(^3\)](image)

The structural framework of offretite Figure 3-34 consists of rotationally aligned stacks of cancrinite cages that are connected together through D6Rs. These stacks are then connected together through stacks of gmelinite cages around a central pore. This central pore runs along the c-direction of the framework and consists of 12 – membered rings, the pore diameter is 6.8 Å. A second pore type running perpendicular to the main pore is a 2-dimensional interlinked pores system consisting of 8 membered rings, the pore diameter is 4.9 Å.\(^3\)
Naturally occurring offretite Figure 3-35 is not widely observed to be pure, with the main sources coming from France and Italy. Naturally occurring crystals exhibit hexagonal rod like morphology and usually express the hexagonal \{001\} faces. Although large single crystals in nature are rare, smaller crystals are often found clustered together in parallel bundles.\textsuperscript{17}

Synthetic offretite Figure 3-36 is relatively easy to synthesise in the laboratory with a high degree of purity. It is always produced using organic templating agents such as choline chloride or tetramethylammonium, the general habit of synthetic offretite consists of either parallel bundles of hexagonal needles or single crystals of hexagonal rods.\textsuperscript{18}
3.3.2 Problem Breakdown

In this section details about the experimental approach will be discussed.

3.3.2.1 Connectivity

To tackle the offretite system from a computational point of view, similar ideas were drawn upon as those used to tackle the LTL system. Utilizing the idea of closed cages, the offretite structure can be constructed entirely through linking cancrinite cages together much like with the LTL system.

Using the logic of interconnecting cancrinite cages from the LTL model, it is possible to come up with a schematic representation of the offretite structure in exactly the same way as before. Figure 3-37 shows the schematic representation of the offretite system when the cancrinite cage is considered as the building unit, each cancrinite cage links to six others in the (a,b) plane via single bridging linkers, whilst linking to two in the c direction via 6 (hexadentate) bonds.

Figure 3-36 showing synthetic offretite crystals exhibiting the hexagonal rod morphology.\textsuperscript{19}
Using similar logic to the derivation of site types as that used in the LTL system it is possible to identify site types based on their numbers of mono and hexa dentate connections, a similar bracketed notation will be used here to discriminate between site types bases on the numbers and types of connections. It is now possible to consider the energetics of this system.

3.3.2.2 Energetics
This study uses the same energetics model as that of the LTL system, now that all of the site types and their given numbers of connections are known it is possible to then make an energy level diagram. If the number of bonds are considered in each type 1 (monodentate) and 6 (hexadentate) then it is not unreasonable to assume that the energy of making 6 bonds is 6 times greater than that to form 1 bond. With this simple logic it is then possible to assign internal energies that act as multipliers to the
number of each connection within a given site type, these internal energies will be denoted $U_M$ (monodentate) and $U_H$ (hexadentate).

If a simple energy relationship of 6:1 is assumed then the energy level diagram can be constructed. Figure 3-38 shows the energy level diagram for a relationship between $U_M$ and $U_H$ of 1:6 respectively.

![Energy Level Diagram](image)

*Figure 3-38 showing the energy level schematic as based upon the attachment energy model. The internal energies of attachment of a single monodentate attachment $U_M$ and hexadentate attachment $U_H$ have been indicated. Also the arbitrary zero energy line has been indicated to aid visualisation. A similar notation to the LTL model (bracketed numbers) has been used to identify each site type with reference to the number of monodentate connection (left number) and number of hexadentate connections (right number).*

The probabilities for growth and dissolution can thus be expressed in the following equations. The notation $L_{m,n}$ refers to the specific site type as defined by the number of monodentate linkages (m) and the number of hexadentate linkages (n). The parameters available to change during the simulations are the internal energies $U_M$ and $U_H$, these can be changed so that the spacing between each rung on the energy level diagram stays the same (ie they are scaled together) or they can be changed independently (ie they are changed separately) and the effect upon the crystal morphology and topography be observed.
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)

\[ P_{G}^{Lm,n} = \exp\left( \frac{0.5(mU_{M} + nU_{H})E}{kT} + \frac{0.5\Delta\mu}{kT} \right) \]  

Equation 3-3

And

\[ P_{D}^{Lm,n} = \exp\left( \frac{-0.5(mU_{M} + nU_{H})E}{kT} + \frac{0.5\Delta\mu}{kT} \right) \]  

Equation 3-4

3.3.3 Results and Discussion

The results presented here are presented purely to demonstrate that the model is scalable to other systems within the ABC-6 family of zeolites. The experimental observations required for a full theoretical study were very difficult to obtain due to various technical and synthetic reasons. As with the study on zeolite L the simulation program can be operated in two modes, the fixed energy level regime and the varied energy level regime. The results and discussion for these two modes will be presented separately.

3.3.3.1 Fixed Energy Level Regime

As described in the previous section the fixed energy regime is one where the energy levels are scaled but kept within their relative positions. This is done by giving the parameter \( E \) in Equation 3-3 and Equation 3-4 different value whilst keeping \( U_{M} \) and \( U_{H} \) constant. To keep the study comparative to that of the study on zeolite L the energies used in this study will be set in a similar manner. In the zeolite L case the parameters \( U_{B} \) and \( U_{H} \) where set in the ratio 1:3 and thus given energies of 1 kcal mol\(^{-1}\) and 3 kcal mol\(^{-1}\) respectively. In this model for offretite the ratio of energies can be assumed to be 1:6 for \( U_{M} \) and \( U_{H} \) respectively. To keep the studies comparable the energy of the hexadentate must be set at 3 kcal mol\(^{-1}\) so this means the monodentate energy should be set to 0.5 kcal mol\(^{-1}\). Figure 3-38 shows how the energy level schematic would appear if such a relationship is assumed, the energy levels have been grouped into 3 groups based on the number of hexadentate connections for clarity and simplicity.
Figure 3-39 showing a sequence of images where each of the columns represent 1M, 5M, 10M, 15M, 20M and 25M going from left to right. For rows a) $E = 1 \text{ kcal mol}^{-1}$ b) $E = 2 \text{ kcal mol}^{-1}$ c) $E = 3 \text{ kcal mol}^{-1}$ d) $E = 4 \text{ kcal mol}^{-1}$
Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite)

Figure 3-39 shows a selection of morphologies obtained with increasing $E$, there is no need for special 2D simulations in this case since advancements in the program development mean that crystals approaching the actual size for this system are now possible. The results in this sequence of images show the same trends as those presented in the zeolite L study. With increasing $E$ and elongation of the crystal in the c direction along with more detailed surfaces are found. This is for similar reasons of the probabilities and therefore the rates of growth in the two different directions are dependent upon exponential mathematics. The hexadentate probability and therefore rate of growth will rapidly become much larger than that of the monodentate rates.

<table>
<thead>
<tr>
<th>Energy parameter $E$ (kcal mol$^{-1}$)</th>
<th>Aspect ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1:1</td>
</tr>
<tr>
<td>2</td>
<td>1:2.8</td>
</tr>
<tr>
<td>3</td>
<td>1:5</td>
</tr>
<tr>
<td>4</td>
<td>1:12.2</td>
</tr>
</tbody>
</table>

*Table 3-5 showing the aspect ratio versus energy parameter $E$*

The aspect ratios of the resulting crystals simulated are listed in Table 3-5, experimental observations provided by Amy Holmes put the crystal aspect ratio of her syntheses at an average of 1:4.4 diameter vs length.$^{20}$ The simulations that give best correlation with surface details shown via AFM and the aspect ratio come around $E = 3$ kcal mol$^{-1}$. Figure 3-40 shows how the SEM, AFM and a simulation of $E = 3$ kcal mol$^{-1}$ compare. One can see that the surface features such as terrace shape and appearance agree quite well as does the aspect ratio and size of the overall crystal.
Although the objective of this particular study was to test the extensibility of the model rather than perform an extended study of offretite it is worth noting that compared with zeolite L, offretite appears to exhibit far more lateral spread of terraces than that observed in zeolite L. This is both apparent in the experimental observations and simulations. This is likely due to the difference in connectivity when there is no bridging needed to cross the pore structure like in zeolite L, since the site types in offretite for lateral spread are all identical unlike that in zeolite L where there are multiple types, varying the driving force in the case of offretite has little effect upon the shape of terraces. This helps to add weight and value to the reasons presented in zeolite L model as to why the driving force has an effect upon lateral spread, and the idea that the driving force could reside somewhere relating to the energy levels rather than at some point much higher than the highest energy level.

3.3.3.2 Varied Energy Level Regime
An interesting side study into the extensibility and also into the future use of the model as a predictive tool, it would be prove interesting to see what morphological range this model can show for offretite. Experimental accounts nearly always produce hexagonal rods or lozenges for offretite with little else being reported.\textsuperscript{20} As with the previous study of zeolite L the simulations can be performed in the varied

*Figure 3-40 showing a) and AFM image of the [100] face of offretite b) showing an SEM of offretite crystals exhibiting a hexagonal rod morphology c) showing a simulation at $E = 3$ kcal mol$^{-1}$ exhibiting similar morphology and surface features, also size. The AFM and SEM studies were performed by Holmes et al.\textsuperscript{20}
energy level mode where $E$ is held constant and the energies of $U_M$ and $U_H$ can be altered.

Figure 3-41 shows a matrix of the resultant morphologies as a function of the energies $U_M$ and $U_H$, it can be seen from this that by changing the stability of connections that propagate growth in different directions that it is possible to produce a diverse range of morphologies. It is a very similar case to that presented in more detail for the zeolite L study in this chapter.

On the bottom left long hexagonal rods where the hexadentate connections grow at faster rates then the bidentate connections. Towards the top left crystals are more spherical in nature owing the fact that energetically there is not much distinction between the energy levels in that region of phase space. Towards the top right where the bidentate connections grow at the faster rate hexagonal plates are found. There is a band between the hexagonal rods and plates of rounded cylinders, this occurs for exactly the same reasons as explained in detail in the previous study.

It would prove interesting to see if synthetic zeolite chemists could take this knowledge of stabilising or destabilising certain connections, by either poisoning site types or using organic templates/structure directing agents and produce morphologies in the offretite system previously thought inaccessible. A similar approach to the Brent et al. study on zeolite L, whereby changing various synthetic parameters one at a time within given ranges, utilizing the information presented in this study could assist synthetic zeolite chemists in reaching new morphological ranges.
Figure 3-41 showing an array of the resultant morphologies when the internal energies $U_M$ (top) and $U_H$ (left) are given various energy values. The experiments from the previous section on the fixed energy level regime with $E = 1, 2, 3$ and $4$ kcal mol$^{-1}$ are marked with red borders.
3.4     Erionite

In this subsection some brief information about the ERI structure and its properties will be presented and will serve to put the results obtained into perspective. A discussion on how the problem was approached from a computational perspective will be presented in this subsection. Conclusions and future studies arising from this subsection will be included in chapter 6 under the relevant subcategories.

3.4.1     Introduction to Erionite

Natural erionite was first discovered in Oregon, USA in 1898 by Eakle. The name erionite was derived from the Greek word epiov which translates to wool, this was due to natural erionite often exhibiting a wooly appearance. The structure of erionite was first elucidated by Staples and Gard in 1959. Erionite exhibits a hexagonal unit cell of lattice parameters $a = 13.2 \, \text{Å}$, $b = 13.2 \, \text{Å}$, $c = 15.0 \, \text{Å}$ and belongs to the P6$_3$/mmc space group.

Natural erionite was used commercially as a cracking catalyst in the petroleum industry. In the 1960’s Mobil Corporation started using erionite for ‘selectoforming’, in this process erionite was used as a shape selective catalyst to improve octane ratings of cracked fuels. Erionite has since been identified as being a cause of mesothelioma if the fibers are inhaled, this is a rare form of cancer commonly associated with asbestos. Since this discovery industrial use of erionite has declined by a considerable amount.

Like LTL and OFF, erionite consists of stacks of cancrinite cages linked together by D6Rs Figure 3-42, however in ERI each cancrinite cage along the stack is rotated through an angle about the stack of $60^\circ$. Three of these stacks are grouped together forming a stack of large supercages down the centre of the group, these supercages are stacked one on top of the other along the $c$ direction of the crystal. The reason this stack is not considered as a pore is because the 6 membered ring situated at each end of the supercage limits the length of the pores running along the $c$ directions. There is a network of 3 dimensional pore structures consisting of interconnected 8 membered ring pores of diameter 5.1 Å.
Natural erionite Figure 3-43 exists in abundance worldwide, with a variety of different framework compositions in terms of cation and Si/Al ratio, however the morphology the crystals exhibit is almost always hexagonal rod like, displaying the hexagonal \{001\} faces. The crystal sizes are quite varied but not usually bigger than 3mm in length.\(^{17}\)

Synthetically pure erionite is very difficult to produce, this is probably in part due to the system not being as heavily investigated as that of offretite. Most synthetic procedures for erionite will produce varying amounts of intergrown offretite.\(^{24}\) However in one synthetic study it was observed that offretite was formed from gels of low Si/Al ratio, whilst pure erionite was formed from high Si/Al gels, whilst intergrown systems were found at intermediate ratios.\(^{25}\)
3.4.2 Problem Breakdown

In this section details about the experimental approach will be discussed.

3.4.2.1 Connectivity

The approach to dealing with the erionite system was very much the same as that of dealing with the offretite system. The connectivity of erionite is exactly the same as that of offretite when considering each layer of cancrinite cages separately, however going from any specific layer to one of the layers either side of it in the \( c \) direction, the orientation of the cancrinite cages change by 60°. Figure 3-44 shows this alternating orientation and how its connectivity information can be taken account of from a computational perspective. The red and blue triangles represent the orientation and connectivities of cancrinite cages on each layer. The final image in the scheme shows two layers in the \( c \) direction overlaid against each other and presented against a Cartesian grid system.
Figure 3-44 showing how the erionite system can be broken down into simple building blocks and represented in a grid structure. a)–e) shows the thought process in going from the physical structure of erionite to a representation of the structure within a Cartesian coordinate system. The two columns show two different but adjacent layers in the (a,b) plane and how the orientation of the cancrinite cages alters the connectivity information.
Despite this rotation along the c direction, each cancrinite cage is still connected to the same number of neighbouring cages that is six in the (a,b) plane via monodentate linkages and two cages in the c direction via hexadentate linkages. This enables the site types to easily be permuted based upon connectivity, the resulting energy level scheme is exactly the same as that used for offretite, it is simply the connectivity of the cages that is different between the two systems.

3.4.2.2 Energetics
The resulting energy level scheme Figure 3-45 is identical to that of the offretite system, due to its derivation being solely based upon permuting all of the possible occupancies of neighbours for the cancrinite cage.

![Energy Level Schematic](image)

*Figure 3-45 showing the energy level schematic as based upon the attachment energy model. The internal energies of attachment of a single monodentate attachment $U_M$ and hexadentate attachment $U_H$ have been indicated. Also the arbitrary zero energy line has been indicated to aid visualisation. A similar notation to the LTL model (bracketed numbers) has been used to identify each site type with reference to the number of monodentate connection (left number) and number of hexadentate connections (right number).*

The probabilities for growth and dissolution can thus be expressed in the following equations. The notation $L_{m,n}$ refers to the specific site type as defined by the number of monodentate linkages (m) and the number of hexadentate linkages (n). The parameters available to change during the simulations are the internal energies $U_M$
and $U_H$, these can be changed so that the spacing between each rung on the energy level diagram stays the same (i.e. they are scaled together) or they can be changed independently (i.e. they are changed separately) and the effect upon the crystal morphology and topography be observed.

\[
P_{G}^{Lm,n} = \exp \left( \frac{0.5(mU_{M} + nU_{H})E}{kT} + \frac{0.5\Delta\mu}{kT} \right) \quad \text{Equation 3-5}
\]

And

\[
P_{D}^{Lm,n} = \exp \left( -\frac{0.5(mU_{M} + nU_{H})E}{kT} - \frac{0.5\Delta\mu}{kT} \right) \quad \text{Equation 3-6}
\]

3.4.3 Results and Discussion

The results presented in this subsection follow the same structure as the previous subsections of this chapter.

3.4.3.1 Fixed Energy Level Regime

As with the previous studies on zeolite L and offretite the simulations are performed in the fixed energy level regime where the energy levels are scaled relative to each other by the parameter $E$. The derivation of the energy level schematic for erionite was identical to that of offretite, only the connectivity differs in this system. So the same logic of using 0.5 kcal mol$^{-1}$ and 3 kcal mol$^{-1}$ for $U_M$ and $U_H$ applies to this system.
Figure 3.46 showing a sequence of simulations where the columns going from the left to right show the result at 1M, 5M, 10M, 15M, 20M, 25M iterations. Each row shows a different energy separation a) $E = 1$ kcal mol$^{-1}$  b) $E = 2$ kcal mol$^{-1}$  c) $E = 3$ kcal mol$^{-1}$  d) $E = 4$ kcal mol$^{-1}$. 

Chapter 3 – Results and Discussion (Zeolite L, Offretite & Erionite) 156
Figure 3-46 shows a selection of simulations for the erionite system, where the parameter $E$ is varied from 1-4 a) kcal mol$^{-1}$. It shows similar trends as those that have been described by the previous two models, the larger that $E$ is set, the greater the separation between energy levels. However due to the exponential nature of the probability formulae, probabilities depending on energy levels with hexadentate connections scale much higher more readily than those that depend upon the monodentate connections. This means the larger $E$ is, the greater the difference in rates of growth between the a/b and c directions.

<table>
<thead>
<tr>
<th>Energy parameter $E$ (kcal mol$^{-1}$)</th>
<th>Aspect ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1:1</td>
</tr>
<tr>
<td>2</td>
<td>1:2.8</td>
</tr>
<tr>
<td>3</td>
<td>1:5</td>
</tr>
<tr>
<td>4</td>
<td>1:12.2</td>
</tr>
</tbody>
</table>

*Table 3-6 showing the aspect ratio versus energy parameter $E$*

Table 3-6 shows how the aspect ratio of the simulated crystals change versus the energy scaling parameter $E$. Amy Holmes reports aspect ratios around 1:7.2 in syntheses of erionite. From the table this would put the range for $E$ between 3 and 4 kcal mol$^{-1}$. Figure 3-47 shows an AFM image detailing the surface features of erionite crystals, and an SEM image detailing the morphology of the crystals produced by Amy Holmes. Also included is a simulation ran with $E$ at 3.5 kcal mol$^{-1}$, this crystal exhibits and aspect ratio of around 1:8 which is not too far away from the experimentally reported results. In addition to this the surface features are of the similar nature to those displayed on the AFM image. Although the subject of this study was to gauge the extensibility of this model to other systems, it is important to note that the results obtained here could form the basis of an in depth study of both the offretite and erionite crystal growth. The results obtained for the energies when compared to energies for forming Si-O-Si bonds are comparable to those found in literature sources.
Figure 3-47 showing a) an AFM image on the \{100\} of a 2 μm long erionite crystal b) shows the SEM images of a 2 μm long erionite crystal (slightly up and right of centre) c) showing the result of a large simulation producing a crystal 2 μm long at $E = 3.5 \text{ kcal mol}^{-1}$. Both the AFM and SEM study was performed by Holmes et al.\textsuperscript{20}

3.4.3.2 Varied Energy Level Regime

As an interesting side study into the extensibility and also into the future use as a predictive tool, it would be prove interesting to see what morphological range this model can show for erionite. Experimental accounts nearly always produce hexagonal rods for erionite with little else being reported. Like with the previous studies of zeolite L and offretite the simulations can be performed in the varied energy level mode where $E$ is held constant and the energies of $U_M$ and $U_H$ can be altered.
Figure 3-48 showing an array of the resultant morphologies when the internal energies $U_M$ (top) and $U_H$ (left) are given various energy values. The experiments from the previous section on the fixed energy level regime with $E = 1, 2, 3$ and 4 kcal mol$^{-1}$ are marked with red borders.
Figure 3-48 shows the results of mass screening the parameter phase space for the bidentate and hexadentate energies. It is a similar story to what has been seen in the previous studies in this section, with long hexagonal rods towards the bottom left of the results matrix. At the top right where the bidentate connections are energetically stable and the hexadentate connections are energetically penalised hexagonal platelet morphologies are found. Running between these two main zones are a band of short rounded cylinders. It would be interesting to see if a similar story would unfold from a synthetic perspective as that in zeolite L, such that if one was to manipulate the synthetic conditions in a similar manner morphologies other than the fibrous hexagonal rods could be obtained. It is this morphology which makes the use of erionite so dangerous to health and hinders its use in practical and industrial applications.

3.5 References

Chapter 4

Results and Discussion

(ZIF-8 & Sodalite)
This section deals with extending the model to other types of materials, specifically in this study Zeolitic Imidazolate Framework 8 (ZIF-8) is studied along with a short expansion to test the model on silica sodalite. The main theme of this section will be the thinking and the development that has led to the current model used towards the end of this project.

4.1 Aims

- To extend the model that was developed to study zeolite L, erionite and offretite to study other zeolites with more complex frameworks in terms of connectivity.

- To branch out from zeolites and test the models suitability of modelling the crystal growth of other materials such as metal organic frameworks.

- To apply the model to study how the morphological and topographical features observed experimentally come about during crystal growth of ZIF-8.

4.2 Introduction to the Sodalite Framework

In this study both sodalite (zeolite) and the Metal Organic Framework (MOF) Zeolitic Imidazolate Framework 8 (ZIF-8) were studied, the metal organic framework has exactly the same framework as sodalite but has a totally different chemical makeup as will be discussed here. The main focus of this study is the MOF ZIF-8, however the final model developed in this study was tested on a small set of experimental data for sodalite to assess its applicability. Since the framework of ZIF-8 is based on the framework of sodalite, both materials will be introduced.

4.2.1 Sodalite (SOD)

Sodalite in its natural form was first described by Pauling in 1930.¹ Sodalite is one of the simplest zeolites in that it is constructed from only β cages. These β cages are directly fused together via 6-rings (6R’s) or 4-rings (4R’s) to make the sodalite framework. Sodalite belongs to the Im-3m space group and has unit cell parameters \( a = b = c = 8.9561 \, \text{Å} \).² This structural simplicity and the fact that many other zeolites such as LTA, EMT and FAU are built from β cages, make this system very attractive for study.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Materials with the sodalite framework have been shown to be of interest in applications such as hydrogen storage.\(^3\) It has been shown to be useful in the storage of the nuclear waste product \(^{85}\)Kr.\(^4,\)\(^5\) It has been shown that sodalite framework materials are useful industrially as molecular sieves where removal of water or small molecules would be the target.\(^14,\)\(^15\)\(^6,\)\(^7\) Materials with the sodalite framework have also been studied for use in applications based on photochromism.\(^8\)

4.2.2 Zeolitic Imidazolate Framework 8 (ZIF-8)

Metal organic frameworks (MOF’s) were introduced in chapter 1 of this thesis. Zeolitic MOF’s are materials that have framework topology based on the framework of zeolites. This means that zeolitic metal organic frameworks possess the same high surface area and systematic pore structures as their corresponding zeolites. Once more because of the nature of the framework being organic, unique properties and functionalities can be introduced into the materials.\(^9,\)\(^10\)

Zeolitic imidazolate framework 8 (ZIF-8) was initially synthesised by Huang et al.\(^11\) It is a hybrid compound constructed from 2-methylimidazolate (meIm\(^+\)) as linkers and Zn\(^{2+}\) ions. Each Zn\(^{2+}\) ion coordinates to four meIm\(^+\) linkers through coordination with the nitrogen atoms. This constitutes a single tetrahedral unit. Tetrahedral units can link to other Zn\(^{2+}\) ions and this forms an extended framework. In the case of this material, the framework has the topology of sodalite Figure 4-2.

\[\text{Figure 4-1 showing the structural framework of sodalite as viewed along the [001] direction.}^2\]
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-2 a) shows the makeup of the ZIF-8 framework, meIm ligands (linkers) coordinate Zn$^{2+}$ ions to the form tetrahedral units in the framework. These link together to form the material with sodalite topology (viewed in the [111] direction) b) shows the ZIF-8 β cage viewed along the [100] direction, the 4R can clearly be seen at the centre of the image (Note: yellow = Zn, blue = N, red = C, and green = H).

The ZIF-8 framework exhibits $I\bar{4}3m$ symmetry with unit cell parameter $a = b = c = 16.991$ Å. It contains small pores in the [111] direction with a diameter of 3.4 Å. ZIF-8 has incredible physical and chemical properties that make it unusual in the MOF family of materials, when produced using $N,N$-dimethylformamide (DMF) as the solvent ZIF-8 is thermally stable up to 450 °C and resistant to being refluxed with benzene, methanol and water for significant lengths of time. It has a surface area
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

(1630 m²·g⁻¹ BET) and a pore volume (0.636 cm³·g⁻¹) making it a good performer in absorption applications.¹²,¹³

4.3 Zeolitic Imidazolate Framework 8 (ZIF-8)

During the course of this study, three different models were considered which represent significant steps forward in thinking surrounding the crystal growth of nanoporous materials. Each model will be introduced separately along with the results obtained from corresponding simulations.

4.3.1 Model Approach 1

Until this point, all of the systems attempted so far had cages that were linked together through bonds that could be cleaved to form a single Monte Carlo building unit. The sodalite framework contains only the β cage which are all directly fused to each other, thus making it impossible to use the same building unit. This is due to the fact that the more surrounded a single site on the crystal surface is by other grown sites, the less material is needed to be added to complete the cage. In this model this problem of fused frameworks will be dealt with.

4.3.1.1 Connectivity

The approach of dealing with the sodalite system first started, by looking at the connectivity of a single β cage. Each β cage is directly fused to 14 surrounding β cages, six of these connections are through the four rings (4R) and the remaining eight are through the six rings (6R) this is show in Figure 4-3. The same logic used in the previous models was brought into this model, the idea of using closed cages as the most stable surface structure being the most important one. A full treatment of this logic can be found in coarse graining part of chapter 3.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4.3 illustrating the connections through the two different ring types (four rings (4R) and six rings (6R)).

By comparing the logic of this approach with experimental observations it can be seen that terraces on the ZIF-8 crystal exhibit heights consistent with the height of a single β cage above the surface. This is further evidenced by the fact that surface terminations of closed cages represent the most stable terminations.\(^{14,15}\)

Figure 4.4 figure taken from Pak Yan Moh’s thesis showing the terrace heights on the (110) face matching the height of a single β cage in height.\(^{15}\)
The connections described earlier in Figure 4-3 can be mapped onto a Cartesian grid Figure 4-5. Due to the fact that the sodalite framework is body centred (i.e. there is a cage at half the dimension of the unit cell) the coordinates of the connections in the 4R directions have to be two spaces apart, this facilitates the incorporation of the 6R connections on the layers in between. This can be seen in Figure 4-5 where the sites running along 4R connections are shaded red and the sites running along the 6R directions are shaded blue.

By once again invoking the Kossel model, commonly utilised in the theoretical study of crystal growth. The Kossel model is one that enables the description of site types based on the connectivity with neighbours. By considering the following scenarios it is possible to categorise site types by connectivity through rings. The most stable unit of structure is that of a site type that resides within the crystal bulk which is fully surrounded by neighbouring cages. The least stable structures would be those which only bond to a single neighbour through either the 4R or 6R. It is then possible to consider all possible sites in terms of their connections through the β cages rings.

Figure 4-6 shows one particular site that has been highlighted using two colours. The portion of the structure that is coloured green represents the part of the structure required to be attached to complete that given cage, whilst the portion of the structure that is coloured red shows the part of the structure that is already present due to neighbouring cages.
Figure 4-6 showing a site type in the framework highlighted in two colours. The part of the structure denoted by the green colour represents the portion of the structure required to be added to complete the cage, the part denoted by the red colour represents the portion of the structure that exists due to the surrounding material being present.

Utilising this method of determining the portion of the structure that needs to be added (the green portion) by counting which of the neighbours are present or not forms the premise of this study. The energetics of this approach can now be discussed in detail.

4.3.1.2 Energetics

Utilising the counting method described in the previous section for the determination site type connectivity, one can begin to consider the problem energetically. Considering sites from the energy of attachment perspective, it is clear that a site that requires little material to be added to complete a cage would be much easier energetically to complete than one requiring much more material. The most stable site would be that of crystal bulk which is totally surrounded by neighbours, and the least stable would be a cage that is only bonded to a single neighbour. There are two types of site in this model approach that are bonded to a single neighbour, those that bond through the 4R and those that bond through the 6R. The site that bonds through a single 6R requires less material to be added to complete the cage than the 4R case and would therefore be more stable under hypothetical conditions. By permuting all of the combinations of neighbour connections that are theoretically possible one can
place them on an energy scheme assuming that the 6R to 4R energy difference is 3:2 respectively Figure 4-7.

<table>
<thead>
<tr>
<th>U_{6R} (0,1)</th>
<th>(1,0)</th>
<th>U_{4R} (2,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,2)</td>
<td>(1,1)</td>
<td>(2,1)</td>
</tr>
<tr>
<td>(0,3)</td>
<td>(1,2)</td>
<td>(2,2)</td>
</tr>
<tr>
<td>(0,4)</td>
<td>(1,3)</td>
<td>(2,3)</td>
</tr>
<tr>
<td>(0,5)</td>
<td>(1,4)</td>
<td>(2,4)</td>
</tr>
<tr>
<td>(0,6)</td>
<td>(1,5)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>(0,7)</td>
<td>(1,6)</td>
<td>(2,6)</td>
</tr>
<tr>
<td>(0,8)</td>
<td>(1,7)</td>
<td>(2,7)</td>
</tr>
<tr>
<td>(0,9)</td>
<td>(1,8)</td>
<td>(2,8)</td>
</tr>
</tbody>
</table>

\[ \Delta \mu \]

<table>
<thead>
<tr>
<th>Arbitrary zero energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,1)</td>
</tr>
<tr>
<td>(0,2)</td>
</tr>
<tr>
<td>(0,3)</td>
</tr>
<tr>
<td>(0,4)</td>
</tr>
<tr>
<td>(0,5)</td>
</tr>
<tr>
<td>(0,6)</td>
</tr>
<tr>
<td>(0,7)</td>
</tr>
<tr>
<td>(0,8)</td>
</tr>
</tbody>
</table>

Figure 4-7 showing the energy schematic for the first model. Each energy level has a unique identifier which is based on the number of 4-Rings (4R) present (left number in bracketed expression) and number of 6-Rings (6R) present (left number in the bracketed expression). Also indicated on the schematic are the arbitrary zero energy, the driving force and the energy and the energies of attaching the remainder of a cage when a single 6R \( (U_{6R}) \) or 4R \( (U_{4R}) \) is present.

Figure 4-7 shows the energy scheme for this model approach, the energies of the two single neighbour site types are denoted by the energies \( U_{4R} \) and \( U_{6R} \) for the 4R and
6R case respectively. By permuting out all of the site types in terms of number of each type of connection and multiplying by these energies in the hypothetical relationship 2:3 4R and 6R respectively each site type can be placed onto the schematic. A notation (in brackets on energy scheme) that describes each site type can be utilised which describes each site type in terms of the number of 4R connections (left number) and the number of 6R connections (right number). Each site type can thus be represented in the form \( L_{(m,n)} \) where \( m \) is the number of 4R connections and \( n \) the number of 6R connections. This connectivity information can be incorporated into formulae to calculate probabilities for each site type, Equation 4-1 and Equation 4-2 show the probability for growth and dissolution cases respectively.

\[
P_{G}^{L_{m,n}} = \exp\left(\frac{0.5(2mU_{4R} + 3nU_{6R})}{kT} + \frac{0.5\Delta\mu}{kT}\right)
\]

\text{Equation 4-1}

And

\[
P_{D}^{L_{m,n}} = \exp\left(-\frac{0.5(2mU_{4R} + 3nU_{6R})}{kT} - \frac{0.5\Delta\mu}{kT}\right)
\]

\text{Equation 4-2}

The parameters \( U_{4R} \) and \( U_{6R} \) can be given various values which will change the attachment energies of bonding to 4R and 6R neighbours independently. This could potentially be used to probe the effects of cations/templates which often interact with the structure around these rings.

4.3.1.3 Results and Discussion

In this model the parameters \( U_{4R} \) and \( U_{6R} \) are available for variation to observe the effect of stabilising either the 4R or 6R structures of the \( \beta \) cage. This system makes a nice system for model development due to the fact that a high degree of control has been exerted upon the growth of ZIF-8 crystal by P. Y. Moh, the crystal topology is simple as far as nanoporous materials are concerned. The synthesis is relatively simple, being performed at room temperature and in plastic bottles producing crystals of a large size relatively quickly. All of these attributes have made it possible to gain a great insight into the crystal growth of ZIF-8. Figure 4-8 shows the overall crystal morphology taken via SEM and a sequence of images taken via ex-situ AFM. The AFM images were taken over a long period of time and represent the long term
The study into the crystals growth of ZIF-8 started with 2D simulations, this is due to the fact that 2D simulations are much simpler to interpret and understand and the evolution of ZIF-8 crystals. Such data in zeolite syntheses can be hard to come by due to the small crystals in early syntheses and the difficult conditions they require.

![Figure 4-8 showing an SEM of ZIF-8 morphology from a methanolic synthesis (top left) and a sequence of ex-situ AFM images of what the surface topography looks like over a long synthesis (2 months). This work was performed by P. Y. Moh.](image)

The study into the crystals growth of ZIF-8 started with 2D simulations, this is due to the fact that 2D simulations are much simpler to interpret and understand and the
code much simpler to develop than its 3D counterpart. ZIF-8 crystals exhibit a rhombic dodecahedral morphology as shown on the SEM in Figure 4-8. This means the crystal is bound by (110) facets only. For this reason 2D programs were constructed that dealt with the 2 dimensional connectivity of β cages on the (110) face.

Figure 4-9, shows a matrix of results when the energies $U_{4R}$ and $U_{6R}$ are modified. Some interesting patterns to note are that when the 6 rings are given low stability (small $U_{6R}$) the shape of the terrace tends towards an elongated structure, the elongation occurs due to favourable growth in sites with 4 rings (4R) which occur in the [001] direction running left to right in the simulated images. When the 4 rings are given low stability (small $U_{4R}$) in comparison to the 6R, growth tends to be slightly favoured in the [110] direction running vertically in the simulation images. Running diagonally through the results matrix along the line $U_{4R} = U_{6R}$ it can be seen that all of the terrace shapes are created Figure 4-8, however they are at different energies.
Figure 4-9 showing a selection of terrace shapes obtained when the internal energies $U_{4R}$ and $U_{6R}$ are modified, simulations where performed on a 5 by 5 μm grid with the average size being 2 μm.

Figure 4-10 shows some 3D simulations for the ZIF-8 system, the row (a-d) shows the 3D crystals that are representative of moving along the line $U_{4R} = U_{6R}$ as can be seen in Figure 4-9, moving along this line is effectively separating the energy levels. As mentioned earlier the terraces moving along that line represent the terraces observed at various stages of a real crystal growth Figure 4-8. However, it makes little sense to have the spacing between rungs on the energy level diagram to change over the evolution of the crystal growth. One observation that can be made is that the morphology (apart from the hypothetical case (a)) is the same for each crystal with the rounding of terraces being caused presumably by energy levels becoming too indistinct. Another observation to note is the apparent density of growing terraces decreases as the energy separation increases; this is caused by the rates of growth associated with the sites involved in spreading being so high that terraces are completing at a rate much greater than new ones can be nucleated.
Figure 4-10 showing 3D simulations of energies a) $U_{4R} = 0$ kcal mol$^{-1}$ $U_{6R} = 0$ kcal mol$^{-1}$ b) $U_{4R} = 1$ kcal mol$^{-1}$ $U_{6R} = 1$ kcal mol$^{-1}$ c) $U_{4R} = 2$ kcal mol$^{-1}$ $U_{6R} = 2$ kcal mol$^{-1}$ d) $U_{4R} = 3$ kcal mol$^{-1}$ $U_{6R} = 3$ kcal mol$^{-1}$ e) $U_{4R} = 2$ kcal mol$^{-1}$ $U_{6R} = 0$ kcal mol$^{-1}$ f) $U_{4R} = 2$ kcal mol$^{-1}$ $U_{6R} = 1$ kcal mol$^{-1}$ g) $U_{4R} = 1$ kcal mol$^{-1}$ $U_{6R} = 2$ kcal mol$^{-1}$ h) $U_{4R} = 2$ kcal mol$^{-1}$ $U_{6R} = 0.75$ kcal mol$^{-1}$. Crystal sizes are approximately 1 $\mu$m$^3$.

The crystals shown in Figure 4-10 e) - h) show a selection of morphologies associated deviating either side of the line $U_{4R} = U_{6R}$. Simulated crystals exhibit cubic morphology when $U_{4R} \gg U_{6R}$, quickly reverting to rhombic dodecahedral morphology. This change in morphology occurs when the nucleation site types for the facets (100) and (110) approach each other and eventually swap positions as to which is the most stable. Under the circumstances when $U_{4R} = U_{6R}$ or $U_{4R} < U_{6R}$ the site type of nucleation on the (110) $L_{(2,2)}$ is much less stabilised (higher on the energy level scheme) than that of the (100) $L_{(1,4)}$. This means that the (100) face will grow faster than the (110) face and thus the slowest growing face will be the one expressed. In the case of $U_{4R} \gg U_{6R}$ the nucleation energy level for the (100) face is higher than the (110) and as a result the (100) face is slowest growing and therefore expressed. A condition arises such that the rates of growth on each of the two facets can be balanced and as such both facets seen, this is exhibited in image h). In this model, no
access to the (111) facets can occur without artificially altering the stability of the nucleation site \( L_{(3,4)} \), this is due to the fact that no matter how the parameters \( U_{4R} \) and \( U_{6R} \) are changed this face will always be the fastest of the three facet types growing.

The next step in the study was to try to determine if the change in terrace shape could be addressed without changing the energy levels during a simulation. This was investigated by taking a 2D simulation in which the aspect ratio of the terraces matched those from the AFM. The most suitable scenario for this was using the internal energies \( U_{4R} = 2 \text{ kcal mol}^{-1} \) and \( U_{6R} = 2 \text{ kcal mol}^{-1} \) depicted in Figure 4-11. The driving force was initially held high for 200,000 iterations (first two images in the sequence), then allowed to equilibrate. This was performed to mimic the conditions thought to cause the effects seen in the real experiments shown in Figure 4-8. However in the simulation, not only does the evolving terrace not go through the stages of change in shape as the real case, but also the effects of equilibrium appear to also have a “rounding” effect upon the terraces. This would give the appearance that the model is producing the opposite observation to that of the experimental work. It could be the case that the experimental crystals never achieved true equilibrium and the subsequent “rounding” associated with crystals at equilibrium has not yet occurred, and thus alludes to the shape change during growth being as a result of more complex chemistry.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-11 showing a 3D simulation for $U_{3k} = 2 \text{ kcal mol}^{-1}$ and $U_{6k} = 2 \text{ kcal mol}^{-1}$ and a corresponding sequence of 2D experiments at the same energy. Each image in the sequence corresponds to 100,000 iterations with the final terrace measuring around 1 $\mu$m$^2$. The 3D crystal is approximately 1 $\mu$m$^3$.

This model approach proved too limited in its application, but it was a very important step in the development towards current thinking. It was not obvious at the time how to go from dealing with attachments of constant building units and attachments of given energetic value towards the current level of thinking. It was this initial attempt
at branching out to other more complex systems that provided the initial foundations for the next models presented in this section along the theme of the development of current thinking.

4.3.2 Model Approach 2

From the first model, using the stabilisation of ring structures whilst useful, poses problems with understanding the results with respect to real experimental data. Also extending this model to other zeolite/crystal or even chemical systems would prove problematic and would limit the application of the model. However the first model provided a launch pad for further development with extensibility in mind. The next model approach introduces the idea of programming in the connectivity of individual T-sites and calculating the energy required to complete a given cage based on addition of the remaining T-sites.

4.3.2.1 Connectivity

One of the main weaknesses of the previous model was that it only scales to systems that contain ring structures. This would make it difficult to extend that model to framework materials or even molecular crystals. This approach builds upon the connectivity approach described in the previous section about mapping the sodalite framework to Cartesian coordinates. However the idea of using the ring structures will be dropped from the energetic considerations.

In this model, the connectivity of individual tetrahedral sites (T-sites) will be chosen as the energetic basis. The T-sites can assume five different configurations based on their connectivity; these are $Q_0$, $Q_1$, $Q_2$, $Q_3$ and $Q_4$. This represents increasing number of connections to other T-sites. $Q_0$ would represent a free tetrahedral unit in solution whilst $Q_4$ would represent a T-site fully saturated with bonds to other T-sites. In this model, because only closed cages are being considered, sites with $Q_1$ and $Q_2$ connectivity can be disregarded because the surface will only consist of $Q_3$ and $Q_4$ and the remaining number of $Q_0$ required to complete the cage.

When considering the site type shown in Figure 4-6 the portion of the crystal that is coloured green in this model will be constructed from tetrahedral units in solution ($Q_0$). The portion of the site type coloured red will be constructed from a mixture of $Q_3$ and $Q_4$ connected T-sites. This T-site connectivity can be calculated by the fact that each T-site is connected through 4 fused cages. The site type being calculated
represents one of them so that means the occupancy of the remaining three determine the connectivity of a given T-site. For a T-site connected through one neighbouring cage its connectivity is $Q_3$, for a T-site that is connected through 2 or more T-sites its connectivity is $Q_4$. This set of rules can be used with the connectivity map of how the $\beta$ cages connect together to calculate how many $Q_3$ and $Q_4$ thus any remaining T-sites that make up the total number for the cage must be $Q_0$. It is now possible to consider the energetic aspect of this approach.

4.3.2.2 Energetics

The energy of attaching a given amount of structure to complete a cage is related to that of converting T-sites from their pre-event connectivity to their post-event connectivity. Any T-sites that are $Q_0$ before the growth even become $Q_3$ after and require the attachment of three connections. Any T-sites that are $Q_3$ before the growth even become $Q_4$ after and require the attachment of one connection. Thus one would expect that the energy $U_{Q0}$ should be 3 times larger than that of $U_{Q3}$. Any T-sites that are $Q_4$ before the growth event remain $Q_4$ after the growth event.

By considering the fact that there are 24 T-sites in the $\beta$ cage, it is possible to permute all of the combinations of $Q_0$, $Q_3$, and $Q_4$ T-sites which encompass a given site type. Each site type will have a unique combination of these three T-site connectivities. Using the fact that each site type has a unique number of these three T-site connectivity values one can use the relationship $U_{Q3}:U_{Q0}$ of 3:1 an initial energy scheme can be derived Figure 4-12.
Figure 4-12: An energy level schematic for the sodalite system model approach 2. Each energy level is calculated by counting how many $Q_0$, $Q_3$, and $Q_4$ in a given site type. These numbers are then multiplied by their corresponding internal energy parameter $U_{Q0}$ or $U_{Q3}$, in this case $U_{Q0} = 3$ kcal mol$^{-1}$, $U_{Q3} = 1$ kcal mol$^{-1}$. The energy levels corresponding to the nucleation site type of each of the three facets have been labelled.

The probabilities for growth and dissolution of a given site type $L_{(m,n,o)}$, where $m$ is the number of T-sites of $Q_0$ connectivity, $n$ is the number of T-sites of $Q_3$ connectivity and $o$ is the number of T-sites of $Q_4$ connectivity is given by Equation 4-3 and Equation 4-4. The parameter $E_T$ is to make sure that the energy level schematic is the correct way around with bulk being the most stable and the free β cage being least stable.
\[ P_{G}^{L_{m,n,o}} = \exp\left( \frac{0.5(E_T - (mU_{Q0} + nU_{Q3}))}{kT} + \frac{0.5\Delta \mu}{kT} \right) \]  
*Equation 4-3*

And

\[ P_{D}^{L_{m,n,o}} = \exp\left( \frac{-0.5(E_T - (mU_{Q0} + nU_{Q3}))}{kT} - \frac{0.5\Delta \mu}{kT} \right) \]  
*Equation 4-4*

where

\[ E_T = 24 \times U_{Q0} \]  
*Equation 4-5*

The parameters \( U_{Q0} \) and \( U_{Q3} \) can be varied to investigate the effects of changing the energy of incorporating T-sites of different connectivity upon the morphology and topography of the resulting crystal.

4.3.2.3 Results and Discussion

In this model the parameters \( Q_0 \) and \( Q_3 \) are available for variation to observe the effect of changing their stability upon the morphology and topography of the resulting crystals. Both the 3D and 2D ((110) face) variants of the algorithm were available for use at the start of this study, for some energies particularly at the larger end prove difficult to observe terraces on the 3D simulations alone. This is due to the effect large energy parameters have upon the energy levels. Large parameters separate the energy levels causing massive differentiation energetically between site types. This means that sites responsible for terraces spreading being much more stabilised than those of nucleation cause the terrace to reach the edge of the crystal before new ones can be nucleated. This gives the impression of crystals with flat faces which could be in part due to the fact that the crystals created with the 3D simulations are very small.
Figure 4-13 showing 3D simulations (top) of energies a) $U_{Q0} = 1.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 0.25 \text{ kcal mol}^{-1}$ b) $U_{Q0} = 2.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 0.50 \text{ kcal mol}^{-1}$ c) $U_{Q0} = 4.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 1.00 \text{ kcal mol}^{-1}$ d) $U_{Q0} = 8.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 2.00 \text{ kcal mol}^{-1}$ and 2D simulations of the (110) face (bottom) of energies e) $U_{Q0} = 1.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 0.25 \text{ kcal mol}^{-1}$ f) $U_{Q0} = 2.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 0.50 \text{ kcal mol}^{-1}$ g) $U_{Q0} = 4.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 1.00 \text{ kcal mol}^{-1}$ h) $U_{Q0} = 8.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 2.00 \text{ kcal mol}^{-1}$. 3D simulations produce crystals of approximate size of 0.5 μm$^3$, 2D simulations produce terraces of size 2 μm$^2$.

Figure 4-13 shows a set of 3D and 2D (110) simulations with the energy parameters that satisfy the ratio $U_{Q0} = 4:U_{Q3} = 1$. The overall morphology exhibited by the 3D simulations is that of rhombic dodecahedron, whilst the 2D simulations show a similar pattern as to what was seen in the previous model. At small energetic separations the terrace is circular, whilst with increasing separation the terrace becomes more anisotropic in shape until at the highest separation is a diamond shape. This is a similar trend to what was exhibited in the first model, that the full range of morphologies could be produced by varying the energy levels.

The morphology observed in the 3D simulations can be explained by considering the energetics of the nucleation sites on different faces of the crystal. General crystal growth theory states that the faces that are expressed are the ones which are the slowest growing. By considering the energy levels of simulations with this ratio of energies Figure 4-14, the nucleation site types have been indicated on the energy level schematic. The higher up on the diagram a site is, the more difficult that site is to grow and thus the rate of growth is lower. This means that due to the nucleation site type on the (110) being the highest on the energy level scheme, means that
growth on that face is the slowest. This is the reason that 3D simulations with this energy level configuration are comprised solely of (110) faces.

Figure 4-14 showing how the energy level diagram is configured for an energy relationship of 4:1 $U_{Q_0}:U_{Q_3}$ respectively. The nucleation site for the (100), (110) and (111) are marked.

The observations produced for the terrace shapes from the 2D simulations are not apparent until one takes a look at the site types on each of the sides of the terrace. Figure 4-15 shows a diagram of the site types responsible for nucleating a new row on each of the two terraces edges. Site A is the nucleation site of rows in the [100] direction, whilst site B is the nucleation site of rows in the [110] direction. Site A appears lower on the energy level diagram than site B by virtue of their respective configurations of $Q_0$ and $Q_3$ connected T-sites. This means at large energy separations the rate of growth in the [100] direction is fast and thus the terrace edge grows out to a point and as a result the terrace forms a diamond shape. At low energetic separation the sites are not that energetically different so growth tends to be circular. At intermediate energy separations, differing amounts of the terrace edge in the [100] direction can be observed.
Figure 4-15 showing the site configuration for creating a new row in each of the two terrace front directions.

Figure 4-16 shows the resulting 3D and 2D (110) simulations for energies satisfying the ratio $U_{Q0} = 3:U_{Q3} = 1$. The morphology exhibited by the 3D simulations show similar trends to the previously discussed case, taking rhombic dodecahedral morphology. The 2D simulations exhibit the same trend as the 4:1 case, this is for similar energetic reasons as has already been discussed Figure 4-17.

3D simulations produce crystals of approximate size of 0.5 μm$^3$, 2D simulations produce terraces of size 2 μm$^2$. 
Figure 4-17 showing how the energy level diagram is configured for an energy relationship of 3:1 \( U_{Q0}:U_{Q3} \) respectively. The nucleation site for the (100), (110) and (111) are marked.

Figure 4-18 shows the resulting 3D and 2D (110) simulations for the energies satisfying the ratio \( U_{Q0} = 2:U_{Q3} = 1 \). The morphology exhibited is still showing rhombic dodecahedral morphology. The terrace shapes shown from 2D simulations show the same trend in terms of shape that the previous experiments fitting the relationship \( U_{Q0} > U_{Q3} \). There is an overall trend in all of the simulation sets discussed thus far in this model that when the energy separation is increased in general, a reduction in the number of terraces (layers) is observed. This is because the rate of growth of sites responsible for spreading become so much greater than those responsible for nucleation new layers. If sufficiently large simulations could be achieved such that a density of terrace nucleation per unit area could calculated, then this could perhaps become a useful metric if such a parameter could be determined from experimental images. This would enable one to narrow down the energy ranges within the simulations by reducing the number of degrees of freedom within the energy parameter phase space explored.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-18 showing 3D simulations (top) of energies a) $U_{Q0} = 1.00$ kcal mol$^{-1}$ $U_{Q3} = 0.50$ kcal mol$^{-1}$ b) $U_{Q0} = 2.00$ kcal mol$^{-1}$ $U_{Q3} = 1.00$ kcal mol$^{-1}$ c) $U_{Q0} = 4.00$ kcal mol$^{-1}$ $U_{Q3} = 2.00$ kcal mol$^{-1}$ d) $U_{Q0} = 8.00$ kcal mol$^{-1}$ $U_{Q3} = 4.00$ kcal mol$^{-1}$ and 2D simulations of the (110) face (bottom) of energies e) $U_{Q0} = 1.00$ kcal mol$^{-1}$ $U_{Q3} = 0.50$ kcal mol$^{-1}$ f) $U_{Q0} = 2.00$ kcal mol$^{-1}$ $U_{Q3} = 1.00$ kcal mol$^{-1}$ g) $U_{Q0} = 4.00$ kcal mol$^{-1}$ $U_{Q3} = 2.00$ kcal mol$^{-1}$ h) $U_{Q0} = 8.00$ kcal mol$^{-1}$ $U_{Q3} = 4.00$ kcal mol$^{-1}$. 3D simulations produce crystals of approximate size of $0.5 \mu m^3$, 2D simulations produce terraces of size $2 \mu m^2$.

Figure 4-19 shows the corresponding energy level schematic for the experiment in Figure 4-18, the (110) face is still energetically the slowest growing face.

Arbitrary zero energy

Figure 4-19 showing how the energy level diagram is configured for an energy relationship of 2:1 $U_{Q0}:U_{Q3}$ respectively. The nucleation site for the (100), (110) and (111) are marked.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-20 showing 3D simulations (top) of energies a) $U_{Q0} = 1.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 1.00 \text{ kcal mol}^{-1}$
b) $U_{Q0} = 2.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 2.00 \text{ kcal mol}^{-1}$ c) $U_{Q0} = 5.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 5.00 \text{ kcal mol}^{-1}$ d) $U_{Q0} = 10.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 10.00 \text{ kcal mol}^{-1}$ and 2D simulations of the (110) face (bottom) of energies e) $U_{Q0} = 1.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 1.00 \text{ kcal mol}^{-1}$ f) $U_{Q0} = 2.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 2.00 \text{ kcal mol}^{-1}$ g) $U_{Q0} = 5.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 5.00 \text{ kcal mol}^{-1}$ h) $U_{Q0} = 10.00 \text{ kcal mol}^{-1}$ $U_{Q3} = 10.00 \text{ kcal mol}^{-1}$. 3D simulations produce crystals of approximate size of 0.5 $\mu$m$^3$, 2D simulations produce terraces of size 2 $\mu$m$^2$.

Figure 4-20 shows the 3D and 2D simulations for energies satisfying the $U_{Q0} = 1:U_{Q3}$ = 1 ratio. The 3D simulations are still exhibiting the rhombic dodecahedral morphology, whilst the 2D simulations are showing that it is becoming increasingly difficult to produce the diamond shaped terraces even at large energy separations. This is interesting because one would expect if the shape of the terraces change then some kind of change in the morphology of the crystal should be observed. This is implying that the terraces are more sensitive to the changing energy ratio than the overall morphology is. This presumably is related to subtly different site types involved in the growth of terraces on the (100) face and the growth of terraces on the (110) face that are growing in the [100] direction.
Figure 4-21 showing how the energy level diagram is configured for an energy relationship of 1:1 $U_{Q0}/U_{Q3}$ respectively. The nucleation site for the (100), (110) and (111) are marked.

According to the energy level scheme Figure 4-21 for the experiment show in Figure 4-20, the nucleation site type on the (110) face is still the least energetically favourable of the three nucleation site types. This fact explains why the crystal morphology is still entirely governed by the (110) faces.
Figure 4-22 showing 3D simulations (top) of energies a) \( U_{Q0} = 0.25 \text{ kcal mol}^{-1} \quad U_{Q3} = 0.50 \text{ kcal mol}^{-1} \)
b) \( U_{Q0} = 1.00 \text{ kcal mol}^{-1} \quad U_{Q3} = 2.00 \text{ kcal mol}^{-1} \)
c) \( U_{Q0} = 2.50 \text{ kcal mol}^{-1} \quad U_{Q3} = 5.00 \text{ kcal mol}^{-1} \)
d) \( U_{Q0} = 5.00 \text{ kcal mol}^{-1} \quad U_{Q3} = 10.00 \text{ kcal mol}^{-1} \)

and 2D simulations of the (110) face (bottom) of energies e) \( U_{Q0} = 0.25 \text{ kcal mol}^{-1} \quad U_{Q3} = 0.50 \text{ kcal mol}^{-1} \)
f) \( U_{Q0} = 1.00 \text{ kcal mol}^{-1} \quad U_{Q3} = 2.00 \text{ kcal mol}^{-1} \)
g) \( U_{Q0} = 2.50 \text{ kcal mol}^{-1} \quad U_{Q3} = 5.00 \text{ kcal mol}^{-1} \)
h) \( U_{Q0} = 5.00 \text{ kcal mol}^{-1} \quad U_{Q3} = 10.00 \text{ kcal mol}^{-1} \)

3D simulations produce crystals of approximate size of 0.5 \( \mu \text{m} \)

3D simulations produce terraces of size 2 \( \mu \text{m}^2 \).

Figure 4-22 shows the 3D and 2D (110) simulations for the energy ratio \( U_{Q0} = 1:U_{Q3} = 2 \). The 3D simulations for this energy ratio exhibit truncated cube morphology, with both the (100) and (110) facets present. The 2D simulations show the significant elongation of the terrace edge growing in the [100] direction has occurred. This implies that the rate of growth on the (100) is now much slower than in previous experiments. In fact, looking at the energy levels for this experiment Figure 4-23, it can be seen that the nucleation sites on the (100) and (110) faces are approaching each other energetically. It is for this reason that both facets are expressed in the final crystal morphology.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-23 showing how the energy level diagram is configured for an energy relationship of 1:2 $U_{Q0}:U_{Q3}$ respectively. The nucleation site for the (100), (110) and (111) are marked.

As in the previous model it has been seen in this model that all of the morphological variety can be produced using different energy configurations. It is difficult to imagine that the site types are changing their respective energetics as the crystal grows. It is possible that the driving force could be producing the changing shape observed experimentally.

Figure 4-24 showing a typical AFM image taken at 168 hours of growth as an overlay of a 2D simulation of energies $U_{Q0} = 2.0$ kcal mol$^{-1}$ $U_{Q3} = 0.50$ kcal mol$^{-1}$. Simulation and real terrace from AFM are both of size 2 μm$^2$. AFM image provided by P. Y. Moh.\textsuperscript{15}
By testing the real terrace dimensions against those produced via simulations, it was possible to select a good candidate energy configuration for investigations probing the effects of driving force. This was done by taking a real AFM terrace and producing an image overlay which could then be inserted over the top of the simulation image Figure 4-24. The best correlation was found to be the energies $U_{Q0} = 2.0$ kcal mol$^{-1}$ $U_{Q3} = 0.50$ kcal mol$^{-1}$.

![Arbitrary zero energy](image)

Figure 4-25 showing the energy level schematic of energy parameters $U_{Q0} = 2.00$ kcal mol$^{-1}$ $U_{Q3} = 0.50$ kcal mol$^{-1}$, with the constant driving force ($\Delta\mu$) experiments displayed by red lines and are of $\Delta\mu = -15$ kcal mol$^{-1}$, $\Delta\mu = -30$ kcal mol$^{-1}$, $\Delta\mu = -35$ kcal mol$^{-1}$, $\Delta\mu = -37$ kcal mol$^{-1}$ and $-37.5$ kcal mol$^{-1}$. The equilibrium driving force for this particular energy configuration has been indicated by a dashed blue line and is approximately $-42$ kcal mol$^{-1}$. The nucleation site for the (100), (110) and (111) are marked.

This energy configuration was then exposed to successive experiments where the driving force was held constant and at increasingly lower driving force. This has been illustrated how the driving force compares to the energy levels Figure 4-25. Each experiment has been denoted by a red line, not all values could be written on for space reasons, the equilibrium driving force for this energetic configuration has also been indicated by a blue line.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-26 showing a series of 2D (110) experiments of energies $U_{Q0} = 2.0$ kcal mol$^{-1}$ $U_{Q1} = 0.50$ kcal mol$^{-1}$ that were performed at constant driving force a) $\Delta\mu = -15$ kcal mol$^{-1}$ b) $\Delta\mu = -30$ kcal mol$^{-1}$ c) $\Delta\mu = -35$ kcal mol$^{-1}$ d) $\Delta\mu = -37$ kcal mol$^{-1}$ e) $\Delta\mu = 37.5$ kcal mol$^{-1}$. Each image left to right represents 100,000 iterations.

Figure 4-26 shows the resulting simulation sequences for decreasing driving force a) - e), it can be seen that progressively decreasing the driving force has little to no effect upon the shape of the terraces. However an interesting observation is that a driving force approaching that of equilibrium causes an induction period to be observed. This is a completely new concept to appear from these models and has not in the past been observed. This will form the premise of a more in-depth study into nucleation of LTA in chapter 5. The lower the driving force is the longer this
induction time is, and the longer the simulation time becomes until it is simply unfeasible to perform the calculation in a reasonable amount of time.

It is difficult to believe that the change in shape of the terraces during the crystal growth comes about as a constantly changing energy level system over time. This problem is entirely new to this line of models since in zeolites the growth can be difficult to track by AFM like that of ZIF-8. Having been presented simply with an image of the crystal morphology and some AFM images capturing the equilibrium terrace shapes, it would have been all too easy to conclude that the model could explain the observations quite well. The fact that ZIF-8 has provided this insight into the growth of such complex materials has created a significant shift in thinking towards simpler models that can act as a launch pad for developing new lines of models that incorporate more advanced concepts. It would be difficult to build in such advancements into a model that is already energetically extremely complex, however with a much simpler energetic model the complexities of cations, templates and defects could be dealt with. This is the idea behind the creation of the next model.

4.3.3 Model Approach 3

The second model went a long way to introduce more structural complexity into the model and would be extendable to other zeolitic/crystal or chemical systems. However the energy scheme was quite complicated, this makes it very difficult to understand and also has ramifications upon performance. The next model simplifies the counting process making the energy scheme much simpler. In addition to this, the idea of considering the energetics from a destabilisation rather than energy of attachment will be introduced and tested.

4.3.3.1 Connectivity

In this model, the connectivity builds upon that of the previous model however Q_3 T-sites will be neglected in this approach. The Q_3 T-sites can be neglected because by counting how many Q_0 T-sites are incorporated already factors in the energy required to make the connections to the Q_3 T-sites already present on the crystal surface. This massive simplification paved the way for further simplifications to be made in the energetic aspect of the model as will be discussed in the following section.
4.3.3.2 Energetics

By switching from the energy of attachment to considering the energetics from a destabilisation perspective, simplifications can be made to the model. In addition to this, it is much easier to relate stabilising or destabilising given cage structures to the chemistry of the crystal system from a synthetic perspective. Often the incorporation of different cations/anions or the use of organic templates/structure directing agents can significantly alter the stability of certain cage structures.

This model represents the latest thinking in the development of this project and will be the basis of work beyond this point. The basic premise behind the energetics is to consider two extremes in terms of stabilisation. The first extreme is that of the most stable or least destabilised, this is that of the bulk crystal. For the sodalite framework this would represent a fully completed β cage in which all T-sites are Q₄ connectivity due to the fact it is completely surrounded by neighbouring cages. This site type would be zero units of destabilisation energy and therefore represents a fixed zero rather than an arbitrary zero as with the attachment energy models.

The other extreme is that of creating a β cage in total isolation, this would require taking 24 Q₀ T-sites and converting them into Q₃ connected T-sites. It is then possible to imagine that any site type configuration at the surface of a crystal would fall between these two extremes. With this logic it is possible to calculate the amount by which a given site is destabilised from the crystal bulk by knowing how many Q₀ T-sites are incorporated to complete the cage.

Figure 4-27 shows such an energy level scheme, on this diagram is indicated where extremes fall and one example of where a site type on the crystal surface would fall on the energy level scheme.
Figure 4-27 showing the simplified energy level schematic for model approach 3. The crystal bulk is the theoretically most stable unit and is placed at zero destabilisation energy, the other extreme is a fully isolated β cage in which all T-sites would need to be added to complete the cage. All site types at the crystal surface fall between these two extremes.

For any site type L\textsubscript{(m)} where m is the number of Q\textsubscript{0} T-sites required to complete the cage, the probability of growth or dissolution is given by Equation 4-6 and Equation 4-7 respectively. The parameter Q\textsubscript{0} can be varied to investigate the effect upon the resulting crystal morphology and topology of changing the relative destabilisation of each site type.
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

\[ P_G^{ln} = \exp \left( -\frac{0.5(mU_{Q0})}{kT} + \frac{0.5\Delta\mu}{kT} \right) \quad \text{Equation 4-6} \]

And

\[ P_D^{ln} = \exp \left( \frac{0.5(mU_{Q0})}{kT} - \frac{0.5\Delta\mu}{kT} \right) \quad \text{Equation 4-7} \]

4.3.3.3 Results and Discussion

The model that is presented in this study represents a massively simplified model. Ideas behind this model will most likely form the basis of all future studies at the CNM which seek to add more complexity to crystal growth modeling. The only parameters available for manipulation are that of the \( U_{Q0} \) energy and the driving force \( \Delta\mu \). The main goal of this model is to capture all of the results observed in the much more complex models previously presented in this section, but produce them from much simpler ideas.

Figure 4-28 shows both the 2D and 3D simulations for increasing \( U_{Q0} \) going under high driving force conditions a) – h) and equilibrium driving force conditions i) – p). The resultant morphologies are all rhombic dodecahedron, this is due to the fact that the nucleation site type for the (110) face is the most destabilised and thus the (110) face is the slowest growing face (n(100) face requires 8\( Q_0 \), n(110) face requires 16\( Q_0 \) and n(111) requires 6\( Q_0 \). Due to the fact that in this model only the separation between energy levels can be altered, access to the (100) and (111) faces are prohibited without physically manipulating individual energy levels. However in future studies complexities such as cations and templates may provide the required energetic information required when justifying such manipulations. This will not be investigated in this study due to the enormity of the project.

Going from low to high separation of the energy levels (increasing \( U_{Q0} \)), a similar trend to the previous model can be observed where all of the different stages of terrace shape observed experimentally Figure 4-8 are obtained. This however is still at different energy separations and not in a single simulation of a specific energy.
Figure 4-28 showing 3D and corresponding 2D simulations at high driving force

- a) $U_{Q0} = 0.5$ kcal mol$^{-1}$
- b) $U_{Q0} = 1.0$ kcal mol$^{-1}$
- c) $U_{Q0} = 1.5$ kcal mol$^{-1}$
- d) $U_{Q0} = 2.0$ kcal mol$^{-1}$
- e) $U_{Q0} = 2.5$ kcal mol$^{-1}$
- f) $U_{Q0} = 3.0$ kcal mol$^{-1}$
- g) $U_{Q0} = 3.5$ kcal mol$^{-1}$
- h) $U_{Q0} = 4.0$ kcal mol$^{-1}$

and at equilibrium driving force

- i) $U_{Q0} = 0.5$ kcal mol$^{-1}$
- j) $U_{Q0} = 1.0$ kcal mol$^{-1}$
- k) $U_{Q0} = 1.5$ kcal mol$^{-1}$
- l) $U_{Q0} = 2.0$ kcal mol$^{-1}$
- m) $U_{Q0} = 2.5$ kcal mol$^{-1}$
- n) $U_{Q0} = 3.0$ kcal mol$^{-1}$
- o) $U_{Q0} = 3.5$ kcal mol$^{-1}$
- p) $U_{Q0} = 4.0$ kcal mol$^{-1}$

Approximate sizes are 1 μm$^3$ for 3D crystals and 5 μm$^2$ per 2D panel.
At low $U_{Q0}$ the terraces exhibit more rounded shape, this is due to the sites responsible for growth in any particular direction being almost indistinguishable energetically. At high $U_{Q0}$, growth prevails in the [110] direction versus the [100] direction. Due to the simplicity of this model, the only other parameter available for manipulation is the driving force $\Delta \mu$. Simulations at constant driving force were performed in an iterative decreasing manner to investigate any impact upon the shapes of growing terraces. Figure 4-29 shows how a selection of experiments close to equilibrium driving force and how they compare to the energy level diagram for $U_{Q0} = 3.5$ kcal mol$^{-1}$, the resulting images are shown in Figure 4-30.

Figure 4-29 showing how the constant driving force experiments compare with the energy level scheme.
Figure 4-30 showing a sequence of images at decreasing levels of constant driving force where a) $\Delta \mu = 22.1$ kcal mol$^{-1}$ b) $\Delta \mu = 22.0$ kcal mol$^{-1}$ c) $\Delta \mu = 21.9$ kcal mol$^{-1}$ d) $\Delta \mu = 21.8$ kcal mol$^{-1}$ e) $\Delta \mu = 21.7$ kcal mol$^{-1}$ f) $\Delta \mu = 21.6$ kcal mol$^{-1}$ g) $\Delta \mu = 21.5$ kcal mol$^{-1}$ h) $\Delta \mu = 21.4$ kcal mol$^{-1}$ i) $\Delta \mu = 21.3$ kcal mol$^{-1}$. Approximate 2D simulation panel size is 5 $\mu$m$^2$.

This particular value of $U_{Q0}$ was chosen due to the fact that the aspect ratio of the simulated terraces showed good correlation with those measured experimentally. As
can be seen in Figure 4-30 the overall final shape of the terraces in the final frames of each sequence remains the same. However at ever decreasing driving force, the final terrace is observed to shrink in size. This is due to the fact that at driving forces approaching equilibrium, a lot more dissolution events occur than at higher driving forces. Another thing to note is in the first frame of each sequence, the initial terrace is becoming much smaller as a function of decreasing driving force; this is also due to the increased occurrence of dissolution events. If one was to decrease the driving force still further towards equilibrium, an induction time effect would be seen much like that reported in the previous model study.

Another key feature can be seen in the lowest driving force case i). The first frame image indicates a different shape to those of higher driving forces such as that seen in a). Figure 4-31 shows the sequence of images in Figure 4-30 i) enlarged, it is clear to see when comparing it to higher driving forces that something has occurred at this low driving force. The key site types around the edges of the terraces responsible for propagating new layers in the [110] (vertically on figure) and [100] (left and right on figure) directions are both in dissolution mode but to different degrees. As mentioned earlier in Figure 4-15, the site responsible for growth in the [100] contains 7Q₀ T-sites whilst that in the [110] direction contains 9Q₀ T-sites. At this level of driving force, for small terraces, the numbers of each site type are highly likely helping to balance the only slight difference in energetics. However as the terrace grows larger, the site type with more favourable energetics starts to dominate growth due to being more numerous.

Figure 4-31 shows an enlarge version of the sequence i) in Figure 4-30.

It is unclear at this stage whether the change in shape of the terraces during crystal growth is related to the driving force. The in-situ experiments are very young on this class of materials. Further study experimentally is required to determine what might be happening in the crystal growth of ZIF-8. However a number of reasons could be responsible for the deviations between the results presented here in this series of
models and those of experimental study. The chemistry of MOF’s is fundamentally different to that of zeolites, so it is a possibility that the closed cage model alone is not sufficient to model all processes surrounding the crystal growth of MOF’s.

One reason could be due to the fact that in MOF’s, metal ions and linkers join together via the formation of coordinate bonds. This could potentially present complexities with the concentrations and subsequent depletions of each resource during crystal growth. A solution to this would include introducing into the model some form of expansion to the driving force to investigate the effects of varied depletion rates of metal ions or linkers respectively. This would form the basis of a large scale project and for these reasons was not attempted as part of this study due to the constraint of time.

Another reason could be due to how the MOF’s grow. Recent observations have yielded some interesting information as to how the birth and spread growth mechanism potentially differs in MOF’s to that reported in zeolites. This difference could potentially be due to the fact in zeolites, the same observations would be within the detection error and could not possibly be ruled in with current equipment. Moh et al. studied the growth on ZIF-8 in-situ, he observed that during the early stage of nucleation of a new terrace on the crystal surface, the heights did not coincide with those of closed cages for significant lengths of time Figure 4-32.
Figure 4.32 Real-time AFM deflection images and cross-sectional analyses of a developing step on the (110) face of a ZIF-8 crystal grown in the methanolic growth solution at (a) 0, (b) 2.9, (c) 4.9, (d) 7.8, and (e) 12.8 min before the stable extended growth step is formed at (f) 15.6 min. Cross sectional profiles were taken in directions parallel to the fast scan direction that is parallel to the horizontal edge of the images. The nucleus for which growth is being monitored is highlighted by the white arrow in (a). Image sizes are all 0.5 x 0.4 μm². This study was performed by Moh et al.¹⁵
What this could possibly mean is that in MOF’s in general or possibly just in the ZIF-8 system, other species than closed cages could be energetically very stable or of stabilities approaching those of closed cages. In the models derived and presented within this thesis, the assumption that the closed cages are significantly more stable than any other species formed the basis of the connectivity and energetics. However in experimental evidence reported by Moh et al., during the early formation of a new terrace, heights corresponding to various levels of completion of closed cages were observed Figure 4-33.

This could introduce the possibility of different growth regimes, the regime where growth is not governed by the completion of closed cages, for example the attachment of a layer of linkers. The second regime is that where the height of the terrace is approaching the height of the closed cages and thus the growth is shaped by the incorporation of cages. Both regimes would present different effects upon the evolution of the terrace. At early stages when the height corresponds to a layer of linkers it is plausible to consider the case where growth is less influenced by the framework topology (more circular shaped). However when the terrace corresponds to the height of closed cages; the terrace growth would be more likely to be heavily shaped by the framework topology (more diamond shaped) Figure 4-34.
This could possibly explain the reasons that the experimental observations appear to go the opposite way to those of classical crystal growth theory. Classical crystal growth theory would see the effects of equilibrium having a rounding effect upon the crystal features due to Ostwald ripening etc,\textsuperscript{16} so it is entirely plausible that near equilibrium was not attained in the study by Moh et al.

A simple test to see if simply starting with crystal terraces of a round shape and then exposing them to the energetic conditions that create the diamond shaped terraces, could potentially point in the direction that the growth is influenced by two slightly different regimes. Thus the observations in between the circular and diamond shapes could simply be intermediate stages of growth switching between the two regimes. This could perhaps indicate that crystals suffer some kind of morphology “locking” whereby it is difficult for crystals to change their morphology significantly under different conditions, taking many growth/dissolution events over a significant length of time.
Figure 4-35 showing a sequence of images in which a spherical seed (a) was subjected to energy conditions that give the diamond shaped terraces (b – f). Crystals are 0.5 μm³.

Figure 4-35 shows the result of a test where a spherical seed was pre-generated, before then subjecting it to conditions known to produce diamond shaped terraces. As can be seen the terraces go from more rounded shapes through similar shapes to those reported experimentally through to diamonds. This is just a simple test case, however it points to a more significant study that could be performed in the future whereby the model is augmented to deal with other “stable” structures.

Such a study could incorporate the different species into the driving force expressions so that the linker and the metal ions can have different depletion rates. The different species could have different starting amounts such that the effect of early depletion of one species compared to the other can be investigated. In addition to this, atomistic modeling could be incorporated into the study to try and elucidate what the stabilities of various species might be. This could be done prior to making the energetic considerations for any future models, or even incorporated into the model as a pre-calculation step. These improvements are described in the future work section of the final chapter of this thesis.

4.4 Silica Sodalite

This is a very short study that tests the extendibility of the model to other systems of the same topology. Sam Stevens carried out a study into making silica sodalite using pyrocatechol in his syntheses. Sam noticed that by synthesising crystals using
pyrocatechol he could produce crystals with a rhombic dodecahedral morphology but the terraces on the surface exhibited round shape Figure 4-36.

![Image of a sodalite crystal exhibiting mainly (110) and some (100) faces.](image)

**Figure 4-36** showing a) an SEM image of a sodalite crystal exhibiting mainly (110) and some (100) faces b) an AFM image of the (110) face exhibiting circular terraces. The images were produced by Stevens et al. 17

It is well known that pyrocatechol chelates with silica species, it is thought that silica that is bound to pyrocatechol can somehow be incorporated into the crystal framework at much lower energies. 17 If this is the case then it should have the effect of narrowing the energy levels in this model for the sodalite system. Sam was able to prove that the pyrocatechol was not having any template or structure directing effects by showing that it did not become incorporated into the crystal framework, he did this via nuclear magnetic resonance (NMR).
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

Figure 4-37 images a) – l) show a sequence of images from a simulation using the model described under approach 3 in this chapter and using an energy separation $Q_0 = 1$ kcal mol$^{-1}$. Each image represents 20M iterations, and the final crystal size is 1μm$^3$.

Figure 4-37 shows an experiment performed using the model described in the third approach in this chapter. The energy levels were set at 1 kcal mol$^{-1}$ apart, and ran at high driving force for 40M iterations Figure 4-37 a) - b) before being allowed to equilibrate naturally c) - l). This sequence of simulations would appear to add weight to the proposed argument that pyrocatechol acts as if to contract the energy levels, this is equivalent to decreasing the energy of attachment of the silica building units to the crystal. This also suggests that pyrocatechol does this indiscriminately of site type due to the fact that all energy levels in the simulation were affected by the scaling. This explains the circular nature of the terraces due to the fact that site types associated with anisotropic growth are now too close together to be energetically distinguishable.

Another key feature to note is the agreement of the overall morphology with those produced experimentally. The points of the crystal are absent in both the SEM images and the simulation results thus exhibiting small (100) faces. This is due to the fact that close to equilibrium driving force, the site types on the (100) face are under mild dissolution conditions. This is due to the fact that site types on that face are
Chapter 4 – Results and Discussion (ZIF-8 & Sodalite)

represented by the energy level $7Q_0$ and the equilibrium driving force is between $6Q_0$ and $7Q_0$.

This was a short study to test if the model was extensible to other systems with the sodalite topology. It turns out that for zeolite materials the model is showing better correlation than that of the MOF’s. This could add further evidence that zeolites and MOF crystal growth is slightly different, as described in detail earlier in the chapter. Further work on the sodalite system can be performed and is described in the future work section of this thesis.

4.5 References


Chapter 5

Results and Discussion (Zeolite A)
This chapter deals with the expansion of the model to study systems in a much finer level of detail. The focus of this chapter will be on the zeolite A system studied from the perspective of considering each of the three cages as separate entities. This will allow the scope to probe the effects of changing the stability of specific cages.

5.1 **Aims**

- To adapt the model to deal with the complexities of considering different cage types and their respective connectivity differences using a method easily generalisable (first step to generalised algorithm).
- To bring online high-throughput capability to mass screen complex systems drawing on high throughput computation.
- To study the zeolite A system from the perspective of stabilising/destabilising specific cages in the structure.
- To gain an insight as to how the connectivity and the stabilisation of specific cage types in the LTA structure affect the morphology of the crystal and also the shapes of the terraces.
- To gain an insight into the sorts of surface species (cages) that may be present under certain stability conditions.
- To gain an insight into the process that governs nucleation of the crystal with respect to driving force and what this could mean for what crystal growth scientists class as ‘high’ driving force.

5.2 **Introduction to Zeolite A**

Zeolite A was first synthesised in 1956 at the research laboratories at the Linde Air Products Company,\(^1\)\(^2\) now part of the Union Carbide Company.\(^3\) Zeolite A has the LTA (Linde Type A) framework type exhibiting the cubic space group Fm-3c and unit cell parameters a=b=c=24.61Å.\(^4\)

The zeolite A structure Figure 5-1 exhibits a three dimensional pore network with pores running perpendicular to each of the a,b and c axes. The pore opening is defined by an eight membered ring of T-sites forming an aperture of 4.2Å. The pore aperture leads into a much larger cavity 11.4Å in diameter, this cavity is the inner volume of the α-cage, the larger of the three cages that constitute the LTA framework.
Each α cage is fused to eight β cages through six membered rings, each sodalite cage in turn connects to each other via the square four membered rings and constitutes the smallest cage the double four ring (D4R).

Zeolite A can be synthesised with various morphologies, syntheses leading to various configurations of the (100), (110) and (111) have been reported. Figure 5-2 shows two examples of different morphologies of zeolite A. Image a) shows the standard synthesis from the international zeolite association standard book of syntheses (green book), crystals prepared with this method exhibit mainly (100) and (110) with the occasional occurrence of crystals with all three faces. Image b) shows a synthesis in which an organic growth modulator was used, these crystals exhibit uniform presence of the (100) face only. The main reason for such a wide range of morphologies discovered is because of the wide commercial and industrial use of zeolite A, this means the crystal growth has been subject to intense investigation.
Figure 5-2 showing typical morphologies of zeolite A a) crystals exhibit (100) and (110) faces, also is one example of a crystal exhibiting (111) face b) showing crystals exhibiting the (100) face only.\textsuperscript{10}

The major commercial application of zeolite A is in that of ion exchange in washing powders, the role of the zeolite is to soften water by removing calcium ions. Figure 5-3 shows a schematic diagram of this process, calcium ions that diffuse into the pores are captured thus releasing two sodium ions for each capture event. Aluminosilicate zeolite A shows a greater affinity for Ca\textsuperscript{2+} and Mg\textsuperscript{2+}. By capturing calcium ions from the water in detergent applications, the zeolite softens the water thus improving the performance of surfactants and also reducing the buildup of scale on machine parts.\textsuperscript{11}

Figure 5-3 showing a schematic representation of ion exchange of Ca\textsuperscript{2+} for Na\textsuperscript{+} a) the Ca\textsuperscript{2+} diffuses into the zeolite pores b) each Ca\textsuperscript{2+} replaces two Na\textsuperscript{+} ions in the framework c) the Na\textsuperscript{+} ions diffuse out of the zeolite pores.
Additional research in the detergent area has focused on the morphology of the zeolite A crystals, it has been shown that crystals that are bound by (100) faces only (cubes) damage machine parts and textile fibers over time. This is due to the sharp points and edges that these crystals exhibit. Studies showed that by changing the synthesis such that beveled edges (110 faces) mediated this problem by removing the sharp edges from the crystals.\textsuperscript{7, 8, 12, 13}

Size has also been shown to be important in ion exchange applications, one studies have shown that crystals must fall within the size range ($> 0.1 \text{ μm}$) and ($< 10 \text{ μm}$) to be effective ion exchanging crystals for calcium removal.\textsuperscript{14, 15} One study showed that the speed in which this process occurs is faster for smaller crystals approaching 1μm.\textsuperscript{14} It is clear to see how an advanced understanding of crystal growth of zeolites is extremely important to optimising their efficiency.

### 5.3 Problem Breakdown

By the time this system was investigated a generalised algorithm that could handle closed cages was available for use. The configuration file for zeolite A for use with this program can be found in appendix 3. High-throughput processing of the experimental results was also available so a high volume of simulations (approx. 150,000) simulations were performed whilst investigating this system. Before proceeding to present the results and discussions associated with this study, it is important to first discuss how this problem was approached from a connectivity and energetic perspective. This information is important in the interpretation and understanding of the results presented later in this chapter.

#### 5.3.1 Connectivity

The expansion of this model to deal with the three types of cages α, β and D4R required a vastly different approach to anything that has been seen in these studies thus far. The connectivity net of each cage was treated as separate entities, each with its own set of site types and thus energetics. The first layer of complexity deals with how the cages interconnect, Figure 5-4 shows a schematic of the LTA framework with each cage type coloured differently. The D4R (green) connects to two β cages and four α cages, in addition to this an extra complexity is that there are 3 different orientations of this cage. This was handled computationally by treating each orientation as a separate entity such that the correct net could be prescribed much
simpler. The β cage (orange) connects to six D4R’s and eight α cages, unlike in previous models of LTA by Chong and Umemura the β cages would not be permitted to form without having the D4R separating them present first.\textsuperscript{10, 16} The α cage (yellow) connects to twelve D4R’s and eight β cages.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{image.png}
\caption{Figure 5-4 showing the connectivity of cages for the LTA framework.}
\end{figure}

Due to the fact that each cage type has its own connectivity net it becomes trivial to deal with site type determination through counting neighbouring cages. However another layer of complexity exists in the form of determining the number of T-sites present and thus the number of T-sites $n$ needed to complete the cage, it is this number $n$ that will determine the site type identity and thus its energetics. This extra complexity is handled by providing information as to which T-sites are shared between which neighbours. This means that if a given neighbouring cage is present then the T-site already exists and from this logic the number of T-sites required for completing the cage can be determined trivially.

This connectivity information enabled the foundations of producing a fully generalised program that can handle any zeolitic framework consisting of space filling closed cages. It is now possible to introduce the energetics behind this system.

5.3.2 Energetics

The energetics used in this study are based on the destabilisation energy model introduced in previous sections of this thesis. The ideas that the crystal bulk is the most stable site, and that a cage in total isolation with no surrounding neighbours is the least stable configuration forms the central pillars of this model approach. In this
model an extra complexity has been added of multiple cage types has been introduced. However, once the complexities associated with the connectivities have been overcome the energetics becomes trivial. The idea that the bulk configuration is the most stable and therefore zero units of destabilisation energy remains the same. The only difference in zeolite A is that there are three bulk site configurations, one for each cage type. The same goes for the isolated cage configuration; there is one for each of the cage types.

Thus the logic follows that all of the site type configurations that can exists on a growing surface in terms of the number of T-sites \( n \) required to complete a cage of given type fall between the extremes. This means that an energy level is included for each of the possible T-site counts plus the zero bulk crystal case, this leads to 9 energy levels for D4R, 25 for \( \beta \) cage and 49 for the \( \alpha \) cage. This can be illustrated schematically Figure 5-5, in which the energy levels for each cage type are shown along with some of the surface configurations and energetic extremes (thick lines). Also indicated are the energy parameters \( U_{D4R} \), \( U_\beta \) and \( U_\alpha \) that form the basis for investigating the effects of stabilising/destabilising specific cage types.

![Figure 5-5 showing an energy schematic for the zeolite A model, indicating the energy level structure of each cage type. Also indicated is the driving force \( \Delta \mu \), and some illustrations of surface configurations for different cage types (thicker lines).](image-url)
From the energetic information the probabilities of growth and dissolution simply become:

\[ P^G_{T_n} = \exp \left( \frac{-0.5(nU)}{kT} + \frac{0.5\Delta\mu}{kT} \right) \]  \hspace{1cm} \text{Equation 5-1} \\

And

\[ P^D_{T_n} = \exp \left( \frac{0.5(nU)}{kT} - \frac{0.5\Delta\mu}{kT} \right) \]  \hspace{1cm} \text{Equation 5-2} \\

Where \( T_n \) represents the site type based on the number of T-sites \( n \) required to complete the cage. The parameter \( U \) in the above formulae simply represents the internal energies \( U_{D4R}, U_{\beta} \) or \( U_{\alpha} \) depending on which cage type is being considered.

## 5.4 Results and Discussion

The results and discussion for zeolite A are split into two sections, the first section presents the results governed by probing the stabilities of different cage types. The second section focuses on introducing the idea of investigating nucleation; this is by no means a complete study. However the science surrounding nucleation is extremely important and these are the first observations to ever be presented from this type of model. Conclusions and future work arising from this chapter can be found in chapter 6.

### 5.4.1 Crystal Growth of Zeolite A

In this model there are three parameters available for variation \( U_{D4R}, U_{\beta} \) and \( U_{\alpha} \). At about the same time as the creation of the code for this model, the scale of the computation that could be performed moved into the high-throughput paradigm. This required a number of issues to be addressed specific to simulations of this type. Where not only a vast amount of numerical data is produced but also a vast amount of image based data. This improvement was in part due to two main factors; firstly the simulation code was redeveloped to target the University of Manchester Condor Pool, this required the code to be refactored and improved as described in chapter 2 of this thesis. Secondly it required bespoke visualisation software that not only could handle the real time viewing of image based results, but also process the extremely high volumes of data generated by computing on such a scale. Both these issues were
dealt with extremely successfully, this means that this system can be studied at extreme high energetic detail.

For this system only 3D models were constructed due to the increase in complexity of a given 2D surface in which different surface terminations may arise. Initially, a very large screening experiment was performed upon this system by using the brute force approach of simply varying the three parameters $U_{D4R}$, $U_\beta$ and $U_\alpha$. The three parameters were varied between the limits of 0-10 kcal mol$^{-1}$ at an interval of 0.5 kcal mol$^{-1}$ and all permutations of these generated. By omitting the cases where a parameter was 0 kcal mol$^{-1}$, 8000 initial simulations were performed. The input parameters can be plotted in 3D space for clarity Figure 5-6, the points have only been plotted at interval of 1.0 kcal mol$^{-1}$ simply for illustrative and clarity reasons.

![Diagram](image.png)

*Figure 5-6 illustrating the variation of the three cage stability parameters $U_\alpha$, $U_\beta$ and $U_{D4R}$ when all configurations within the limits of 0-10 kcal mol$^{-1}$ are plotted. The cases where cage stability energies were 0 kcal mol$^{-1}$ were not permitted, in the real experiment the interval was 0.5 kcal mol$^{-1}$ but here for illustration purposes the interval is plotted at 1.0 kcal mol$^{-1}$. 
Chapter 5 – Results and Discussion (Zeolite A)

The initial mass screening was simply to gain an understanding of the kinds and range of morphologies that can be produced, and yielded some interesting results. It sufficed to create a basic map of how the morphology of the resulting simulations varied with input parameter phase space.

The basic trends through the data are as follows: at very small energies (~0.5 kcal mol\(^{-1}\)) the morphology is spherical owing to a very small separation of the energy levels. The vast majority of the phase space was covered by cube shape crystals (100) largely owing to the cubic interconnectivity of the substituent cages. However there is an “interesting zone” where combinations of higher order faces such as (110) and (111) can be observed. This interesting zone occurs within the parameter space of limits of \(U_{D4R} = 4\text{–}10\text{ kcal mol}^{-1}\), \(U_\beta = 3\text{–}7\text{ kcal mol}^{-1}\) and \(U_\alpha = 0.5\text{–}3\text{ kcal mol}^{-1}\). Some of the resulting morphologies from the initial screening can be seen in Figure 5-7 along with the energy parameters they occur.

![Figure 5-7 showing a selection of equilibrium morphologies from the experiments marked in Figure 5-6](image)

a) \(U_{D4R} = 4\text{ kcal mol}^{-1}\) \(U_\beta = 8\text{ kcal mol}^{-1}\) and \(U_\alpha = 1\text{ kcal mol}^{-1}\) b) \(U_{D4R} = 1\text{ kcal mol}^{-1}\) \(U_\beta = 2\text{ kcal mol}^{-1}\) and \(U_\alpha = 2\text{ kcal mol}^{-1}\) c) \(U_{D4R} = 8\text{ kcal mol}^{-1}\) \(U_\beta = 4\text{ kcal mol}^{-1}\) and \(U_\alpha = 0.5\text{ kcal mol}^{-1}\) d) \(U_{D4R} = 2\text{ kcal mol}^{-1}\) \(U_\beta = 4\text{ kcal mol}^{-1}\) and \(U_\alpha = 1\text{ kcal mol}^{-1}\) e) \(U_{D4R} = 8\text{ kcal mol}^{-1}\) \(U_\beta = 1\text{ kcal mol}^{-1}\) and \(U_\alpha = 0.5\text{ kcal mol}^{-1}\) f) \(U_{D4R} = 8\text{ kcal mol}^{-1}\) \(U_\beta = 2\text{ kcal mol}^{-1}\) and \(U_\alpha = 1\text{ kcal mol}^{-1}\) g) \(U_{D4R} = 4\text{ kcal mol}^{-1}\) \(U_\beta = 4\text{ kcal mol}^{-1}\) and \(U_\alpha = 2\text{ kcal mol}^{-1}\) h) \(U_{D4R} = 8\text{ kcal mol}^{-1}\) \(U_\beta = 4\text{ kcal mol}^{-1}\) and \(U_\alpha = 1\text{ kcal mol}^{-1}\). Approximate size of crystals is 0.5 μm\(^3\).
This “interesting zone” could then be explored at a much higher resolution. This is in order to try to gain a better understanding as to why the morphologies observed in that zone occur there. To investigate this special zone and try to elucidate if crystals with realistic morphologies could be produced by the model, simulations were performed within the limits $U_{D4R} = 4-10$ kcal mol$^{-1}$, $U_\beta = 3-7$ kcal mol$^{-1}$ and $U_\alpha = 0.5-3$ kcal mol$^{-1}$.

These simulations were performed at a resolution of 0.1 kcal mol$^{-1}$ in each of the parameter space dimensions within the range specified. This yielded a further 60,000 datasets of specific interesting morphologies. Figure 5-8 shows this “interesting zone” marked in red, however for clarity not all the experiments that were performed have been plotted.

*Figure 5-8 shows the simulations performed at higher parameter space resolution marked red with the previous experiments marked black. For simplicity and clarity only half the resolution in the $U_\alpha-U_\beta$ has been plotted and one tenth of the resolution in the $U_{D4R}-U_\beta$ has been plotted.*
Within this new much higher resolution dataset, morphologies that resemble those of crystals produced experimentally were produced Figure 5-9. Some exhibiting all three of the main faces (100), (110) and (111). Interestingly, in simulations with the faceting ratio exhibited by such as the one in f); the terraces on all of the three crystal faces, correlated extremely will with those observed in experimental studies of the zeolite A synthesis from the international zeolite association book of standard syntheses.

![Figure 5-9 showing a selection of equilibrium morphologies from the experiments marked in red in Figure 5-8](image)

- **a)** $U_{d4r} = 7$ kcal mol$^{-1}$, $U_\beta = 1.6$ kcal mol$^{-1}$ and $U_\alpha = 0.6$ kcal mol$^{-1}$
- **b)** $U_{d4r} = 8$ kcal mol$^{-1}$, $U_\beta = 3.8$ kcal mol$^{-1}$ and $U_\alpha = 0.8$ kcal mol$^{-1}$
- **c)** $U_{d4r} = 8$ kcal mol$^{-1}$, $U_\beta = 4.6$ kcal mol$^{-1}$ and $U_\alpha = 1$ kcal mol$^{-1}$
- **d)** $U_{d4r} = 8$ kcal mol$^{-1}$, $U_\beta = 5.2$ kcal mol$^{-1}$ and $U_\alpha = 1.4$ kcal mol$^{-1}$
- **e)** $U_{d4r} = 8$ kcal mol$^{-1}$, $U_\beta = 5.2$ kcal mol$^{-1}$ and $U_\alpha = 1$ kcal mol$^{-1}$
- **f)** $U_{d4r} = 8$ kcal mol$^{-1}$, $U_\beta = 5.4$ kcal mol$^{-1}$ and $U_\alpha = 1$ kcal mol$^{-1}$
- **g)** $U_{d4r} = 9$ kcal mol$^{-1}$, $U_\beta = 5.4$ kcal mol$^{-1}$ and $U_\alpha = 1$ kcal mol$^{-1}$
- **h)** $U_{d4r} = 10$ kcal mol$^{-1}$, $U_\beta = 5.4$ kcal mol$^{-1}$ and $U_\alpha = 1$ kcal mol$^{-1}$

Approximate size of crystals is 0.5 μm$^3$.

The initial screening experiments demonstrated the power of this model however, understanding the observations produced were another matter. Early attempts to try to gain an overall understanding were hampered by: the fact that there was so much diversity in the morphology of resulting crystals simulated. To complicate things even further, early evidence started to point to diversity in the topographical detail on the surfaces of simulated crystals. In addition to these complications, the model was highly sensitive to changing those input parameters $U_{d4r}$, $U_\beta$ and $U_\alpha$. 221
Chapter 5 – Results and Discussion (Zeolite A)

Figure 5-10 showing a series of equilibrium images that demonstrate the sensitivity of the model where in all cases $U_{D4R} = 8$ kcal mol$^{-1}$ and $U_{a} = 1$ kcal mol$^{-1}$ whilst $U_{\beta}$ is varied as follows a) $U_{\beta} = 1.84$ kcal mol$^{-1}$ b) $U_{\beta} = 1.85$ kcal mol$^{-1}$ c) $U_{\beta} = 1.86$ kcal mol$^{-1}$ d) $U_{\beta} = 1.87$ kcal mol$^{-1}$ e) $U_{\beta} = 1.88$ kcal mol$^{-1}$ f) $U_{\beta} = 1.89$ kcal mol$^{-1}$ g) $U_{\beta} = 1.90$ kcal mol$^{-1}$ h) $U_{\beta} = 1.91$ kcal mol$^{-1}$ i) $U_{\beta} = 1.92$ kcal mol$^{-1}$ j) $U_{\beta} = 1.93$ kcal mol$^{-1}$ k) $U_{\beta} = 1.94$ kcal mol$^{-1}$ l) $U_{\beta} = 1.95$ kcal mol$^{-1}$. Approximate crystal size is 0.5 μm$^3$.

This extreme sensitivity can be seen in the sequence of simulations shown in Figure 5-10. In this experiment the energies $U_{D4R}$ and $U_{a}$ were both held constant at 8 kcal mol$^{-1}$ and 1 kcal mol$^{-1}$ respectively, whilst $U_{\beta}$ was varied by 0.01 kcal mol$^{-1}$ in the range 1.80 - 2.00 kcal mol$^{-1}$. A snippet of this range is shown, and one can observe a significant change in the morphological and topological features, this occurs over an extremely small change of energy. This was starting to create problems with proposing new experiments, with some proposed that would exceed the 1 million
simulations mark. Experiments of this size would almost certainly require storage in the tens of terabyte scale, thus introduced difficulty. However to avoid such scale of computation, a new approach to data set visualisation was investigated at this point. It was noticed that the same patterns of morphologies were occurring throughout the data set. The repeat patterns occurred on a plane by plane basis through the data set, where the edges of these planes would intersect the $U_{D4R}$, $U_{\beta}$ and $U_{a}$ axes at the same value (thus the planes would be triangular Figure 5-11).

![Figure 5-11 showing a schematic illustrating the plane by plane pattern of data trends through the data set.](image)

To verify that this is the case; a number of experiments that would lie at the same point within a given plane, but on different successive planes were chosen. Figure 5-12 shows three different series of experiments that prove that this plane relationship is true. In each of the three cases at higher energies (planes further from the origin) the terrace density decreases, this is an observation that has been seen in previous models in this thesis for similar energetic effects. This means that a single triangular plane can be fully explored at a reasonable resolution. Then with suitable experimental data where the density of terraces can be compared the energetics can be pinned down. This is a vast simplification that shows the power of data visualisation concepts in interpreting complicated mixed quantitative-qualitative systems.
Chapter 5 – Results and Discussion (Zeolite A)

Figure 5-12 showing three series of images of crystals at equilibrium driving force with the same energy ratios a) – d) in the ratio 1:1:1 where a) $U_{D4R} = 1$ kcal mol$^{-1}$, $U_{β} = 1$ kcal mol$^{-1}$ and $U_{α} = 1$ kcal mol$^{-1}$, b) $U_{D4R} = 2$ kcal mol$^{-1}$, $U_{β} = 2$ kcal mol$^{-1}$ and $U_{α} = 2$ kcal mol$^{-1}$, c) $U_{D4R} = 3$ kcal mol$^{-1}$, $U_{β} = 3$ kcal mol$^{-1}$ and $U_{α} = 3$ kcal mol$^{-1}$, d) $U_{D4R} = 4$ kcal mol$^{-1}$, $U_{β} = 4$ kcal mol$^{-1}$ and $U_{α} = 4$ kcal mol$^{-1}$, e) – h) in the ratio 5:5:1 where e) $U_{D4R} = 2.5$ kcal mol$^{-1}$, $U_{β} = 2.5$ kcal mol$^{-1}$ and $U_{α} = 0.5$ kcal mol$^{-1}$, f) $U_{D4R} = 5$ kcal mol$^{-1}$, $U_{β} = 5$ kcal mol$^{-1}$ and $U_{α} = 1$ kcal mol$^{-1}$, g) $U_{D4R} = 7.5$ kcal mol$^{-1}$, $U_{β} = 7.5$ kcal mol$^{-1}$ and $U_{α} = 1.5$ kcal mol$^{-1}$, h) $U_{D4R} = 10$ kcal mol$^{-1}$, $U_{β} = 10$ kcal mol$^{-1}$ and $U_{α} = 2$ kcal mol$^{-1}$, i) – l) in the ratio 2:2:1 where a) $U_{D4R} = 1$ kcal mol$^{-1}$, $U_{β} = 1$ kcal mol$^{-1}$ and $U_{α} = 0.5$ kcal mol$^{-1}$, b) $U_{D4R} = 2$ kcal mol$^{-1}$, $U_{β} = 2$ kcal mol$^{-1}$ and $U_{α} = 1$ kcal mol$^{-1}$, c) $U_{D4R} = 4$ kcal mol$^{-1}$, $U_{β} = 4$ kcal mol$^{-1}$ and $U_{α} = 2$ kcal mol$^{-1}$, d) $U_{D4R} = 8$ kcal mol$^{-1}$, $U_{β} = 8$ kcal mol$^{-1}$ and $U_{α} = 4$ kcal mol$^{-1}$. Approximate crystal size is 0.5 μm$^3$

Using the logic described thus far it was then possible to construct bespoke applications that can enable the visualisation of the results from such an approach. The basic premise used for such an application was to plot the image based data onto a ternary plot, where the input parameters $U_{D4R}$, $U_{β}$ and $U_{α}$ where expressed as a percentage of the total energy of the three parameters summed Equations 5.3 to 5.6.
Using percentages makes the comparison of successive planes much simpler than absolute energies, thus simplifying things further.

\[ E_T = U_{D4R} + U_\beta + U_\alpha \]  

\[ E_{D4R}(\%) = \frac{U_{D4R}}{E_T} \times 100 \]  

\[ E_\beta(\%) = \frac{U_\beta}{E_T} \times 100 \]  

\[ E_\alpha(\%) = \frac{U_\alpha}{E_T} \times 100 \]

It is possible to generate a set of points mathematically that correspond to exploring a given plane at given percentage interval, where 100% of either energy parameter represents a point intersecting the axis of that given energy parameter.

The three cage types can then be placed at the corners of a ternary plot where their energy would represent 100% of the total, and all of the images created by running experiments at a given interval can then be rendered at the corresponding coordinate of the ternary plot. Figure 5-13 shows an example of doing just this. This ternary plot is a very small example created simply for illustration in this thesis. It contains 1176 images when a given plane is explored at a percentage interval of 2%. The ternary plot used in this study was of much higher resolution consisting of 17,955 data points, corresponding to exploring the plane at 0.5% intervals. However this larger plot is difficult to include since it is so large that one would see virtually nothing when scaled down to even A0 paper size.

What this means in terms of the destabilisation energies used in this model, is that the closer to one of the points of the ternary plot a data point is, the more destabilised that particular cage type is (larger energy of destabilisation). The other two cage types will be smaller percentages and therefore more stabilised by comparison.
Figure 5.13 showing a ternary plot consisting of 1176 simulation data points corresponding to exploring the energy space at 2% intervals, all images are at equilibrium driving force. The highest destabilisation for each cage is at the corner for which its name appears and the corresponding axes are colour coded. The directions of the grid lines for each axis can be seen in the small diagram top right.
Figure 5-14 showing the same ternary plot with some interesting morphologies pulled into focus. All crystals are at equilibrium driving force.
Chapter 5 – Results and Discussion (Zeolite A)

The ternary plot application is a fully interactive map of morphologies, by clicking a given thumbnail image a large image can be obtained. Figure 5-14 shows a selection of enlarged images that exhibit different morphological features as well as topographical features. By inspection of the data in the ternary plot, the morphological features can be illustrated schematically Figure 5-15.

![Ternary Plot Diagram]

*Figure 5-15 a schematic representation of the equilibrium morphology zones exhibited in the ternary plots. The point marked as the centre of stability is the central point where the percentage destabilisation energy for each cage is equal.*

From this ternary schematic it is possible to get a greater understanding of the energy phase space. The zone mark as spherical (blue) is an area that is governed by destabilisation energies that very small for the β cage and the D4R. This means that the energy levels are not that distinguishable in terms of energy, thus spherical crystals result. The vast majority of the energy phase space is occupied by crystals that are exhibit the cube (red) morphology (100 only). This includes the energies at and around the centre of stability point where all the destabilisation energies are equal.

The “interesting zone” that was referred to in earlier experiments can now be accurately mapped as consisting of a centre band (green), consisting of dodecahedral crystals (110 only). Surrounding this central band is a transition zone (orange) where intermediate stages between the extremes of being a cube and dodecahedron. It is
within this transition band that crystals with combinations of all of the three main faces (100), (110) and (111) can be found.

The sensitivity of the effect of the input parameters on morphology can be clearly seen by the extremely narrow nature of this band. Another interesting observation to note is that crystals that involve the (111) face, are even more sensitive that those containing only the (100) and (110). They would occupy an extremely narrow band running along the central region of the transition band both above and below the central band marked in green. The area where crystals would be found that exhibit the (111) would not be much wider than 0.25% at its widest point. Since a wide array of diverse morphologies with the (111) face present exist, and they all have to fit into such a narrow band of energy percentages. This means that it is extremely easy to miss some of the detail is sufficient resolution is not used.

Another mode that can be utilised within the visualisation platform is that of colouring the cages present at the crystal surface independently. Since the colouring convention of red (α cage), blue (β cage) and green (D4R cage) has been used in all energy diagram axes used throughout this thesis, this convention will be adhered to when colouring the cages themselves. It is possible to create ternary plots that utilise the colouring of cages separately, thus it is possible to explore the energy phase space in terms of surface terminating cages. This knowledge has long been one of the cruxes when studying the zeolite A system, since so many of the same combinations of the terminating units produce the same heights or are approaching the resolution measurable by the AFM.\textsuperscript{7, 8, 12, 13}

Figure 5-16 shows some examples of crystals that have been coloured in such a way that each of the cage types has the colours mentioned earlier. The method of comparing morphology and the surface terminations allow access to a very powerful method of screening large volumes of visual data, with the potential to provide an answer to some old lingering questions.
Figure 5-16 a ternary plot displaying a selection of equilibrium crystals that have been coloured on a cage by cage basis α cages (red), β cages (blue) and D4R cages (green). The axes directions through the ternary plot have been included top right.
It can be seen from Figure 5-16 that the enlarged crystals adopt a certain colour as a result of the combination of colours that the surface cages present take. As with the morphology case, a schematic representation of these features can be created to show where the different features occur in energy phase space for equilibrium crystals Figure 5-17.

Although the combinations can be quite complex, it is possible to utilise simple logic to deduce the surface terminations and understand why they occur where they do. Surfaces coloured primarily by one cage type can be found towards the centre of the axis opposite to the cage label. Crystals coloured by α cages (red) occur in the region centre and bottom of the triangle, this is where α cage is the most stabilised and the β and D4R cages are roughly destabilised by the same amount. This same logic applies to the other two cages, crystals coloured by β terminations (blue) occur centre upper left. Whilst crystals that are coloured primarily by D4R (green) occur centre upper right.

Moving towards the points of the ternary plot, the surface gets a little more complicated, but can be explained rather simplistically in terms of the destabilisation of the three cages. The top point the colour cyan, this is the result of blending blue

*Figure 5-17 showing a ternary plot that has a blended colour chart to represent what the average colour of the crystal takes as a result of its surface terminations. The morphology zones have been overlaid for clarity.*
and green, energetically the α cage (red) is most destabilised at this point whilst the β (blue) and D4R (green) cages are equally destabilised. By having a crystal surface terminating with β and D4R cages, one would expect the crystal to appear cyan from distance. The same logic can be applied to explain the colours of the regions around the other two points of the ternary plot. Towards the centre of stability, the surface terminations are mixtures of all three of the cage types.

A logical place to start when investigating experimental observations is that of the centre of stability Figure 5-18. At this point all three cages have the same destabilisation energy percentage, and therefore have the same physical energy values for the parameters $U_{D4R}$, $U_β$ and $U_α$. It is unclear whether or not real crystals can be made that correspond to this point, since any interference from structure directing agents, framework stabilising species or even structural steric properties could cause deviation. However, this point is a good starting point of reference since any of these types of effects will deviate the stability of certain cage structures, the magnitude by which is probably nowhere near as extreme as those ranges produced by the ternary plots.

![Figure 5-18 showing the crystal and (100) surface for a) high driving force (coloured by layers) b) high driving force (coloured by cages) c) equilibrium driving force (coloured by layers) d) equilibrium driving force (coloured by cages) for the centre of stability case where $U_{D4R} = U_β = U_α$. Crystal size is approximately 0.5 μm³.](image)

As can be seen the resulting energy for the centre of stability lies well within the cube morphology zone and as such this is the morphology seen in the simulation. The
terraces however are circular both at high driving force and equilibrium driving force. This is due to the fact that the parameters $U_{D4R}$, $U_\beta$ and $U_\alpha$ are equal in energy. Looking at the surface it can be seen that a carpet of D4R cages (green) is visible on the surface of the crystal, this cage being the easiest to grow since the lowest numbers of $Q_0$ T-sites are required to complete the D4R cage. However due to the fact these sites are quickly saturated, the shape is mainly governed by the inclusion of $\beta$ (blue) and $\alpha$ (red) cages. This is due to the fact that the number of $Q_0$ T-sites required to complete either of the $\alpha$ or $\beta$ cages on such a surface populated by D4Rs is approximately the same, owing to a rounded shape.

By first considering the Thompson synthesis of zeolite A.\textsuperscript{17, 18} A standardised synthesis from the book of verified syntheses, the effect of moving away from the centre of stability. This synthesis has been studied experimentally via AFM and SEM in extremely great detail.\textsuperscript{19}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5-19}
\caption{An SEM of zeolite A produced via the Thompson synthesis.\textsuperscript{19}}
\end{figure}

From the SEM image shown in Figure 5-19 it can be seen that the crystals are of truncated cubic morphology, with the (100), (110) and (111) facets present to varied degrees. So when considering the energy phase space ternary plots, it would make sense to move away from the centre of stability towards the transition morphology zone. This has been illustrated in Figure 5-20. A number of simulations at a high resolution were conducted along the line indicated on the diagram by a white arrow.
Figure 5-20 showing the direction of exploration through energy phase space when investigating the Thompson synthesis.

Figure 5-21 shows the resulting 3D simulation images obtained for the point represented by the black star in Figure 5-20. The crystals exhibit all three facets (100), (110) and (111) as also shown by the experimental images. The facet fraction is extremely difficult to capture in exactly the same ratio, however all simulations performed within the local vicinity of this same data point have the same terrace terminations on each facet. The following simulated crystal was grown at $E_{UD4R} = 37.5\%$, $E_{\beta} = 37.5\%$ and $E_{\alpha} = 25\%$ or physical energies $U_{UD4R} = 1.5 \text{ kcal mol}^{-1}$ $U_\beta = 1.5 \text{ kcal mol}^{-1}$ $U_\alpha = 1.0 \text{ kcal mol}^{-1}$. 
Figure 5-21 showing the resulting crystals grown at $E_{UDR} = 37.5\%$, $E_{U\beta} = 37.5\%$ and $E_{U\alpha} = 25\%$ or physical energies $U_{UDR} = 1.5$ kcal mol$^{-1}$, $U_{U\beta} = 1.5$ kcal mol$^{-1}$, $U_{U\alpha} = 1.0$ kcal mol$^{-1}$ a) shows the high driving force image b) equilibrium driving force image. Crystals are approximately $0.5m^3$.

By colouring the crystal on a per cage basis it is possible to compare the surface terminating cages with those proposed in earlier experimental studies. Figure 5-22 a) shows the (100) face of a simulation that has been colour by cage type. By comparing the cage terminations with those in b) it can be seen that the surface corresponds exactly with the top structure proposed by Agger.$^{13}$ The shape of the terraces shown in c) also shows good correlation with those seen in Figure 5-21.

Figure 5-22 showing a) the surface terminating units on the (100) face near equilibrium b) showing the possibilities of surface terminations proposed in experimental studies$^{14}$ c) an AFM image of terraces near equilibrium.$^{19}$
The structural terminations on the (110) face from the simulation can be seen in Figure 5-23 a). When comparing with the result observed experimentally the result is extremely similar, the only difference being is that the D4R cage is present at the top of the α cage position. By measuring the heights via the AFM it would be very difficult to distinguish between the two terminations since they have the same height. However, in simulations performed in the local energy phase space, this surface termination does not change. From c) it can be seen that the shape of the terraces are of the same shape as those on the (110) face in Figure 5-21.

The structural terminations on the (111) face from the simulation can be seen from Figure 5-24. By comparing the surface terminations obtained from the simulation a) and those proposed from an experimental study by Cubillas,19 it can be seen that the simulated and proposed surface terminations match exactly. Further to this, by comparing the shape of the terrace in c) to those on the (111) face in the 3D crystals shown in Figure 5-21 a good correlation can be seen with the shape. This is particularly evident on the high driving force image; however a merger of two terraces has complicated the shape on the equilibrium crystal. Due to the small size of the (111) face on the simulated crystals it can be difficult imaging the terraces.
Figure 5-24 showing a) the surface terminating units on the (111) face near equilibrium b) showing the most stable surface termination proposed from experimental studies c) an AFM image of terraces near equilibrium.

The correlations seen thus far between the modelling and experimental data show huge promise in gaining an understanding of the processes involved in the crystallisation of zeolites. In terms of our model the Thompson synthesis can be thought of as stabilising the α cage with respect to the centre of stability point. Other studies on the zeolite A system have revolved around the use of organic molecules as morphology modifiers.

An on-going experimental study by Pablo Cubillas in which the organic molecules triethanolamine (TEA) and diethanolamine (DEA), were added to syntheses of zeolite A to investigate their effect on morphological and topological detail. Figure 5-25 shows a series of SEM images of zeolite A prepared with a) TEA b) TEA + DEA and c) DEA only. The cases where TEA are present, exhibit sharp cubic morphology with the (100) being the only face expressed. When DEA only is present both the (100) and (110) faces are present on the crystals observed.
Figure 5-25 showing examples of the morphology of zeolite A a) in the presence of TEA b) in the presence of TEA + DEA c) in the presence of DEA. SEM images were provided by Pablo Cubillas.

As with the previous system it makes sense to consider the energy phase space with respect to the centre of stability and the transition zone. It was seen earlier that moving towards the transition zone that good correlation with experimental observations can be found. Whilst performing the previous experiment to find the (111) face for the Thompson synthesis, a number of candidates for the TEA and DEA cases were identified.

Figure 5-26 showing the resulting crystals at equilibrium driving force grown at a) $E_{UDAR} = 36.4\%$ $E_{U\beta} = 36.4\%$ and $E_{U\alpha} = 27.2\%$ b) $E_{UDAR} = 41.7\%$ $E_{U\beta} = 41.7\%$ and $E_{U\alpha} = 16.6\%$. Crystals are approximately 0.5m$^3$.

Figure 5-26 shows two candidates for the TEA (a) and DEA (b) cases. To produce the TEA candidate, the $\alpha$ cage was stabilised with respect to the centre of stability data point, it was not stabilised as much as the case for the Thompson synthesis case,
hence the morphology remains cubic. The DEA case was created by stabilising the $\alpha$ cage even further and beyond the Thompson synthesis case, well into the morphology transition zone. The AFM images performed upon the (100) face of the crystals show good topographical correlation with the simulation candidates Figure 5-27.

Another observation Cubillas made was that upon reacting the DEA synthesis for a length of time longer than 10 days the appearance of the (211) facets became apparent, it is thought that this is due to the metastability of zeolites and that a competing phase is causing dissolution of the zeolite A crystals. Figure 5-28 shows the comparison of the DEA simulation candidate described previously when exposed to mild dissolution conditions. Also shown are an SEM exhibiting the (211) faces, and an AFM of the (110) face during this dissolution period. As can be seen a good correlation can be seen in this scenario with both the mild dissolution conditions producing the (211) faces on the simulated crystal. But also with the terraces shapes on the (110) face of the crystal becoming a very rough diamond shape under dissolution. This is still an on-going study, where more understanding will be gained using experimental evidence along with supporting simulations in the near future.

Figure 5-27 showing AFM images of the (100) faces of a) the TEA synthesis b) the DEA synthesis. Images were provided by Pablo Cubillas.
The work presented within this chapter shows that this line of thinking surround the destabilisation of specific cage units can show strong correlation with experimental data. One can imagine that it would be a less complex design process if synthetic zeolite chemists could relate the stabilisation/destabilisation of certain structural units to specific crystal habits. This is where the predictive element of this model could provide assistance with the design of crystals with previously difficult to envisage morphology and topological properties.

5.4.2 Nucleation

As noticed in the studies presented in previous sections of this thesis; when decreasing the driving force close to the equilibrium driving force, an induction period is introduced. This has been shown to occur on 2D simulations, so it was interesting to see if the effect could be produced in the 3D simulations. This will enable the investigation of nucleation processes, such as the dependence of the induction period upon driving force and the critical nucleus size. This study was only to lay the foundations of a much larger future study into nucleation due to time constraints. However considering how important this process is and the fact it has not been observed/reported in any model of the kind previously it is worth giving a treatment of its current state.
As discussed in chapter 1 of this thesis, for a crystal to be nucleated and proceed to further growth it must reach a critical size of nucleus. The time taken to reach this critical nucleus is known as the induction time. The period of interest along the crystallisation curve for this study is the part highlighted in red in Figure 5-29.

Experimental zeolite chemists can produce crystallisation curves that can track the size of the crystal over synthesis time. One could then find the intercept with the time
axis along the part where the curve is linear to yield an approximate induction time. The ratio of the time taken to form the critical sized cluster $t_1$ and the time taken to achieve a given sized crystal $t_2$ can then be compared to the same values from the simulation.

A number of technical obstacles had to be circumvented to produce an algorithm that could be used to study nucleation. The first major obstacle was, until this point the equivalence of the crystallisation curve from the simulations was measured versus iteration number rather than time. Many authors have shown that the time at a given iteration $t$ can be incorporated into Monte Carlo algorithms by incrementing the previous iterations time $t_s$ by a stochastic time $\Delta t_n$ at each iteration $n$.

$$ t = t_s + \Delta t_n $$

*Equation 5-7*

They showed that $\Delta t_n$ depended only on the sum of the probabilities $Q_n$ of all events at a given iteration $n$, and a time scaling factor $\tau$. The usual dependence upon the random number seen in most Monte Carlo approaches can be dropped in this case. Since the random numbers and the calculation of even probabilities are not correlated.

$$ \Delta t_n = \frac{\tau}{Q_n} $$

*Equation 5-8*

Where,

$$ Q_n = \sum P_n $$

*Equation 5-9*

Another technical obstacle to performing nucleation studies was the fact that until this point the simulation always started with a single cage at the center of the simulation grid. In the LTA model, the three different cages were treated separately. For events that are unfavorable, they would have excessively low probabilities and from the formulae presented here the time interval reported would also be excessively large. So due to the fact that in previous studies the initial grid was always initialised with a single grown cage at the centre of the grid, the case could arise where the one chosen was actually unfavourable for nucleation. What would happen in this case, it would dissolve immediately and would then report such an excessively large time interval. This would pollute the measured times to such an extent that all of the interesting information would be drowned out. To circumvent this, for multi cage systems the initial cage would be selected based upon energetic
favourability. This approach eliminated the effects of the time function being polluted with excessively large time intervals at the initial growth.

Another technical aspect arising from dealing with the continual growth and dissolution of such small structural units; was that the situation can arise where small chains of cages can become split up into small units. These units can then migrate around the simulation grid and become crystals of their own. Due to the way the crystallisation information is recorded, this would introduce errors in the way the nucleation is measured. This effect was eliminated by at iterations during the nucleation phase of growth; a check per iteration is performed for site types associated with small break off clusters. If a break off is detected then it is cleaned up, also if the cluster starts migrating due to repeated growth and dissolution then it is moved back to the centre.

A good candidate crystal was chosen for the initial test study, Figure 5-31 shows the high and equilibrium driving force image for the crystal that was chosen. The chosen crystal has the energies $U_{D4R} = 8 \text{ kcal mol}^{-1}$ $U_{\beta} = 5.4 \text{ kcal mol}^{-1}$ $U_{\alpha} = 1.0 \text{ kcal mol}^{-1}$. The energy levels are shown in Figure 5-32, also marked on the energy level schematic are the driving force conditions behind the two images in Figure 5-31. The region of interest in the nucleation experiments is the region between the high and equilibrium driving force lines. This is where the induction period was observed in previous experiments. It can be seen that all of the energy levels associated with
nucleating new crystals (the ones near the top of each cage type), are all in dissolution mode at the driving forces required to make an induction period. This is an interesting concept, since the induction time is a result of crystal nuclei undergoing many growth and dissolution events before they reach a critical size.

![Energy Level Diagram](image)

Figure 5-32 shows the energy level diagram for $U_{\text{D4R}} = 8 \text{ kcal mol}^{-1}$, $U_\beta = 5.4 \text{ kcal mol}^{-1}$, and $U_\alpha = 1.0 \text{ kcal mol}^{-1}$. The driving forces for high (red) 30.7 kcal mol$^{-1}$ and equilibrium (purple) 15.5 kcal mol$^{-1}$ have been indicated.

By performing successive simulations where the driving force was successively lowered, starting from 31.0 kcal mol$^{-1}$ and decreasing at intervals of 0.1 kcal mol$^{-1}$. The evolution of the induction period can be observed and plotted on a graph Figure 5-33. As can be seen, at higher driving force the induction time (time taken to form a nucleus of critical size) $t_{\text{critical}}$ is virtually absent. Upon decreasing this driving force this induction time increases by an amount that appears to become larger between successive decreases.
Figure 5-33 showing the simulated crystallisation curves for driving forces in the range 31.0-28.5 kcal mol\(^{-1}\) at an interval of 0.1 kcal mol\(^{-1}\). Overlay showing a zoomed in image of the curve, the constant growth and dissolution of cages on the nucleus can be seen.

The simulations record snapshots of the crystallisation along with timestamps in simulation time (not wall clock time). This allows the ability to track at which point the critical nucleus was formed. By plotting this time \(t_{\text{critical}}\) against driving force, it can be seen that the nucleation time increases exponentially as the driving force is dropped Figure 5-34.
This is the current state of this study, however it can be seen that significant foundations have been laid that will enable a significant future study into nucleation processes possible. With sufficient experimental data in the form of a crystallisation curve, it will be possible to start pinning down the driving force conditions within the model an increase the quality of the data produced by this line of models.

As can be seen in Figure 5-33 it is extremely difficult to capture all of the information that is going on during nucleation in a simple plot. Future studies should address this by drawing upon data visualisation principles to make the data more clear and understandable. In addition to this, as can be seen from Figure 5-34 the data points at successively lower $\Delta\mu$ appear to fluctuate from a strict mathematical relationship. This is due to the randomness of the technique being used; it is possible by running multiple experiments with the same conditions to achieve different nucleation times. To circumvent this, a large number of simulations should be performed for each data point to apply a statistical weighting to each one. In future studies, with adequate statistical sampling it should be possible to also investigate the critical nucleus radius in addition to the induction time as a function of driving force.
5.5 References


Chapter 6

Conclusions and Further Work
6.1 Conclusions

Modelling has proved highly successful in my scientific disciplines at explaining concepts and phenomena that are highly complex and/or inaccessible to experimental techniques. This is no exception in crystal growth studies in which a plethora of different techniques have found successful application in aiding scientists to understand the science behind crystal growth. In this study we developed our model further applying it to systems more complex than simple cubic ones, in addition to this the complexity of the model was also increased. As opposed to coarse graining the structure into small cubic building blocks, this model is now capable of handling much more structural detail in the way that the actual cages are used and their respective connectivities with neighbouring cages. Systems with multiple cage geometries can be handled with the new model now. This is a significant step forward in being able to probe the growth rates of certain cages if they are stabilised or destabilised via structure directing agents, this tool would allow experimentalists to target specific morphologies and gain some insight using their chemical knowledge in how to produce those effects.

Modelling Zeolite L

To model zeolite L it was necessary to introduce new methods of enumerating zeolite frameworks. The energetics of the system were dealt with via an attachment energy approach in which the connections in the a,b vs c directions were discriminated by the number of bonds in the attachment. The programs constructed were able to run both 2D and 3D simulations. The 3D simulations facilitated the probing of the crystal habit when manipulating the bidentate and hexadentate attachment energies. The 2D simulations facilitated much larger faces of the crystal to be used as a simulation surface. This gave the ability to probe the effects of manipulating the attachment energies on topographical features.

The model was run in two different regimes, the fixed energy scale regime and the varied energy scale regime. In the fixed case the energy levels could be scaled using a scaling parameter $\mathcal{E}$, however the relative position with respect to each other could not be probed. In the varied energy level regime, the energy levels could be manipulated with respect to each other by modifying the bidentate and hexadentate energies separately.

250
Chapter 6 – Conclusions and Further Work

For the fixed energy regime it was seen that with increasing $E$ the aspect ratio of the crystals increased. In the range of 0-4 kcal.mol$^{-1}$ this increase in length/width of the crystal was fairly modest, however at energies 5+ the aspect increases very sharply. The best correlation with experimental results was found to be in the range of 3-4 kcal.mol$^{-1}$. It was also possible to probe the growth further as a function of driving force. It was suspected that the reason that the lateral spread is so poor in zeolite L, was due to the fact that it is difficult to form bridging linkers across the LTL pore channel. This was shown to be more likely as a feature of the driving force, due to the energetics of nucleating a single cage on the surface is the same as making an a-b attachment to a single terrace. At modest driving force the bridging was more likely to occur when two stabilised terraces grow side by side making the bridge reasonably stable.

In the varied energy regime, it was possible to create a 2D matrix of results when running all permutations of the bidentate and hexadentate attachments energies over a given range. It was possible to produce a much more diverse set of crystal habits in this mode. It was shown that these data sets could be combined with experimental results obtained by experimental colleagues to produce a map of effects upon crystal habit as a function of changing experimental parameters.

This means that the model presented in chapter 3 can not only be used to explain experimental observations produced by Rhea Brent to good effect. But also use in a predictive manner to assist experimental chemists in making informed synthetic choices when designing crystals with specific properties.

Modelling Offretite and Erionite

Taking the model applied to the zeolite L system a little further it was shown that with slight connectivity and energetic modifications the model could be applied to other systems within the ABC-6 family. These studies were conducted mainly to test how extendible the approach was when considering other zeolites when compared with experimental observations.

The models used on both systems were constructed only in 3D form. This was mainly due to the fact that the crystals that could be produced in the simulations were of approximately the same size as those that were being analysed experimentally.
Chapter 6 – Conclusions and Further Work

The same approach of using a fixed and varied energy regimes were applied to both the study of offretite and erionite.

For the fixed energy level regime, it was shown that good correlation was found when the energy scaling parameter was around 3-3.5 kcal.mol\(^{-1}\) for both systems. The crystal aspect ratios obtained and the terrace shapes produced concur with those observed experimentally for the syntheses conducted by Amy Holmes.\(^{2}\)

The varied energy level regime simulations were run to investigate the potential for scope for modifying the erionite and offretite synthesis, much in the same way that Brent did with zeolite L. It was shown that for both systems a similar diverse range of habits as those seen in the zeolite L system could be made by stabilising/destabilising certain attachments. Similar parameters can be changed in the erionite and offretite syntheses as where changed in the zeolite L system, it would be interesting if the effects could be predicted by this model in advance of the syntheses.

These two brief studies form the basis of two further examples of how the effects of stabilising/destabilising certain structural units can affect crystal habit. In addition when these effects are further rationalised with known chemistry, it can be seen that these models could find potential application in synthetic crystal design.

**Modelling the Sodalite Framework**

The modelling of the sodalite framework, in the form of an in depth study of the metal organic framework ZIF-8 and a brief study of the all silica zeolite opened up opportunities to further develop the model methodology. ZIF-8 was a nice system to use for model development since the growth of the crystals could be monitored in real time in-situ via AFM. This is normally very complicated with zeolites due to the difficult synthesis conditions require to create them. This new access to growth information provided a fresh insight into the growth of nanoporous material and presented a new challenge for modelling their growth.

During the course of this study three separate approaches were investigated. The first was born out of the obscurity of moving from modelling using fixed coarse grain units, to a system where each site type uses a different unit of material to complete the cage. This study used the relative stabilities of 6R and 4R present on the surface material to determine site type energies. This was an attempt to investigate the effects
of using different cations and organic agents that often stabilise the ring structures in the framework. The results from this approach showed that it was difficult to produce the evolution of the terrace shapes during the different stages of growth observed experimentally. Even by varying the driving force modes it was difficult to produce all the shapes in a single simulation without changing the energies of the system.

The second model was to approach the problem from a different perspective altogether. This approach centred on working out the energy required to complete a cage by converting T-sites of given connectivity to their bulk configurations. The idea behind this approach was that the effects of stabilising certain surface species could elevate specific sites with respect to others based on their configuration of $Q_0$ and $Q_3$ connected T-sites. Similar trends were observed where the morphology was in agreement with experiment. The difficulty came when trying to fit the growth of the surface at different stages of driving force. This model was extremely complicated with many energy levels that just complicated the picture.

A third approach was then constructed based around the energy that a given site configuration is destabilised with respect to the crystal bulk. This was shown in chapter 4 to be proportional to the number of T-sites that are required to complete a given cage. The energetic approach was totally different to that in all of the models that came before it in the respect that rather than calculating what the attachment energies are; the energies were calculated in reverse with the crystal bulk being the most stable and also the zero energy site. Exactly the same observations as produced in the previous but more complex models were replicated by this much simpler to understand model.

It was proposed that due to the fact that crystal growth in-situ has really only been a real prospect with the advent of room temperature syntheses of nanoporous materials. So a wealth of more complex chemistry is probably behind the observations seen experimentally. Future projects can investigate the incorporation of non-closed cage structures that are relatively stable, these could for instance be pre-calculated using other computational approaches based on molecular dynamics or DFT techniques.

**Modelling Zeolite A**

The simplified model created in the third approach was then applied to the zeolite A system but extended to deal with the multiple cage types separately. This enabled one
to investigate the effects of changing the stability of certain structural units with respect to others. The model used in this study was only for 3D simulations due in part to either having the requirement of multiple layers to deal with the multiple cage types; or the need to perform multiple simulations of single layers that vary terminations. Since computational improvements had allowed larger 3D crystals to be produced, the results obtained were of a reasonable comparison.

In addition to this the code was refactored and optimised for high throughput computation. This adds the bonus of being able to perform mass screening of the energy phase space to gain a greater insight into the processes that govern crystal growth. In addition to this, improvements to the visualiser platform to cope with the overwhelming data sizes included the shift to automated data processing. Incorporation of some high volume data visualisation principles allowed for simplifications in the complexity of the data produced, this enabled a better understanding and location of simulated conditions that agree with experimental observations.

Results showed good correlations, both at the morphological and topological detail on all three of the crystal faces (100), (110) and (111) for the Thompson synthesis. In addition to good correlations the model was able to allude to what the surface terminating stable cages might be. This has been one of the cruxes in the zeolite A system since so many configurations have the same height measurements. The model was also applied to some experimental observations for syntheses that incorporate organic structure directing agents. That particular study is still in the earlier stages, but early results are encouraging.

The model was also modified to investigate the nucleation properties for zeolite A, this was initially noticed by accident in earlier models. This initial result was the start of a very exciting study into the early crystallisation of zeolite A. The model had to be modified slightly to allow for the study to be successful. The approach allows one to measure the size of the critical nucleus and also the time ratio of the induction time:time taken to form a crystal of a given size. The method was fraught with technical obstacles and time constraints, however significant foundations for the proof of concept were laid.
The General Algorithm

The generalised algorithm developed as part of this study will go a long way to making simulations on different/new systems accessible to crystal growth scientists without a hard background in programming. The ease in which new systems can be studied by simply entering the connectivity information into a text file lends to this. In the future this could possibly be done using a graphical user interface or by processing a crystallographic information file (.cif) to make the process even easier.

All of the performance and memory improvements made so far have been incorporated into this version of the program, these enhancements target a multitude of different users, from the scientist wanting to run fast one time simulations to the scientist wanting to run vast screening using supercomputers. The tools developed as part of the visualiser platform further add to the mass screening and processing power of this program.

The chemical science of crystal growth used in this program is extremely easy to modify or alter to incorporate/target crystal growth of different systems. A number of improvements that could extend this program still further have been identified in the next section.

6.2 Future Work

Further work to that presented throughout this thesis can be broadly categorised into two major categories, these are further work to the scientific aspects of the project and further work to the computational aspects of the project.

6.2.1 Further Work (scientific)

In this section, improvements specific to improving the scientific aspects of the project are discussed.

6.2.1.1 Improving the Generality of the Algorithm

To further enhance the number of different systems that the algorithm can be used to study it would be important to include general ways to explore the chemistry of how the following things affect crystal growth

1. Cations and anions, at the current time the effects these have on the crystal growth can only be explored by taking into account their effect upon the
stability of a closed cage unit. It would be more beneficial to include information that would allow specific site configurations with respect to cations/anions might affect a given site's stability, growth/dissolution rates. This information for various configurations could be found by pre-computing energetics for all of the configurations for all site types using a different technique such as DFT.

2. Templates, again these at the moment are taken account of in a sort of macro way by saying that introduction of templates may stabilize/destabilise certain cages by an amount. This does not really allow the elucidation of any real chemistry other than the effect it has upon growth/dissolution rates of certain cages in the presence of such a template and thus usually requires further computation by other techniques to elucidate what the template is doing in terms of chemistry. Once again this could be a pre-computational step performed using another technique such as DFT.

3. Alternating frameworks, introducing alternating framework species would be an interesting study, it could yield a lot of interesting information. Some of the phenomena that could be explored include anisotropic growth where the attachment of units occur at differing rates in different crystallographic directions due to differences in attachment energies of the different framework species such as that seen in ZnPO₄ SOD.³ It would also be possible to investigate the effect on morphology of differing amounts of each of the framework species in solution by introducing different driving forces for each species.

6.2.1.2 Introducing Defects

Defects play a large part in crystal growth, at the moment the crystals in the simulations described in this work are “perfect” they have no defects, this is purely a theoretical simplification and in reality is never the case. Spiral growth is one of the common defects that experimental crystal growth scientists observe, these can drastically change how the surface of the crystal looks, particularly in the low driving force regime. Designing a 2D simulation of a spiral defect would be fairly straightforward and has been done already in simple systems like urea.⁴ The complication comes when adding them into the 3D simulations since currently there is no real
understanding as to why or what causes them to occur, one of the ways it could be done is to introduce a set of dislocation site types and assign them rates of occurrence that can be altered to explore what might happen at differing levels of faulting. Once a screw dislocation occurs the spiral would then grow by creating an endless source of new terrace fronts as explained in the introduction section.

6.2.1.3 Multiphase Systems
At the current state, only single phase systems are being modeled, but it would be possible to allow multiple types of material or even multiple crystals of the same type to grow under the same global driving force conditions. This could be done for example by having multiple sets of energy levels and their corresponding connectivity maps present in the same simulation. This would allow one to probe how competing phases affect the growth of each other, a situation can also be envisaged where the nucleation of the different phases can be investigated.

6.2.1.4 Nucleation
It was observed that if at the start of a given simulation the driving force was lowered sufficiently close enough to the equilibrium, then an induction time is observed whereby the crystal nuclei is undergoing growth and dissolution repeatedly up until a critical size is reached when growth explodes and becomes dominant. This is something that has been investigated as part of this thesis but was unable to be fully completed due to time constraints. Further study alongside practical experiment could allow the actual real world driving force to be correlated with the one in the simulation for the very first time, this would be a very significant step forward in pinning down some of the parameters between theory and experiment.

6.2.2 Further Work (computational)
In this section improvements specific to improving aspects of the project that have their roots mainly in computation.

6.2.2.1 Large Simulations
Development of techniques that allow for ultra large simulations would be beneficial. This is due to the fact that observations from crystals of approximately same size as real crystals would allow for better comparison with real world observations.
Chapter 6 – Conclusions and Further Work

One approach that could be utilized is the partitioning of the simulation grid into chunks. Each chunk would consist of a small 3D section of the main 3D coordinate system. New chunks are created as required to store a growing crystal, these newly created chunks are held in memory so that operations can be performed on coordinates within them at similar speeds to normal array access. Chunks that are deemed to be unlikely to be accessed can be saved into a file of unique id name, and the memory it occupied then released, situations that this condition may arise could include a chunk that contains all grown sites in growth mode or chunks that contain empty sites in dissolution mode. Chunks that contain coordinates that need operations performed on them but that are stored on disk can be simply recalled into memory from file.

It can be seen that using such an approach would allow for the storage of 3D data of a significantly larger size than would be possible with ordinary single 3D array structures. What is more if done correctly then this can be done without a significant penalty on data access times. Some things that should be considered here are the size of the chunk, too large and file parsing times will become significant, too small and the hard disk will be thrashed by the constant storing and retrieval of chunks. Estimated sizes for a space filling cubic system could be in the range of 10-20 μm³ which would be a significant improvement on the 1-2 μm³ currently attainable.

6.2.2.2 Facet Fractions

Since it is now possible to generate vast amounts of qualitative data on a high throughput scale, it would be beneficial to be able to see key information in numerical form instead of relying upon sifting through vast amounts of image based data manually. The introduction of auto facet fraction processing would allow this currently time consuming aspect of data post processing to be performed very quickly. Some preliminary work on the development of such a tool has already been done but this feature needs further work and testing before it can be a stable tool.

The current approach relies on working out what fraction and therefore percentage of the total surface area of a given crystal belongs to which of the three major facet groups (100), (110) and (111). All of the combinations of these facets that a given crystal may exhibit can be plotted on a Venn diagram Figure 6-1. The difficulty in performing this calculation comes from the fact that the data consists of unordered
coordinates due to the nature of how the output file is generated. This prevents any identification of the facets directly.

![Venn diagram showing the different combinations of facets](image)

The vertices at the centre of each of the three facets can be found, in fact these are already found and used in the colouring routines during loading of the input file. Using these points and the fact that the axes of symmetry of the cube, rhombic dodecahedron and the octahedron relate to each other it is then possible to find the area belonging to each facet. This is achieved by bounding the crystal within an imaginary cube such that the 6 vertices at the 3 4-fold axes sit at the centre and in the plane of the 6 faces of the imaginary cube.

Thus by comparing the position of each of the 3 types of vertices associated with the different axes of symmetry of the crystal and the imaginary cube it is possible to find out an approximated area belonging to each facet. If the crystal was a cube (all (100)) then the 3 types of point would line up exactly with those of the imaginary cube. Each of the three cases in Figure 6-1 a,b and c form the limits (pure facets) of how far the configurations of those 3 vertices types deviate from the imaginary cube and these can be expressed in terms of simple geometric equations. If the crystal exhibited a facet configuration that is not all of one type then it may be found in the zones d,e,f or g, the approximate percentages of area can be assigned depending on how close the 3 vertices compare to its respective pure facet crystals.
6.2.2.3 Intelligent Parameter Exploration

Building on the facet fractioning improvement, one could go one step further and utilise one of the many types of artificial intelligence algorithms that exists. One could for example supply desirable output morphology in terms of facet fraction and then the intelligent algorithm attempts to all of the input conditions that satisfy the desired output.

6.2.2.4 Improvements to Usability

To make the software more user friendly such that synthetic crystal growth scientists could use this software with ease the following improvements could be made.

1. Cross platform support, currently the visualiser is writing in C++ and makes numerous calls to the windows API for the GUI. These calls to the windows API could be replaced by using the Nokia Qt platform instead and making calls to the Qt API. Qt is a popular choice amongst companies and developers that choose to maintain versions of their programs across the three major operating systems Windows/Mac and Linux.

2. Introducing more dialog boxes to the program for example that enable the user to easily set up singular or huge sets of simulations, common error checking and tooltips could be introduced to attempt to catch input values that are not of the correct type or would be known to cause problems.

3. Introduce the ability to save high resolution images, at the moment images are saved by reading the OpenGL buffer directly and saving its data to a bitmap, however this means that any image generated is of the same resolution of the screen being used. One way to obtain high resolution images is to simply use an offline buffer of a much larger size and render the scene to that buffer upon saving and then read that buffer to a bitmap, this would achieve greatly increase image size and resolutions.

4. Database support, linking the program with a database for example by introducing MySQL could allow many useful possibilities to be explored. This could include, storing vast numbers of experiments and their respective input parameters in a searchable and accessible way. This would enable data processing tools to take advantage of this easy access and more informative tools can be created. Databases of susceptible structures could be introduced,
where it would be easy to store all of the configuration data both for the configuration of the simulation program and the visualisation program, this data would be easy to distribute and also have users submit their own, this would pave the way for large scale system exploration.

5. Introduce generic support for cages, this can be achieved by swapping out the display lists for vertex arrays, the user would then either provide configuration information or if the previous improvement about databases has been implemented then the information could be stored there. The configuration information would likely be simply the coordinates of all the vertices and a list of indices for the face polygons.

6. Introduce the ability to change the colours used for layers, cages and background window using a colour picker dialog window, and also the give access to change the number of layers coloured on a per facet type basis.

7. Introduce informative overlays with various information, this information could include experimental parameters, configuration of the current view rotation angle, position in 3D space etc.

8. Integration between the simulation and visualiser programs, the benefits of this could be substantial, for example a user wanting to perform a single simulation or many linear simulations (one after the other) could perform it on their own computer and have the benefit of real time data. Whilst the user with access to a supercomputer and requires many thousands of simulations be run in parallel can use the software in client server mode and have the management of all of the input and output files and commands taken care of automatically.

6.3 References

Appendices
Appendix 1

This program has been hard wired for zeolite A, and is the latest version utilising three different cage types and energy of destabilisation. It is written in fortran 90 and compiles with most fortran compilers

```fortran
module variables
  implicit none
  save
  type coordinates
    integer*4:: x, y, z
  end type coordinates
  type position
  type(coordinates), allocatable, dimension():position
end type position
end module variables

module allocatable_array
  use variables
  implicit none
  save
  integer*4, allocatable, dimension(:,:,:,:): grid
  type(position), allocatable, dimension(): site
  type(coordinates), allocatable, dimension(): neighbour_queue
end module allocatable_array

program main
  use allocatable_array
  implicit none
  integer*4: edgex, edgey, edgez, i, j, neighbour_queue_count, loop, loop1, gst, gposi
  integer*4: new_gst, frame_num, del_frame_num, del_final_frame, dmumode
  integer*4: total_frames, initial_frame, final_frame, del_total_frames, del_initial_frame
  integer*8: time, iterations, num, del_num
  integer*4, dimension(172): c
  real*8: energy_scaling_d4r, energy_scaling_sod, energy_scaling_lta, delta_mu
  real*8: s_delta_mu, KT, temperature, q0, r, ranselect, newdmu
  real*8: g_event, s_event_1, s_event_2, high_satu, total_prob, sum_prob, global_time
  real*8, dimension(172): p
  character(len=100): file_name, sol_file, crys_file, mu_file, site_file
  type(coordinates): gs, last_gs

  call random_seed()

  print *,"File name for output data?"
  read '(A)', file_name
  open(1, file=file_name)
  print *,"File name for output the number of solution?"
  read '(A)', sol_file
  open(3, file=sol_file)
  print *,"File name for output the number of crystals?"
  read '(A)', crys_file
  open(4, file=crys_file)
  print *,"File name for output the delta mu [kcal/mol]?"
  read '(A)', mu_file
  open(7, file=mu_file)
  print *,"File name for output the number of site types (113 ~ 192)?"
  read '(A)', site_file
  open(8, file=site_file)
  print *,"Input mode for delta mu."
```

263
Appendices – Appendix 1

read*,  dmumode
print*,"Input starting delta mu."
read*,  s_delta_mu
print*,"Input energy gap (-a+b) [kcal/mol] For First Sphere Coordination."
read*,  energy_scaling_d4r
print*,"Input energy gap c [kcal/mol] to control SSC of TWO sites on the energetic diagram."
read*,  energy_scaling_sod
print*,"Input energy gap c [kcal/mol] to control SSC of TWO sites on the energetic diagram."
read*,  energy_scaling_lta
print*,"Input energy [kcal/mol] for formation of q0."
read*,  q0
print*,"Desired length in x for the simulation (must be an odd number)?"
do
  read*,  edgex
  if (mod(edgex,2)==0) then
    print*,"Failed: Try again."
  else
    exit
  endif
endo
print*,"Desired length in y for the simulation (must be an odd number)?"
do
  read*,  edgey
  if (mod(edgey,2)==0) then
    print*,"Failed: Try again."
  else
    exit
  endif
endo
print*,"Desired length in z for the simulation (must be an odd number)?"
do
  read*,  edgez
  if (mod(edgez,2)==0) then
    print*,"Failed: Try again."
  else
    exit
  endif
endo
print*,"Desired number of iterations?"
read*,  iterations
print*,"Desired length of HIGH supersaturation?"
read*,  high_satu
print*,"Desired temerature (celcius)?"
read*,  temperature
print*,"MOVIE: The number of iterations per frame?"
read*,  num
total_frames = iterations/num
print*,"MOVIE: Initial frame? (default is 1)"
read*,  initial_frame
print*,"MOVIE: Final frame? (default is , total_frames, }
read*,  final_frame
print*,"Delta mu MOVIE: The number of iterations per frame?"
read*,  del_num
del_total_frames = iterations/del_num
print*,"Delta mu MOVIE: Initial frame? (default is 1)"
read*,  del_initial_frame
print*,"Delta mu MOVIE: Final frame? (default is , del_total_frames, }
read*,  del_final_frame
read*,  newdmu
The Array structures are initialised here

```
call memory_initialisation(edgex,edgey,edgez)
```

Any initial sites are setup.

```
call initialise(edgex,edgey,edgez,c)
```

\[
kT = \frac{(6.02E+23*1.38E-23*(temperature+273.15))/4.184/1000}{convert from [kcal/mol]}
\]

\[
s_{\Delta \mu} = s_{\Delta \mu} / kT
\]

\[
\text{newdmu} = \text{newdmu}/kT
\]

```
g_{\text{event}} = 1
s_{\text{event}_1} = \text{high_satu}-1
s_{\text{event}_2} = s_{\text{event}_1}/(1+s_{\Delta \mu})
frame_num = 0
del_frame_num = 0
global_time = 0
neighbour_queue_count = 0
```

Start iteration -MAIN DO LOOP-

```
do time = 1, iterations
```

Delta Mu is determined each iteration based upon decreasing the supersaturation for a 
growth event, and increasing it for dissolution. Thus the parameter \( s_{\text{event}_1} \) is increased 
for growth and decreased for dissolution. The fraction by which \( \text{dmu} \) is changed is 
controlled by \( s_{\text{event}_2} \), which depends on the size of high_satu and the 
starting \( \text{dmu} \) value.

Delta Mu calculation in units of \( kT \), mode 1 allows the supersaturation to equilibrate 
mode 2 holds Delta Mu constant at the starting value, mode 3 holds constant from the 
start until a certain iteration then allows DMU to equilibrate.

```
if (dmumode == 1)then
\[
delta_{\mu} = (s_{\text{event}_1}-s_{\text{event}_2})/s_{\text{event}_2}
\]
elseif (dmumode == 2)then
\[
delta_{\mu} = s_{\Delta \mu}
\]
elseif (dmumode == 3)then
if (time\( \leq 50000000 \))then
\[
delta_{\mu} = s_{\Delta \mu}
\]
else
\[
delta_{\mu} = (s_{\text{event}_1}-s_{\text{event}_2})/s_{\text{event}_2}
\]
endif
elseif (dmumode == 4)then
if (time\( \leq 50000000 \))then
\[
delta_{\mu} = s_{\Delta \mu}
\]
\[
s_{\text{event}_1} = \text{high_satu}-1
\]
elseif (time\( > 50000000 \) and time\( \leq 100000000 \))then
\[
delta_{\mu} = (s_{\text{event}_1}-s_{\text{event}_2})/s_{\text{event}_2}
\]
elseif (time\( > 100000000 \))then
\[
delta_{\mu} = \text{newdmu}
```

endif

Probabilities

```
call probabilities(p,energy_scaling_d4r,energy_scaling_sod,energy_scaling_lta,q0,\delta_{\mu},kT)
```
Global time, adds a fraction to global_time of tau/Q where in this case tau is 1 and Q is the sum of all probabilities.

\[ \text{global_time} = \text{global_time} + \left( \frac{1}{\text{SUM}(p)} \right) \]

Growth or dissolution site is selected by the Monte Carlo method.

This do loop is for calculation of total_prob

\[ \text{total_prob} = 0.0 \]

\[ \text{do loop} = 1, 172 \]

\[ \text{total_prob} = \text{total_prob} + (\text{real}(c(\text{loop})) \times p(\text{loop})) \]

\[ \text{enddo} \]

call random_number(ranselect)

This do loop is for selection of growth or dissolution site type (gst).

\[ \text{sum_prob} = 0.0 \]

\[ \text{proc: do loop1} = 1, 172 \]

\[ \text{sum_prob} = \text{sum_prob} + (\text{real}(c(\text{loop1})) \times p(\text{loop1})) / \text{total_prob} \]

\[ \text{if (sum_prob} \geq \text{ranselect}) \text{ then} \]

\[ \text{call random_number(r)} \]

\[ \text{gst} = \text{loop1} \]

\[ \text{gposi} = \text{int(}\text{real}(c(\text{gst}))) \times \tau + 1.0 \]

\[ \text{gs} = \text{site}(\text{gst})\%\text{position}(\text{gposi}) \]

\[ \text{exit proc} \]

\[ \text{end if} \]

\[ \text{end do proc} \]

Then, switch the site with the last position of the array to delete.

\[ \text{site(}\text{gst})\%\text{position}(\text{gposi}) = \text{site(}\text{gst})\%\text{position}(c(\text{gst})) \]

\[ \text{last_gs} = \text{site(}\text{gst})\%\text{position}(c(\text{gst})) \]

\[ \text{grid}(\text{last_gs}\%x, \text{last_gs}\%y, \text{last_gs}\%z, 2) = \text{gst} \]

\[ \text{grid}(\text{last_gs}\%x, \text{last_gs}\%y, \text{last_gs}\%z, 3) = \text{gposi} \]

\[ \text{c(}\text{gst}) = c(\text{gst})-1 \]

if \( \text{gst} \leq 86 \) then

\[ \text{grid}(\text{gs}\%x, \text{gs}\%y, \text{gs}\%z, 1) = 1 \]

\[ \text{new_gst} = \text{gst} + 86 \]

\[ \text{g_event} = \text{g_event} + 1 \]

\[ \text{s_event}_1 = \text{s_event}_1 - 1 \]

else if \( \text{gst} > 86 \) then

\[ \text{grid}(\text{gs}\%x, \text{gs}\%y, \text{gs}\%z, 1) = 0 \]

\[ \text{new_gst} = \text{gst} - 86 \]

\[ \text{g_event} = \text{g_event} - 1 \]

\[ \text{s_event}_1 = \text{s_event}_1 + 1 \]

endif

if \( \text{size(site(new_gst)\%position)} - c(\text{new_gst}) \leq 10 \) then

\[ \text{call memory_reallocation(new_gst)} \]

endif

Put the new site at the end of the correct site type array

\[ \text{grid}(\text{gs}\%x, \text{gs}\%y, \text{gs}\%z, 2) = \text{new_gst} \]

\[ \text{c(}\text{new_gst}) = c(\text{new_gst}) + 1 \]

\[ \text{grid}(\text{gs}\%x, \text{gs}\%y, \text{gs}\%z, 3) = \text{c(}\text{new_gst}) \]

\[ \text{site(}\text{new_gst})\%\text{position}(c(\text{new_gst})) = \text{gs} \]

Here we are going to process the neighbours of the sites in this queue, the reason for the loop here is simply because as the crystal grows it is possible to complete a cage by growth of all neighbours this is dealt with in one of the subroutines.
Appendices – Appendix 1

!and added to this queue so that the sitetypes are all correct
neighbour_queue(1) = gs
neighbour_queue_count = 1
i = 1
do
   if(i <= neighbour_queue_count) then
      gs = neighbour_queue(i)
call neighbours(gs,c,neighbour_queue_count)
i = i + 1
else
   exit
endif
endo

! Output the grown sites are from the array, site for dissolution (ONE~FIVE).
if (mod(time,num)==0) then
   frame_num = frame_num+1
   if (frame_num>=initial_frame.and.frame_num<=final_frame) then
      do i = 87, 172 ! Dissolution sites ONE~FIVE
         write(8,"(i10)"), c(i)
         do j = 1, c(i)
            if (c(i)<=1) then
               write(1,"(i4,1x,i4,1x,i4)")
site(i)%position(j)
            endif
         enddo
      endo
   endif
endo
write(1,"(i4,1x,i4)") 0, 0
write(8,"(i10)") -1
endif

if (mod(time,del_num)==0) then
   del_frame_num = del_frame_num+1
   if (del_frame_num>=del_initial_frame.and.del_frame_num<=del_final_frame) then
      ! # of solution
      write(3,"(E12.5,1x,F12.2)"), global_time, s_event_1
      ! # of crystals
      write(4,"(E12.5,1x,F12.2)"), global_time, g_event
      ! Supersaturation
      write(7,"(E12.5,1x,F12.6)"), global_time, delta_mu*kT ! delta_mu [kcal/mol]
   endif
endo
! until here -MAIN DO LOOP-
close(1)
close(3)
close(4)
close(7)
close(8)
deallocate(grid,site)
end program

!Initialise the Arrays, the reallocation routine will manage the arrays after this point.
subroutine memory_initialisation(edgex,edgey,edgez)
use allocatable_array
implicit none
subroutine memory_initialisation
!This routine resizes one of the sub arrays as it fills up, at the moment only increases are supported
subroutine memory_reallocation(sitetype)
use allocatable_array
implicit none
type(position), allocatable, dimension(:):: temp
integer*4, intent(in):: sitetype
integer*4:: j, old_size
old_size = SIZE(site(sitetype)%position)
allocate(temp(1))
allocate(temp(1)%position(old_size))
do j = 1,old_size
  temp(1)%position(j) = site(sitetype)%position(j)
end do
deallocate(site(sitetype)%position)
deallocate(site(sitetype)%position((old_size+1000000)))
do j = 1,old_size
  site(sitetype)%position(j) = temp(1)%position(j)
end do
deallocate(temp)
print*, "reallocated sitetype ", sitetype, " to ", old_size+1000000
end subroutine memory_reallocation

! Initialisation
subroutine initialise(edgex,edgey,edgez,c)
use allocatable_array
implicit none
type(coordinates):: gs
integer*4:: neighbour_queue_count
integer*4,intent(in):: edgex,edgey,edgez
integer*4,dimension(172),intent(out):: c

gs%x = (edgex+1)/2; gs%y = (edgey+1)/2; gs%z = (edgez+1)/2
!Initial counters for growth and dissolution sites.
c(1:96) = 0
c(97) = 1
c(98:172) = 0
grid(gs%x,gseq,gs%z,1) = 1
grid(gs%x,gseq,gs%z,2) = 97
grid(gs%x,gseq,gs%z,3) = 1
site(97)%position(1) = gs
neighbour_queue_count = 1
! subroutine initialise
!
! Probability set
!
! subroutine probabilities(p,energy_scaling_d4r,energy_scaling_sod,energy_scaling_lta,q0,delta_mu,kT)
! implicit none
! save
real*8,intent(in):: energy_scaling_d4r, energy_scaling_sod, energy_scaling_lta, q0, delta_mu, kT
real*8,dimension(172),intent(out):: p
!
! Growth site probabilities
!
!D4R
p(1) = exp(-0.5*energy_scaling_d4r*(8*q0)/kT+0.5*delta_mu)
p(2) = exp(-0.5*energy_scaling_d4r*(7*q0)/kT+0.5*delta_mu)
p(3) = exp(-0.5*energy_scaling_d4r*(6*q0)/kT+0.5*delta_mu)
p(4) = exp(-0.5*energy_scaling_d4r*(5*q0)/kT+0.5*delta_mu)
p(5) = exp(-0.5*energy_scaling_d4r*(4*q0)/kT+0.5*delta_mu)
p(6) = exp(-0.5*energy_scaling_d4r*(3*q0)/kT+0.5*delta_mu)
p(7) = exp(-0.5*energy_scaling_d4r*(2*q0)/kT+0.5*delta_mu)
p(8) = exp(-0.5*energy_scaling_d4r*(1*q0)/kT+0.5*delta_mu)
p(9) = 0.0
p(10) = 0.0
!
!SOD
p(11) = exp(-0.5*energy_scaling_sod*(24*q0)/kT+0.5*delta_mu)
p(12) = exp(-0.5*energy_scaling_sod*(23*q0)/kT+0.5*delta_mu)
p(13) = exp(-0.5*energy_scaling_sod*(22*q0)/kT+0.5*delta_mu)
p(14) = exp(-0.5*energy_scaling_sod*(21*q0)/kT+0.5*delta_mu)
p(15) = exp(-0.5*energy_scaling_sod*(20*q0)/kT+0.5*delta_mu)
p(16) = exp(-0.5*energy_scaling_sod*(19*q0)/kT+0.5*delta_mu)
p(17) = exp(-0.5*energy_scaling_sod*(18*q0)/kT+0.5*delta_mu)
p(18) = exp(-0.5*energy_scaling_sod*(17*q0)/kT+0.5*delta_mu)
p(19) = exp(-0.5*energy_scaling_sod*(16*q0)/kT+0.5*delta_mu)
p(20) = exp(-0.5*energy_scaling_sod*(15*q0)/kT+0.5*delta_mu)
p(21) = exp(-0.5*energy_scaling_sod*(14*q0)/kT+0.5*delta_mu)
p(22) = exp(-0.5*energy_scaling_sod*(13*q0)/kT+0.5*delta_mu)
p(23) = exp(-0.5*energy_scaling_sod*(12*q0)/kT+0.5*delta_mu)
p(24) = exp(-0.5*energy_scaling_sod*(11*q0)/kT+0.5*delta_mu)
p(25) = exp(-0.5*energy_scaling_sod*(10*q0)/kT+0.5*delta_mu)
p(26) = exp(-0.5*energy_scaling_sod*(9*q0)/kT+0.5*delta_mu)
p(27) = exp(-0.5*energy_scaling_sod*(8*q0)/kT+0.5*delta_mu)
p(28) = exp(-0.5*energy_scaling_sod*(7*q0)/kT+0.5*delta_mu)
p(29) = exp(-0.5*energy_scaling_sod*(6*q0)/kT+0.5*delta_mu)
p(30) = exp(-0.5*energy_scaling_sod*(5*q0)/kT+0.5*delta_mu)
p(31) = exp(-0.5*energy_scaling_sod*(4*q0)/kT+0.5*delta_mu)
p(32) = exp(-0.5*energy_scaling_sod*(3*q0)/kT+0.5*delta_mu)
p(33) = exp(-0.5*energy_scaling_sod*(2*q0)/kT+0.5*delta_mu)
p(34) = exp(-0.5*energy_scaling_sod*(1*q0)/kT+0.5*delta_mu)
p(35) = 0.0
p(36) = 0.0
p(37) = exp(-0.5 * energy_scaling_lta * (48 * q0 / kT) + 0.5 * delta_mu)
p(38) = exp(-0.5 * energy_scaling_lta * (47 * q0 / kT) + 0.5 * delta_mu)
p(39) = exp(-0.5 * energy_scaling_lta * (46 * q0 / kT) + 0.5 * delta_mu)
p(40) = exp(-0.5 * energy_scaling_lta * (45 * q0 / kT) + 0.5 * delta_mu)
p(41) = exp(-0.5 * energy_scaling_lta * (44 * q0 / kT) + 0.5 * delta_mu)
p(42) = exp(-0.5 * energy_scaling_lta * (43 * q0 / kT) + 0.5 * delta_mu)
p(43) = exp(-0.5 * energy_scaling_lta * (42 * q0 / kT) + 0.5 * delta_mu)
p(44) = exp(-0.5 * energy_scaling_lta * (41 * q0 / kT) + 0.5 * delta_mu)
p(45) = exp(-0.5 * energy_scaling_lta * (40 * q0 / kT) + 0.5 * delta_mu)
p(46) = exp(-0.5 * energy_scaling_lta * (39 * q0 / kT) + 0.5 * delta_mu)
p(47) = exp(-0.5 * energy_scaling_lta * (38 * q0 / kT) + 0.5 * delta_mu)
p(48) = exp(-0.5 * energy_scaling_lta * (37 * q0 / kT) + 0.5 * delta_mu)
p(49) = exp(-0.5 * energy_scaling_lta * (36 * q0 / kT) + 0.5 * delta_mu)
p(50) = exp(-0.5 * energy_scaling_lta * (35 * q0 / kT) + 0.5 * delta_mu)
p(51) = exp(-0.5 * energy_scaling_lta * (34 * q0 / kT) + 0.5 * delta_mu)
p(52) = exp(-0.5 * energy_scaling_lta * (33 * q0 / kT) + 0.5 * delta_mu)
p(53) = exp(-0.5 * energy_scaling_lta * (32 * q0 / kT) + 0.5 * delta_mu)
p(54) = exp(-0.5 * energy_scaling_lta * (31 * q0 / kT) + 0.5 * delta_mu)
p(55) = exp(-0.5 * energy_scaling_lta * (30 * q0 / kT) + 0.5 * delta_mu)
p(56) = exp(-0.5 * energy_scaling_lta * (29 * q0 / kT) + 0.5 * delta_mu)
p(57) = exp(-0.5 * energy_scaling_lta * (28 * q0 / kT) + 0.5 * delta_mu)
p(58) = exp(-0.5 * energy_scaling_lta * (27 * q0 / kT) + 0.5 * delta_mu)
p(59) = exp(-0.5 * energy_scaling_lta * (26 * q0 / kT) + 0.5 * delta_mu)
p(60) = exp(-0.5 * energy_scaling_lta * (25 * q0 / kT) + 0.5 * delta_mu)
p(61) = exp(-0.5 * energy_scaling_lta * (24 * q0 / kT) + 0.5 * delta_mu)
p(62) = exp(-0.5 * energy_scaling_lta * (23 * q0 / kT) + 0.5 * delta_mu)
p(63) = exp(-0.5 * energy_scaling_lta * (22 * q0 / kT) + 0.5 * delta_mu)
p(64) = exp(-0.5 * energy_scaling_lta * (21 * q0 / kT) + 0.5 * delta_mu)
p(65) = exp(-0.5 * energy_scaling_lta * (20 * q0 / kT) + 0.5 * delta_mu)
p(66) = exp(-0.5 * energy_scaling_lta * (19 * q0 / kT) + 0.5 * delta_mu)
p(67) = exp(-0.5 * energy_scaling_lta * (18 * q0 / kT) + 0.5 * delta_mu)
p(68) = exp(-0.5 * energy_scaling_lta * (17 * q0 / kT) + 0.5 * delta_mu)
p(69) = exp(-0.5 * energy_scaling_lta * (16 * q0 / kT) + 0.5 * delta_mu)
p(70) = exp(-0.5 * energy_scaling_lta * (15 * q0 / kT) + 0.5 * delta_mu)
p(71) = exp(-0.5 * energy_scaling_lta * (14 * q0 / kT) + 0.5 * delta_mu)
p(72) = exp(-0.5 * energy_scaling_lta * (13 * q0 / kT) + 0.5 * delta_mu)
p(73) = exp(-0.5 * energy_scaling_lta * (12 * q0 / kT) + 0.5 * delta_mu)
p(74) = exp(-0.5 * energy_scaling_lta * (11 * q0 / kT) + 0.5 * delta_mu)
p(75) = exp(-0.5 * energy_scaling_lta * (10 * q0 / kT) + 0.5 * delta_mu)
p(76) = exp(-0.5 * energy_scaling_lta * (9 * q0 / kT) + 0.5 * delta_mu)
p(77) = exp(-0.5 * energy_scaling_lta * (8 * q0 / kT) + 0.5 * delta_mu)
p(78) = exp(-0.5 * energy_scaling_lta * (7 * q0 / kT) + 0.5 * delta_mu)
p(79) = exp(-0.5 * energy_scaling_lta * (6 * q0 / kT) + 0.5 * delta_mu)
p(80) = exp(-0.5 * energy_scaling_lta * (5 * q0 / kT) + 0.5 * delta_mu)
p(81) = exp(-0.5 * energy_scaling_lta * (4 * q0 / kT) + 0.5 * delta_mu)
p(82) = exp(-0.5 * energy_scaling_lta * (3 * q0 / kT) + 0.5 * delta_mu)
p(83) = exp(-0.5 * energy_scaling_lta * (2 * q0 / kT) + 0.5 * delta_mu)
p(84) = exp(-0.5 * energy_scaling_lta * (1 * q0 / kT) + 0.5 * delta_mu)
p(85) = 0.0
p(86) = 0.0

! Dissolution site probabilities

! D4R
p(87) = exp(0.5 * energy_scaling_d4r * (8 * q0 / kT) + 0.5 * delta_mu)
p(88) = exp(0.5 * energy_scaling_d4r * (7 * q0 / kT) + 0.5 * delta_mu)
p(89) = exp(0.5 * energy_scaling_d4r * (6 * q0 / kT) + 0.5 * delta_mu)
p(90) = exp(0.5 * energy_scaling_d4r * (5 * q0 / kT) + 0.5 * delta_mu)
p(91) = exp(0.5 * energy_scaling_d4r * (4 * q0 / kT) + 0.5 * delta_mu)
p(92) = exp(0.5 * energy_scaling_d4r * (3 * q0 / kT) + 0.5 * delta_mu)
p(93) = exp(0.5*energy_scaling_d4r*(2*q0)/kT-0.5*delta_mu)
p(94) = exp(0.5*energy_scaling_d4r*(1*q0)/kT-0.5*delta_mu)
p(95) = 0.0
p(96) = 0.0

SOD
p(97) = exp(0.5*energy_scaling_sod*(24*q0)/kT-0.5*delta_mu)
p(98) = exp(0.5*energy_scaling_sod*(23*q0)/kT-0.5*delta_mu)
p(99) = exp(0.5*energy_scaling_sod*(22*q0)/kT-0.5*delta_mu)
p(100) = exp(0.5*energy_scaling_sod*(21*q0)/kT-0.5*delta_mu)
p(101) = exp(0.5*energy_scaling_sod*(20*q0)/kT-0.5*delta_mu)
p(102) = exp(0.5*energy_scaling_sod*(19*q0)/kT-0.5*delta_mu)
p(103) = exp(0.5*energy_scaling_sod*(18*q0)/kT-0.5*delta_mu)
p(104) = exp(0.5*energy_scaling_sod*(17*q0)/kT-0.5*delta_mu)
p(105) = exp(0.5*energy_scaling_sod*(16*q0)/kT-0.5*delta_mu)
p(106) = exp(0.5*energy_scaling_sod*(15*q0)/kT-0.5*delta_mu)
p(107) = exp(0.5*energy_scaling_sod*(14*q0)/kT-0.5*delta_mu)
p(108) = exp(0.5*energy_scaling_sod*(13*q0)/kT-0.5*delta_mu)
p(109) = exp(0.5*energy_scaling_sod*(12*q0)/kT-0.5*delta_mu)
p(110) = exp(0.5*energy_scaling_sod*(11*q0)/kT-0.5*delta_mu)
p(111) = exp(0.5*energy_scaling_sod*(10*q0)/kT-0.5*delta_mu)
p(112) = exp(0.5*energy_scaling_sod*(9*q0)/kT-0.5*delta_mu)
p(113) = exp(0.5*energy_scaling_sod*(8*q0)/kT-0.5*delta_mu)
p(114) = exp(0.5*energy_scaling_sod*(7*q0)/kT-0.5*delta_mu)
p(115) = exp(0.5*energy_scaling_sod*(6*q0)/kT-0.5*delta_mu)
p(116) = exp(0.5*energy_scaling_sod*(5*q0)/kT-0.5*delta_mu)
p(117) = exp(0.5*energy_scaling_sod*(4*q0)/kT-0.5*delta_mu)
p(118) = exp(0.5*energy_scaling_sod*(3*q0)/kT-0.5*delta_mu)
p(119) = exp(0.5*energy_scaling_sod*(2*q0)/kT-0.5*delta_mu)
p(120) = exp(0.5*energy_scaling_sod*(1*q0)/kT-0.5*delta_mu)
p(121) = 0.0
p(122) = 0.0

LTA
p(123) = exp(0.5*energy_scaling_lta*(48*q0)/kT-0.5*delta_mu)
p(124) = exp(0.5*energy_scaling_lta*(47*q0)/kT-0.5*delta_mu)
p(125) = exp(0.5*energy_scaling_lta*(46*q0)/kT-0.5*delta_mu)
p(126) = exp(0.5*energy_scaling_lta*(45*q0)/kT-0.5*delta_mu)
p(127) = exp(0.5*energy_scaling_lta*(44*q0)/kT-0.5*delta_mu)
p(128) = exp(0.5*energy_scaling_lta*(43*q0)/kT-0.5*delta_mu)
p(129) = exp(0.5*energy_scaling_lta*(42*q0)/kT-0.5*delta_mu)
p(130) = exp(0.5*energy_scaling_lta*(41*q0)/kT-0.5*delta_mu)
p(131) = exp(0.5*energy_scaling_lta*(40*q0)/kT-0.5*delta_mu)
p(132) = exp(0.5*energy_scaling_lta*(39*q0)/kT-0.5*delta_mu)
p(133) = exp(0.5*energy_scaling_lta*(38*q0)/kT-0.5*delta_mu)
p(134) = exp(0.5*energy_scaling_lta*(37*q0)/kT-0.5*delta_mu)
p(135) = exp(0.5*energy_scaling_lta*(36*q0)/kT-0.5*delta_mu)
p(136) = exp(0.5*energy_scaling_lta*(35*q0)/kT-0.5*delta_mu)
p(137) = exp(0.5*energy_scaling_lta*(34*q0)/kT-0.5*delta_mu)
p(138) = exp(0.5*energy_scaling_lta*(33*q0)/kT-0.5*delta_mu)
p(139) = exp(0.5*energy_scaling_lta*(32*q0)/kT-0.5*delta_mu)
p(140) = exp(0.5*energy_scaling_lta*(31*q0)/kT-0.5*delta_mu)
p(141) = exp(0.5*energy_scaling_lta*(30*q0)/kT-0.5*delta_mu)
p(142) = exp(0.5*energy_scaling_lta*(29*q0)/kT-0.5*delta_mu)
p(143) = exp(0.5*energy_scaling_lta*(28*q0)/kT-0.5*delta_mu)
p(144) = exp(0.5*energy_scaling_lta*(27*q0)/kT-0.5*delta_mu)
p(145) = exp(0.5*energy_scaling_lta*(26*q0)/kT-0.5*delta_mu)
p(146) = exp(0.5*energy_scaling_lta*(25*q0)/kT-0.5*delta_mu)
p(147) = exp(0.5*energy_scaling_lta*(24*q0)/kT-0.5*delta_mu)
p(148) = exp(0.5*energy_scaling_lta*(23*q0)/kT-0.5*delta_mu)
p(149) = exp(0.5*energy_scaling_lta*(22*q0)/kT-0.5*delta_mu)
p(150) = exp(0.5*energy_scaling_lta*(21*q0)/kT-0.5*delta_mu)
Appendices – Appendix 1
p(151)
p(152)
p(153)
p(154)
p(155)
p(156)
p(157)
p(158)
p(159)
p(160)
p(161)
p(162)
p(163)
p(164)
p(165)
p(166)
p(167)
p(168)
p(169)
p(170)
p(171)
p(172)
end subroutine

= exp(0.5*energy_scaling_lta*(20*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(19*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(18*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(17*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(16*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(15*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(14*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(13*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(12*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(11*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(10*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(9*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(8*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(7*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(6*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(5*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(4*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(3*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(2*q0)/kT-0.5*delta_mu)
= exp(0.5*energy_scaling_lta*(1*q0)/kT-0.5*delta_mu)
= 0.0
= 0.0
probabilities

subroutine neighbours(gs,c,neighbour_queue_count)
use allocatable_array
implicit none
type(coordinates),intent(in):: gs
integer*4,intent(inout),dimension(172):: c
integer*4,intent(inout):: neighbour_queue_count
!--- The nearest neighbours of growth site --if(mod(gs%x,2)/=0 .and. mod(gs%y,2)/=0 .and. mod(gs%z,2)/=0) then
call checking_1(gs%x+1,gs%y,gs%z,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y,gs%z,c,neighbour_queue_count)
call checking_1(gs%x,gs%y+1,gs%z,c,neighbour_queue_count)
call checking_1(gs%x,gs%y-1,gs%z,c,neighbour_queue_count)
call checking_1(gs%x,gs%y,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x,gs%y,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y+1,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y+1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y+1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y+1,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y-1,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y-1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y-1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y-1,gs%z+1,c,neighbour_queue_count)
elseif(mod(gs%x,2)==0 .and. mod(gs%y,2)/=0 .and. mod(gs%z,2)/=0) then
call checking_1(gs%x+1,gs%y,gs%z,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y,gs%z,c,neighbour_queue_count)
call checking_1(gs%x,gs%y+1,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x,gs%y+1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x,gs%y-1,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x,gs%y-1,gs%z+1,c,neighbour_queue_count)
elseif(mod(gs%x,2)/=0 .and. mod(gs%y,2)==0 .and. mod(gs%z,2)/=0) then
call checking_1(gs%x,gs%y+1,gs%z,c,neighbour_queue_count)
call checking_1(gs%x,gs%y-1,gs%z,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y,gs%z+1,c,neighbour_queue_count)
call checking_1(gs%x+1,gs%y,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y,gs%z-1,c,neighbour_queue_count)
call checking_1(gs%x-1,gs%y,gs%z+1,c,neighbour_queue_count)

272


elseif(mod(gs%y,2)==0 .and. mod(gs%y,2)==0 .and. mod(gs%z,2)==0) then
    call checking_1(gs%y,gs%y,gs%z+1,c.neighbour_queue_count)
    call checking_1(gs%y,gs%y,gs%z-1,c.neighbour_queue_count)
    call checking_1(gs%y+1,gs%y+1,gs%z,c.neighbour_queue_count)
    call checking_1(gs%y-1,gs%y-1,gs%z,c.neighbour_queue_count)
    call checking_1(gs%y+1,gs%y+1,gs%z,c.neighbour_queue_count)
    call checking_1(gs%y-1,gs%y-1,gs%z,c.neighbour_queue_count)
    endif
end subroutine neighbours

subroutine checking_1(ch_x,ch_y,ch_z,c.neighbour_queue_count)
    use allocatable_array
    implicit none
    integer*4,intent(in):: ch_x, ch_y, ch_z
    integer*4,intent(inout):: neighbour_queue_count
    type(coordinates):: ch, last
    integer*4:: pre_st, pre_posi, st, new_st, cagetype, qcount, i, neighbourcount
    integer*4,intent(inout),dimension(172):: c
    integer*4, dimension(8):: Tsite_d4r
    integer*4, dimension(24):: Tsite_sod
    integer*4, dimension(48):: Tsite_lta
    
    qcount = 0
    neighbourcount = 0
    ch%x = ch_x; ch%y = ch_y; ch%z = ch_z;
    
    !Look up the checked sites information then remove it from is current site type array position
    pre_st = grid(ch%x,ch%y,ch%z,2)
    if (pre_st>1 .and. pre_st<=172) then
        pre_posi = grid(ch%x,ch%y,ch%z,3)
        site(pre_st)%position(pre_posi) = ch
        site(pre_st)%position(pre_posi) = site(pre_st)%position(c(pre_st))
        last = site(pre_st)%position(c(pre_st))
Appendices – Appendix 1

grid(last%x,last%y,last%z,2) = pre_st
grid(last%x,last%y,last%z,3) = pre_posi
c(pre_st) = c(pre_st)-1

endif

if(mod(ch%x,2)==0 .and. mod(ch%y,2)!=0 .and. mod(ch%z,2)!=0) then
cagetype = 1
neighbourcount = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x-1,ch%y,ch%z,1) +
grid(ch%x,ch%y+1,ch%z,1) &
+ grid(ch%x,ch%y-1,ch%z+1,1) + grid(ch%x,ch%y+1,ch%z-1,1) + grid(ch%x,ch%y-1,
ch%z+1,1)

Tsite_d4r(1) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z,1) +
grid(ch%x,ch%y+1,ch%z+1,1)
Tsite_d4r(2) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z,1) +
grid(ch%x,ch%y+1,ch%z+1,1)
Tsite_d4r(3) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z+1,1) +
grid(ch%x,ch%y-1,ch%z+1,1)
Tsite_d4r(4) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z-1,1) +
grid(ch%x,ch%y-1,ch%z,1)
Tsite_d4r(5) = grid(ch%x-1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z,1) +
grid(ch%x,ch%y+1,ch%z+1,1)
Tsite_d4r(6) = grid(ch%x-1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z,1) +
grid(ch%x,ch%y+1,ch%z+1,1)
Tsite_d4r(7) = grid(ch%x-1,ch%y,ch%z,1) + grid(ch%x,ch%y-1,ch%z+1,1) +
grid(ch%x,ch%y-1,ch%z,1)
Tsite_d4r(8) = grid(ch%x-1,ch%y,ch%z,1) + grid(ch%x,ch%y+1,ch%z-1,1) +
grid(ch%x,ch%y-1,ch%z-1,1)
do i = 1,8
if(Tsite_d4r(i)==0)then
qcount = qcount + 1
endif
endo
elseif(mod(ch%x,2)!=0 .and. mod(ch%y,2)==0 .and. mod(ch%z,2)!=0) then
cagetype = 1
neighbourcount = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x,ch%y-1,ch%z,1) +
grid(ch%x+1,ch%y,ch%z+1,1) &
+ grid(ch%x-1,ch%y,ch%z+1,1) + grid(ch%x+1,ch%y,ch%z-1,1) + grid(ch%x-1,
ch%y,ch%z,1)

Tsite_d4r(1) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y,ch%z+1,1) +
grid(ch%x+1,ch%y,ch%z+1,1)
Tsite_d4r(2) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y,ch%z+1,1) +
grid(ch%x,ch%y+1,ch%z+1,1)
Tsite_d4r(3) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y,ch%z+1,1) +
grid(ch%x,ch%y-1,ch%z+1,1)
Tsite_d4r(4) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x-1,ch%y,ch%z+1,1) +
grid(ch%x,1,ch%y,ch%z+1,1)
Tsite_d4r(5) = grid(ch%x,ch%y-1,ch%z,1) + grid(ch%x+1,ch%y,ch%z+1,1) +
grid(ch%x-1,ch%y,ch%z+1,1)
Tsite_d4r(6) = grid(ch%x,ch%y-1,ch%z,1) + grid(ch%x+1,ch%y,ch%z+1,1) +
grid(ch%x,ch%y-1,ch%z+1,1)
Tsite_d4r(7) = grid(ch%x,ch%y-1,ch%z,1) + grid(ch%x+1,ch%y,ch%z-1,1) +
grid(ch%x,ch%y-1,ch%z-1,1)
Tsite_d4r(8) = grid(ch%x,ch%y-1,ch%z,1) + grid(ch%x-1,ch%y,ch%z+1,1) +
grid(ch%x-1,ch%y,ch%z+1,1)
do i = 1,8
if(Tsite_d4r(i)==0)then
qcount = qcount + 1
endif
endo
enddo
elseif(mod(ch%x,2)=0 .and. mod(ch%y,2)=0 .and. mod(ch%z,2)=0) then
  catetype = 1
  neighbourcount = grid(ch%x,ch%y,ch%z+1,1) + grid(ch%x,ch%y,ch%z-1,1) + 
    grid(ch%x+1,ch%y+1,ch%z) & + grid(ch%x-1,ch%y+1,ch%z) + grid(ch%x+1,ch%y-1,ch%z) + 
    grid(ch%x-1,ch%y-1,ch%z) + grid(ch%x-1,ch%y+1,ch%z) + grid(ch%x+1,ch%y-1,ch%z)

  Tsite_d4r(1) = grid(ch%x,ch%y,ch%z+1,1) + grid(ch%x+1,ch%y+1,ch%z,1) + 
    grid(ch%x,ch%y+1,ch%z,1)
  Tsite_d4r(2) = grid(ch%x,ch%y,ch%z+1,1) + grid(ch%x+1,ch%y+1,ch%z,1) + 
    grid(ch%x+1,ch%y-1,ch%z,1)
  Tsite_d4r(3) = grid(ch%x,ch%y,ch%z+1,1) + grid(ch%x+1,ch%y-1,ch%z,1) + 
    grid(ch%x-1,ch%y-1,ch%z)
  Tsite_d4r(4) = grid(ch%x,ch%y,ch%z+1,1) + grid(ch%x-1,ch%y+1,ch%z,1) + 
    grid(ch%x-1,ch%y-1,ch%z,1)
  Tsite_d4r(5) = grid(ch%x,ch%y,ch%z-1,1) + grid(ch%x+1,ch%y+1,ch%z,1) + 
    grid(ch%x-1,ch%y+1,ch%z,1)
  Tsite_d4r(6) = grid(ch%x,ch%y,ch%z-1,1) + grid(ch%x+1,ch%y+1,ch%z,1) + 
    grid(ch%x+1,ch%y-1,ch%z,1)
  Tsite_d4r(7) = grid(ch%x,ch%y,ch%z-1,1) + grid(ch%x+1,ch%y-1,ch%z,1) + 
    grid(ch%x-1,ch%y-1,ch%z,1)
  Tsite_d4r(8) = grid(ch%x,ch%y,ch%z-1,1) + grid(ch%x-1,ch%y+1,ch%z,1) + 
    grid(ch%x-1,ch%y-1,ch%z,1)

  do i = 1,8
    if(Tsite_d4r(i)==0) then
      qcount = qcount + 1
    enddo
  endif
endif
else(mod(ch%x,2)=0 .and. mod(ch%y,2)=0 .and. mod(ch%z,2)=0) then
  catetype = 2
  neighbourcount = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x-1,ch%y,ch%z,1) + 
    grid(ch%x,ch%y,ch%z,1) & + grid(ch%x,ch%y-1,ch%z,1) + grid(ch%x,ch%y,ch%z+1,1) + 
    grid(ch%x,ch%y+1,ch%z,1) & + grid(ch%x,ch%y-1,ch%z+1,1) + grid(ch%x,ch%y,ch%z-1,1) + 
    grid(ch%x,ch%y+1,ch%z-1,1) + grid(ch%x-1,ch%y+1,ch%z,1) + 
    grid(ch%x+1,ch%y+1,ch%z,1) + grid(ch%x-1,ch%y-1,ch%z,1) + 
    grid(ch%x+1,ch%y-1,ch%z,1) + grid(ch%x-1,ch%y-1,ch%z,1) + grid(ch%x-1,ch%y+1,ch%z,1) + 
    grid(ch%x-1,ch%y-1,ch%z,1)

  Tsite_sod(1) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + 
    grid(ch%x-1,ch%y+1,ch%z+1,1)
  Tsite_sod(2) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + 
    grid(ch%x+1,ch%y+1,ch%z-1,1)
  Tsite_sod(3) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z-1,1) + 
    grid(ch%x-1,ch%y+1,ch%z,1)
  Tsite_sod(4) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x-1,ch%y+1,ch%z+1,1) + 
    grid(ch%x-1,ch%y+1,ch%z-1,1)
  Tsite_sod(5) = grid(ch%x,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + 
    grid(ch%x-1,ch%y+1,ch%z-1,1)
  Tsite_sod(6) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x-1,ch%y,ch%z+1,1) + 
    grid(ch%x+1,ch%y,ch%z+1,1)
  Tsite_sod(7) = grid(ch%x+1,ch%y,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + 
    grid(ch%x+1,ch%y+1,ch%z-1,1)
  Tsite_sod(8) = grid(ch%x-1,ch%y,ch%z,1) + grid(ch%x-1,ch%y+1,ch%z+1,1) + 
    grid(ch%x-1,ch%y+1,ch%z,1)
  Tsite_sod(9) = grid(ch%x,ch%y,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z,1) + 
    grid(ch%x+1,ch%y+1,ch%z+1,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + grid(ch%x+1,ch%y+1,ch%z+1,1)
  Tsite_sod(10) = grid(ch%x+1,ch%y+1,ch%z,1) + grid(ch%x+1,ch%y+1,ch%z+1,1) + 
    grid(ch%x+1,ch%y+1,ch%z+1,1)
Appendices – Appendix 1

\[
T_{site\ sol}(11) = grid(ch\%x+1, ch\%y, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x+1, ch\%y-1, ch\%z-1, 1)
\]
\[
T_{site\ sol}(12) = grid(ch\%x, ch\%y, ch\%z-1, 1) + grid(ch\%x+1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x+1, ch\%y-1, ch\%z-1, 1)
\]
\[
T_{site\ sol}(13) = grid(ch\%x, ch\%y, ch\%z-1, 1) + grid(ch\%x+1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x+1, ch\%y-1, ch\%z-1, 1)
\]
\[
T_{site\ sol}(14) = grid(ch\%x-1, ch\%y, ch\%z, 1) + grid(ch\%x-1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x-1, ch\%y-1, ch\%z-1, 1)
\]
\[
T_{site\ sol}(15) = grid(ch\%x-1, ch\%y, ch\%z, 1) + grid(ch\%x-1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x-1, ch\%y-1, ch\%z-1, 1)
\]
\[
\text{do } i = 1, 24
\]
\[
\text{if(Tsite\ sol(i)==0) then}
\]
\[
qcount = qcount + 1
\]
\[
\text{endif}
\]
\[
\text{elseif(mod(ch\%x,2)==0 .and. mod(ch\%y,2)==0 .and. mod(ch\%z,2)==0) then}
\]
\[
\text{cagetype = 3}
\]
\[
\text{neighbourcount = grid(ch\%x+2, ch\%y, ch\%z, 1) + grid(ch\%x-2, ch\%y, ch\%z, 1) + grid(ch\%x, ch\%y+2, ch\%z, 1) &}
\]
\[
+ grid(ch\%x, ch\%y-2, ch\%z, 1) + grid(ch\%x, ch\%y, ch\%z+2, 1) + grid(ch\%x, ch\%y, ch\%z-2, 1)
\]
\[
\text{&}
\]
\[
+ grid(ch\%x+1, ch\%y+1, ch\%z+1, 1) + grid(ch\%x+1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x+1, ch\%y-1, ch\%z+1, 1) &}
\]
\[
+ grid(ch\%x+1, ch\%y-1, ch\%z-1, 1) + grid(ch\%x-1, ch\%y+1, ch\%z+1, 1) &}
\]
\[
+ grid(ch\%x-1, ch\%y+1, ch\%z-1, 1) + grid(ch\%x-1, ch\%y-1, ch\%z+1, 1) &}
\]
\[
+ grid(ch\%x-1, ch\%y-1, ch\%z-1, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1) &}
\]
\[
+ grid(ch\%x+1, ch\%y+1, ch\%z, 1) + grid(ch\%x+1, ch\%y-1, ch\%z, 1) &}
\]
\[
+ grid(ch\%x+1, ch\%y-1, ch\%z, 1) + grid(ch\%x-1, ch\%y+1, ch\%z, 1) &}
\]
\[
+ grid(ch\%x-1, ch\%y+1, ch\%z, 1) + grid(ch\%x-1, ch\%y-1, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1) + grid(ch\%x+1, ch\%y-1, ch\%z, 1) + grid(ch\%x+1, ch\%y-1, ch\%z, 1) + grid(ch\%x-1, ch\%y+1, ch\%z, 1) + grid(ch\%x-1, ch\%y-1, ch\%z, 1)
\]
\[
T_{site\ lat}(1) = grid(ch\%x, ch\%y+2, ch\%z, 1) + grid(ch\%x, ch\%y+1, ch\%z+1, 1) + grid(ch\%x, ch\%y+1, ch\%z+1, 1)
\]
\[
T_{site\ lat}(2) = grid(ch\%x, ch\%y+2, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1)
\]
\[
T_{site\ lat}(3) = grid(ch\%x, ch\%y+2, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1)
\]
\[
T_{site\ lat}(4) = grid(ch\%x, ch\%y+2, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1) + grid(ch\%x+1, ch\%y+1, ch\%z, 1)
\]
\[
T_{site\ lat}(5) = grid(ch\%x, ch\%y+2, ch\%z, 1) + grid(ch\%x, ch\%y+1, ch\%z-1, 1) + grid(ch\%x-1, ch\%y+1, ch\%z-1, 1)
\]

276
Tsite_lta(6) = grid(ch^x, ch^y + 2, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(7) = grid(ch^x, ch^y + 2, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(8) = grid(ch^x, ch^y + 2, ch^z, 1) + grid(ch^x, ch^y + 1, ch^z + 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(9) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x, ch^y + 1, ch^z + 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(10) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y + 1, ch^z, 1) + grid(ch^x + 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(11) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y + 1, ch^z, 1) + grid(ch^x + 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(12) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x, ch^y + 1, ch^z - 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(13) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x, ch^y + 1, ch^z - 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(14) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(15) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(16) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x, ch^y + 1, ch^z + 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(17) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x + 1, ch^y, ch^z + 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(18) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y, ch^z + 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(19) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y, ch^z + 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(20) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x + 1, ch^y, ch^z - 1, 1) + grid(ch^x + 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(21) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x - 1, ch^y, ch^z - 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(22) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y, ch^z + 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(23) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y, ch^z + 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z - 1, 1)
Tsite_lta(24) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x - 1, ch^y, ch^z + 1, 1) + grid(ch^x - 1, ch^y + 1, ch^z + 1, 1)
Tsite_lta(25) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x + 1, ch^y, ch^z + 1, 1) + grid(ch^x + 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(26) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y, ch^z + 1, 1) + grid(ch^x + 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(27) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y, ch^z - 1, 1) + grid(ch^x + 1, ch^y - 1, ch^z - 1, 1)
Tsite_lta(28) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x + 1, ch^y, ch^z - 1, 1) + grid(ch^x + 1, ch^y - 1, ch^z - 1, 1)
Tsite_lta(29) = grid(ch^x, ch^y, ch^z - 2, 1) + grid(ch^x - 1, ch^y, ch^z - 1, 1) + grid(ch^x - 1, ch^y - 1, ch^z - 1, 1)
Tsite_lta(30) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y, ch^z - 1, 1) + grid(ch^x - 1, ch^y - 1, ch^z - 1, 1)
Tsite_lta(31) = grid(ch^x - 2, ch^y, ch^z, 1) + grid(ch^x - 1, ch^y, ch^z + 1, 1) + grid(ch^x - 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(32) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x - 1, ch^y, ch^z + 1, 1) + grid(ch^x - 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(33) = grid(ch^x, ch^y, ch^z + 2, 1) + grid(ch^x - 1, ch^y - 1, ch^z + 1, 1) + grid(ch^x - 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(34) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y - 1, ch^z, 1) + grid(ch^x + 1, ch^y - 1, ch^z + 1, 1)
Tsite_lta(35) = grid(ch^x + 2, ch^y, ch^z, 1) + grid(ch^x + 1, ch^y - 1, ch^z, 1) + grid(ch^x + 1, ch^y - 1, ch^z + 1, 1)
If we don’t do this then they cause voids inside the crystal when rendering which this is indirect growth and therefore the main arrays are updated with grown status.

If the first layer of the main array is 1 then that site is grown and thus probably call do Tsite_lta

if(Tsite_lta(i)==0)then qcount = qcount + 1 endif
donothing
call sitetype_lookup(cagetype,qcount,neighbourcount,st)

! If the first layer of the main array is 1 then that site is grown and thus probably just got its connectivity updated we add the number of site to the new site to! get the dissolution site type. If it is zero and also qcount is zero then in a fused! cage system all of the T-sites have been contributed by neighbouring cages, ! this is indirect growth and therefore the main arrays are updated with grown status. ! If we don’t do this then they cause voids inside the crystal when rendering which ! has a good chance of screwing the colouring routines. Failing both of those ! scenario’s then we arrive at an ordinary growth site

if (grid(ch%x,ch%y,ch%z,1)==1) then new_st = st+86

elseif(grid(ch%x,ch%y,ch%z,1)==0.and.qcount==0) then grid(ch%x,ch%y,ch%z,1) = 1 new_st = st+86 neighbour_queue_count = neighbour_queue_count + 1 neighbour_queue(neighbour_queue_count) = ch

else
    new_st = st
endif

! Check if we need more space to start saving new sites, we always! need a buffer zone of 10 just incase bad things happen.
if((size(site(new_st)%position)-c(new_st)) <= 10) then call memory_reallocation(new_st) endif
grid(ch%x,ch%y,ch%z,2) = new_st

! If the new site type fails this test then it can’t be right so
! it is set to 0 so it can be like an initial element
if (new_st<1 .or. new_st>172) then
  grid(ch%x,ch%y,ch%z,3) = 0
else
  ! Otherwise if it passes then things are all good and the normal
  ! addition to the Counters and arrays ensues here.
c(new_st) = c(new_st)+1
  grid(ch%x,ch%y,ch%z,3) = c(new_st)
site(new_st)%position(c(new_st)) = ch
endif

! This if statement is only intended for advanced debugging, if it starts outputting then there is
something wrong.
if(new_st<1 .or. new_st>172)then
  print*, 'Site Type = ',new_st, 'st = ', st,"qcount = ", qcount, "neighbourcount = ", neighbourcount, "cagetype = ", cagetype
  print*, 'x = ",ch_x, "y = ", ch_y, "z = ",ch_z
end if subroutine checking_1

subroutine sitetype_lookup(cagetype,qcount,neighbourcount,st)
imPLICIT none
integer*4, intent(in):: cagetype,qcount,neighbourcount
integer*4, intent(out):: st
if(cagetype == 1) then
  if(qcount==8) st = 1
  if(qcount==7) st = 2
  if(qcount==6) st = 3
  if(qcount==5) st = 4
  if(qcount==4) st = 5
  if(qcount==3) st = 6
  if(qcount==2) st = 7
  if(qcount==1) st = 8
  if(qcount==0) st = 9
  if(qcount==0.and.neighbourcount==6) st = 10
endif
if(cagetype == 2) then
  if(qcount==24) st = 11
  if(qcount==23) st = 12
  if(qcount==22) st = 13
  if(qcount==21) st = 14
  if(qcount==20) st = 15
  if(qcount==19) st = 16
  if(qcount==18) st = 17
  if(qcount==17) st = 18
  if(qcount==16) st = 19
  if(qcount==15) st = 20
  if(qcount==14) st = 21
  if(qcount==13) st = 22
  if(qcount==12) st = 23
  if(qcount==11) st = 24
  if(qcount==10) st = 25
  if(qcount==9) st = 26
endif
Appendices – Appendix 1

if(qcount==8) st = 27
if(qcount==7) st = 28
if(qcount==6) st = 29
if(qcount==5) st = 30
if(qcount==4) st = 31
if(qcount==3) st = 32
if(qcount==2) st = 33
if(qcount==1) st = 34
if(qcount==0) st = 35
if(qcount==0 .and. neighbourcount==14) st = 36
endif
if(cagetype == 3) then
  if(qcount==48) st = 37
  if(qcount==47) st = 38
  if(qcount==46) st = 39
  if(qcount==45) st = 40
  if(qcount==44) st = 41
  if(qcount==43) st = 42
  if(qcount==42) st = 43
  if(qcount==41) st = 44
  if(qcount==40) st = 45
  if(qcount==39) st = 46
  if(qcount==38) st = 47
  if(qcount==37) st = 48
  if(qcount==36) st = 49
  if(qcount==35) st = 50
  if(qcount==34) st = 51
  if(qcount==33) st = 52
  if(qcount==32) st = 53
  if(qcount==31) st = 54
  if(qcount==30) st = 55
  if(qcount==29) st = 56
  if(qcount==28) st = 57
  if(qcount==27) st = 58
  if(qcount==26) st = 59
  if(qcount==25) st = 60
  if(qcount==24) st = 61
  if(qcount==23) st = 62
  if(qcount==22) st = 63
  if(qcount==21) st = 64
  if(qcount==20) st = 65
  if(qcount==19) st = 66
  if(qcount==18) st = 67
  if(qcount==17) st = 68
  if(qcount==16) st = 69
  if(qcount==15) st = 70
  if(qcount==14) st = 71
  if(qcount==13) st = 72
  if(qcount==12) st = 73
  if(qcount==11) st = 74
  if(qcount==10) st = 75
  if(qcount==9) st = 76
  if(qcount==8) st = 77
  if(qcount==7) st = 78
  if(qcount==6) st = 79
  if(qcount==5) st = 80
  if(qcount==4) st = 81
  if(qcount==3) st = 82
  if(qcount==2) st = 83
  if(qcount==1) st = 84
if(qcount==0) st = 85
if(qcount==0 .and. neighbourcount==26) st = 86
endif
end subroutine sitetype_lookup
Appendices – Appendix 2

Appendix 2

This program is the generalised algorithm, this program takes the connectivity information from a file. It is written in fortran 90 and compiles with most fortran compilers.

module variables
    implicit none
    save
    type coordinates
        integer(kind=3):: x, y, z
    end type coordinates
    type positions
        type(coordinates), allocatable, dimension(:):: position
    end type positions
    type networks
        integer(kind=3), allocatable, dimension(:):: net
    end type networks
end module variables

module storage_structures
    use variables
    implicit none
    save
    integer(kind=3), allocatable, dimension(:):: c
    integer(kind=3), allocatable, dimension(:,:,:):: grid
    integer(kind=3), allocatable, dimension(:,:):: information
    real(kind=2), allocatable, dimension(:):: p
    real(kind=2), allocatable, dimension(:):: int_energies
    real(kind=2), allocatable, dimension(:):: energy_scalars
    type(positions), allocatable, dimension(:):: site
    type(coordinates), allocatable, dimension(:):: neighbour_queue
    type(networks), allocatable, dimension(:):: connections, tsites
end module storage_structures

program main
    use storage_structures
    implicit none
    integer(kind=3):: edgex, edgey, edgez, i, j, k, neighbour_queue_count, loop, loop1, gst, gposi
    integer(kind=3):: new_gst, frame_num, del_frame_num, dmumode, cagetypes, gstc, gstc_tot
    integer(kind=3):: total_frames, initial_frame, final_frame, del_total_frames, tmp, n_mode
    real(kind=2):: delta_mu, kT, temperature, r, ranselect, newdmu, global_time
    real(kind=2):: g_event, s_event_1, s_event_2, high_satu, total_prob, sum_prob
    character(len=100):: input_file, file_name, sol_file, crys_file, mu_file, site_file
    type(coordinates):: gs, last_gs

    call random_seed()

    print*,"File name of file with system info?"
    read '(A)', input_file
    print*,"File name for output data?"
    read '(A)', file_name

282
open(2, file=file_name)
print*("File name for output the number of solution?")
read '(A)', sol_file
open(3, file=sol_file)
print*("File name for output the number of crystals?")
read '(A)', crys_file
open(4, file=crys_file)
print*("File name for output the delta mu [kcal/mol]?"
read '(A)', mu_file
open(7, file=mu_file)
print*("File name for output the number of site types (113 ~ 192)?"
read '(A)', site_file
open(8, file=site_file)
print*("Desired length in x for the simulation (must be an odd number)?")
do
read*, edgex
if (mod(edgex,2)==0) then
   print*, "Failed: Enter an odd number for the x coordinate."
else
   exit
endif
enddo
print*("Desired length in y for the simulation (must be an odd number)?")
do
read*, edgey
if (mod(edgey,2)==0) then
   print*, "Failed: Enter an odd number for the y coordinate."
else
   exit
endif
enddo
print*("Desired length in z for the simulation (must be an odd number)?")
do
read*, edgez
if (mod(edgez,2)==0) then
   print*, "Failed: Enter an odd number for the z coordinate."
else
   exit
endif
enddo
print*("Desired number of iterations?")
read*, iterations
print*("Desired length of HIGH supersaturation?")
read*, high_satu
print*("Desired temerature (celcius)?")
read*, temperature
print*, "MOVIE: The number of iterations per frame?"
read*, num
total_frames = iterations/num
print*, "MOVIE: Initial frame? (default is 1)"
read*, initial_frame
print*, "MOVIE: Final frame? (default is", total_frames, ")"
read*, final_frame
print*, "Delta mu MOVIE: The number of iterations per frame?"
read*, del_num
del_total_frames = iterations/del_num
print*, "Delta mu MOVIE: Initial frame? (default is 1)"
read*, del_initial_frame
print*, "Delta mu MOVIE: Final frame? (default is", del_total_frames, ")"
read*, del_final_frame
print*, "nucleation mode 1/0 for on/off"
read*, n_mode
print*, "Input mode for delta mu."
read*, dmumode
print*, "Input starting delta mu."
read*, s_delta_mu

! system specific setup of data structures
call setup(edgex,edgey,edgez,input_file,cagetypes,gstc,gstc_tot)

if(dmumode==3) then
  print*, "Enter the number of iterations the constant dmu stage should last"
  read*, length1
endif
if(dmumode==4) then
  print*, "Enter the number of iterations the first constant dmu phase should last"
  read*, length1
  print*, "Enter the iteration number that the second contact dmu phase should kick in"
  read*, length2
  print*, "Enter dmu value for second constant dmu stage"
  read*, newdmu
endif
if(n_mode==0) then
  print*, "Which cage type should the starting one be?"
  read*, starting_cage
endif
if(n_mode==1) then
  print*, "how large should the crystal be allowed to go"
  read*, crystalsize
endif
do i = 1, cagetypes
  print*, "Enter energy parameter for cage ", i
  read*, energy_scalars(i)
endo

kT = (6.02E+23*1.38E-23*(temperature+273.15))/4.184/1000

! initialise the first site etc
call initialise(edgex,edgey,edgez,kT,cagetypes,gstc,gstc_tot,n_mode,starting_cage)

s_delta_mu = s_delta_mu / kT !convert from [kcal/mol]
newdmu = newdmu/kT

g_event = 1
s_event_1 = high_satu-1
s_event_2 = s_event_1/(1+s_delta_mu)
frame_num = 0
del_frame_num = 0
global_time = 0
neighbour_queue_count = 0

! MAIN LOOP
do time = 1, iterations

!Delta Mu is determined each iteration based upon decreasing the supersaturation for a
growth event, and increasing it for dissolution. Thus the parameter s_event_1 is increased
!for growth and decreased for dissolution. The fraction by which dmu is changed is
!controlled by s_event_2, which depends on the size of high_satu and the
!starting dmu value.
Delta Mu calculation in units of kT, mode 1 allows the supersaturation to equilibrate.
Mode 2 holds Delta Mu constant at the starting value, mode 3 holds constant from the start until a certain iteration then allows DMU to equilibrate. Mode 4 allows constant DMU followed by an equilibration stage followed by another constant DMU phase.

```plaintext
if (dmumode == 1) then
    delta_mu = (s_event_1 - s_event_2) / s_event_2
elseif (dmumode == 2) then
    delta_mu = s_delta_mu
    s_event_1 = high_satu
elseif (dmumode == 3) then
    if (time <= length1) then
        delta_mu = s_delta_mu
        s_event_1 = high_satu
    else
        delta_mu = (s_event_1 - s_event_2) / s_event_2
    endif
elseif (dmumode == 4) then
    if (time <= length1) then
        delta_mu = s_delta_mu
        s_event_1 = high_satu
    elseif (time > length1 .and. time <= length2) then
        delta_mu = (s_event_1 - s_event_2) / s_event_2
    elseif (time > length2) then
        delta_mu = newdmu
    endif
endif
```

Probabilities
```
call probabilities(delta_mu, cagetypes, gstc)
```

If nucleation mode is on then we need to call the nucleation subroutine
```
if (n_mode == 1) call nucleation(edgex, edgey, edgez, g_event, cagetypes, gstc, gstc_tot)
```

Global time, adds a fraction to global_time of tau/Q where in this case tau is 1 and Q is the sum of all probabilities.
```
global_time = global_time + (1/SUM(p))
```

Growth or dissolution site is selected by the Monte Carlo method. ---
```
This do loop is for calculation of total_prob
```
```
total_prob = 0.0
do loop = 1, gstc_tot
    total_prob = total_prob + (real(c(loop)) * p(loop))
endo
call random_number(ranselect)
```

This do loop is for selection of growth or dissolution site type (gst).
```
sum_prob = 0.0
proc: do loop1 = 1, gstc_tot
    sum_prob = sum_prob + (real(c(loop1)) * p(loop1) / total_prob)
    if (sum_prob >= ranselect) then
        call random_number(r)
        gst = loop1
        gposi = int(real(c(gst)) * r + 1.0)
        gs = site(gst) % position(gposi)
        exit
    endif
endo proc
```
! perform deletion of grown site from old site type

\[
\text{site(}gst\text{)}%\text{position}(gposi) = \text{site(}gst\text{)}%\text{position}(c(\text{gst}))
\]

last_gs = site(\text{gst})%\text{position}(c(\text{gst}))

\[
\text{grid(}last\_gs\text{x, last}\_gs\text{y, last}\_gs\text{z, 2)} = \text{gst}
\]

\[
\text{grid(}last\_gs\text{x, last}\_gs\text{y, last}\_gs\text{z, 3)} = \text{gposi}
\]

c(\text{gst}) = c(\text{gst})-1

! Here the new site type is assigned based on growth or dissolution
! the number of units in the crystal and solution is updated
! Here we check that the site type array has enough space, if not make more
if ((\text{size}(\text{site(new}\_\text{gst})%\text{position})-c(\text{new}\_\text{gst})) <= 10) then
call memory_reallocation(new\_\text{gst})
endif

! Put the new site at the end of the correct site type array

\[
\text{grid(}gs\text{x, gs}\_y, gs\text{z, 2)} = \text{new}\_\text{gst}
\]

c(new\_\text{gst}) = c(new\_\text{gst})+1

\[
\text{grid(}gs\text{x, gs}\_y, gs\text{z, 3)} = c(\text{new}\_\text{gst})
\]

site(new\_\text{gst})%\text{position}(c(new\_\text{gst})) = gs

! Here we are going to process the neighbours of the sites in this queue, the reason
! for the loop here is simply because as the crystal grows it is possible to complete
! a cage by growth of all neighbours this is dealt with in one of the subroutines
! and added to this queue so that the sitetypes are all correct

\[
\text{neighbour}\_\text{queue(1)} = gs
\]

\[
\text{neighbour}\_\text{queue}\_\text{count} = 1
\]

\[
i = 1
\]

do
if(i <= neighbour\_queue\_count) then

\[
gs = \text{neighbour}\_\text{queue(i)}
\]
call neighbours(gs, neighbour\_queue\_count, cagetypes, gstc, gstc\_tot)
i = i + 1
else
exit
endif
endo

! if nucleation mode is off then normal output
if(n\_mode==0) then

! Output the grown surface sites from the sites array, site for dissolution.
if ( \text{mod}(\text{time},\text{num})==0) then

frame\_num = frame\_num+1

286
if (frame_num>=initial_frame and frame_num<=final_frame) then
  do i = gstc+1, gstc_tot
    write(8,"(i10)") c(i)
    tmp = 0
    do k = 1, cagetypes
      if(i==(gstc+information(k,2)-1)) tmp = 1
    enddo
    if(tmp==0) then
      do j = 1, c(i)
        if (c(i)>=1) then
          write(2,"(i4,1x,i4,1x,i4)")
          site(i)%position(j)
        endif
      enddo
    endif
  enddo
endif
enddo
write(2,"(i4,1x,i4)") 0, 0
write(8,"(i10)") -1
endif
if (mod(time,del_num)==0) then
  del_frame_num = del_frame_num+1
endif
if (del_frame_num>=del_initial_frame and del_frame_num<=del_final_frame) then
  ! # of solution
  write(3,"(E12.5,1x,F12.2)") global_time, s_event_1
  ! # of crystals
  write(4,"(E12.5,1x,F12.2)") global_time, g_event
  ! Supersaturation
  write(7,"(E12.5,1x,F12.6)") global_time, delta_mu*kT
endif
enddo
! if nucleation mode is on then we want limitless output at given intervals
elseif(n_mode==1) then
  ! Output the grown surface sites from the sites array, site for dissolution.
  if (mod(time,num)==0) then
    do i = gstc+1, gstc_tot
      write(8,"(i10)") c(i)
      tmp = 0
      do k = 1, cagetypes
        if(i==(gstc+information(k,2)-1)) tmp = 1
      enddo
      if(tmp==0) then
        do j = 1, c(i)
          if (c(i)>=1) then
            write(2,"(i4,1x,i4,1x,i4)")
            site(i)%position(j)
          endif
        enddo
      endif
    enddo
  endif
endif
write(2,"(i4,1x,i4)") 0, 0
write(8,"(i10)") -1
endif
if (mod(time,del_num)==0) then
  ! # of solution
  write(3,"(E12.5,1x,F12.2)") global_time, s_event_1
  ! # of crystals
  write(4,"(E12.5,1x,F12.2)") global_time, g_event
delta_mu [kcal/mol]
    endif
endif

! If nucleation mode is on then we want to reach a certain size then terminate as we are only interested
! up until the critical nuclei size.
if(n_mode==1) then
    if(g_event==crystalsize)then
        exit
    endif
enddo

close(2)
close(3)
close(4)
close(7)
close(8)
deallocate(grid,site)
end program

! This routine resizes one of the sub arrays as it fills up, at the moment only increases are supported
subroutine memory_reallocation(sitetype)
    use storage_structures
    implicit none
type(positions), allocatable, dimension(:):: temp
integer(kind=3), intent(in):: sitetype
integer(kind=3):: j, old_size

old_size = SIZE(site(sitetype)%position)
allocate(temp(1))
allocate(temp(1)%position(old_size))
do j = 1, old_size
    temp(1)%position(j) = site(sitetype)%position(j)
end do
deallocate(site(sitetype)%position)
allocate(site(sitetype)%position((old_size+10000)))
do j = 1, old_size
    site(sitetype)%position(j) = temp(1)%position(j)
end do
deallocate(temp)
end subroutine memory_reallocation

! Initialisation
subroutine setup(edgex,edgey,edgez,input_file,cagetypes,gstc,gstc_tot)
    use storage_structures
    implicit none
integer(kind=3):: status, i, j
integer(kind=3), intent(out):: cagetypes, gstc, gstc_tot
integer(kind=3), intent(in):: edgex,edgey,edgez
character(len=100), intent(in):: input_file

! Open the system information file and start reading in the information and!
! allocate data structures as information arrives.
open(1,file=input_file)
read(1,*) cagetypes
allocate(connections(cagetypes),stat=status)
if(status>0)  stop 'Fail to allocate memory for the connections array'
allocate(tsites(cagetypes),stat=status)
if(status>0)  stop 'Fail to allocate memory for the tsites array'
allocate(information(cagetypes,4),stat=status)
if(status>0)  stop 'Fail to allocate memory for the information array'
allocate(energy_scalars(cagetypes),stat=status)
if(status>0)  stop 'Fail to allocate memory for the information array'

! Here the arrays are populated with the connectivity information of the system
! information(:,1) contains number of TSITES per cage
! information(:,2) contains the offsets
! information(:,3) contains the number of neighbours per cage
! information(:,4) contains the fusion number (number of cages each Tsite is fused to)
gstc = 0
do  i = 1, cagetypes
    gstc = gstc + 2
    information(i,2) = gstc
    read(1,*), information(i,3), information(i,1), information(i,4)
allocate(connections(i)%net(information(i,3),4),stat=status)
if(status>0)  stop 'Fail to allocate memory for the connections array'
allocate(tsites(i)%net(information(i,1),information(i,4)),stat=status)
if(status>0)  stop 'Fail to allocate memory for the connections array'
do  j = 1, information(i,3)
    read(1,*), connections(i)%net(j,1), connections(i)%net(j,2),
    connections(i)%net(j,3), connections(i)%net(j,4)
enddo
do  j = 1, information(i,1)
    read(1,*), tsites(i)%net(j,1), tsites(i)%net(j,2), tsites(i)%net(j,3)
gstc = gstc + 1
enddo
gstc_tot = gstc * 2

! Here the rest of the data structures are initialised now all the information is there
allocate(int_energies(gstc_tot),stat=status)
if(status>0)  stop 'Fail to allocate memory for the internal energy array'
allocate(grid(edgex,edgey,edgez),stat=status)
if(status>0)  stop 'Fail to allocate memory for the Grid array,'
allocate(site(gstc_tot),stat=status)
if(status>0)  stop 'Fail to allocate memory for the Site array'
do  i = 1, gstc_tot
    allocate(site(i)%position(10000),stat=status)
    if(status>0)  stop 'Fail to allocate memory for the Site Array'
end do
allocate(neighbour_queue(100),stat=status)
if(status>0)  stop 'Fail to allocate memory for the Neighbours array'
allocate(c(gstc_tot),stat=status)
if(status>0)  stop 'Fail to allocate memory for the counters'
allocate(p(gstc_tot),stat=status)
if(status>0)  stop 'Fail to allocate memory for the Probability array'
close(1)
end subroutine setup
subroutine initialise(edgex, edgex, edgez, kT, cagetypes, gstc, gstc_tot, n_mode, starting_cage)
  use storage_structures
  implicit none
  type(coordinates):: gs
  integer(kind=3):: neighbour_queue_count, cage, energylevel
  integer(kind=3), intent(in):: edgex, edgex, edgez, cagetypes, gstc, gstc_tot, starting_cage,
  n_mode
  real(kind=2), intent(in):: kT
  gs%x = (edgex+1)/2; gs%y = (edgey+1)/2; gs%z = (edgez+1)/2

  ! Initial counters for growth and dissolution sites.
  c(1:gstc_tot) = 0

  ! If nucleation mode is off then initialise a starting cage
  if (n_mode==0) then
    grid(gs%x, gs%y, gs%z, 1) = 1
    neighbour_queue_count = 0
    call neighbour_check(gs, starting_cage, neighbour_queue_count, gstc, gstc_tot)
    neighbour_queue_count = 1
    call neighbours(gs, neighbour_queue_count, cagetypes, gstc, gstc_tot)
  endif

  ! This is the internal energy part of the energy calculation, this only needs to be performed
  once
  ! so it falls under initialisation.
  do cage = 1, cagetypes
    do energylevel = 1, information(cage,1)
      int_energies(energylevel+information(cage,2)) = exp(-0.5*energy_scalars(cage)*energylevel/kT)
    enddo
    do energylevel = 1, information(cage,1)
      int_energies(energylevel+information(cage,2)+gstc) = exp(0.5*energy_scalars(cage)*energylevel/kT)
    enddo
  enddo
end subroutine initialise

! Probability set
subroutine probabilities(delta_mu, cagetypes, gstc)
  use storage_structures
  implicit none
  integer(kind=3), intent(in):: cagetypes, gstc
  real(kind=2), intent(in):: delta_mu
  real(kind=2):: growthcasedmu, dissolutioncasedmu
  integer(kind=3):: cage, energylevel

  ! The only part of the energetics that changes each iteration is the dmus part so this is done
  here
  growthcasedmu = exp(+0.5*delta_mu)
  dissolutioncasedmu = exp(-0.5*delta_mu)

  ! The two parts are combined, this way caused an unprecedented speed up since hundreds of
  exp calculations
  ! are no longer being done per iteration. This simply uses a mathematical trick involving the
  properties of
  ! powers.
  do cage = 1, cagetypes

290
do energylevel = 1, information(cage,1) 
   p(energylevel+information(cage,2)) = int_energies(energylevel+information(cage,2)) * growthcasedmu 
   enddo 
   do energylevel = 1, information(cage,1) 
   p(energylevel+information(cage,2)+gstc) = int_energies(energylevel+information(cage,2)+gstc) * dissolutioncasedmu 
   enddo
end subroutine

subroutine neighbours(gs.neighbour_queue_count, cagetypes, gstc, gstc_tot)
  use storage_structures
  implicit none
  type(coordinates),intent(in):: gs
  type(coordinates):: ch
  integer(kind=3),intent(inout):: neighbour_queue_count
  integer(kind=3), intent(in):: cagetypes, gstc, gstc_tot
  integer(kind=3):: i, cagetype, intype, tmp, gst

  gst = grid(gs%x,gs%y,gs%z,2)

  ! Figure out which cage type we are dealing with
  if(gst>gstc) tmp = gst - gstc
  do i = 1, cagetypes
     if(tmp<=(information(i,1)+information(i,2))) then
        intype = i
        exit
     endif
  enddo

  ! Generate all of the neighbour coordinates
  do i = 1, information(intype,3)
     ch%x = gs%x + connections(intype)%net(i,1)
     ch%y = gs%y + connections(intype)%net(i,2)
     ch%z = gs%z + connections(intype)%net(i,3)
     cagetype = connections(intype)%net(i,4);
     call neighbour_check(ch, cagetype, neighbour_queue_count, gstc, gstc_tot)
  enddo
end subroutine

subroutine neighbour_check(ch, cagetype, neighbour_queue_count, gstc, gstc_tot)
  use storage_structures
  implicit none
  type(coordinates),intent(in):: ch
  integer(kind=3),intent(in):: cagetype, gstc, gstc_tot
  integer(kind=3),intent(inout):: neighbour_queue_count
  type(coordinates):: last
  integer(kind=3):: pre_st, pre_posi, st, new_st, qcount, i, j, neighbourcount, tmp, 
  tmpx, tmpy, tmpz

  !Look up the checked sites information then remove it from is current site type array position
  pre_st = grid(ch%x,ch%y,ch%z,2)
  if (pre_st>1 .and. pre_st<=gstc_tot) then
     pre_posi = grid(ch%x,ch%y,ch%z,3)
     site(pre_st)%position(pre_posi) = ch
     site(pre_st)%position(pre_posi) = site(pre_st)%position(c(pre_st))
     last = site(pre_st)%position(c(pre_st))
     grid(last%x,last%y,last%z,2) = pre_st
     grid(last%x,last%y,last%z,3) = pre_posi
  endif
end subroutine

291
Appendices – Appendix 2

\[
c(\text{pre}_\text{st}) = c(\text{pre}_\text{st})-1
\]

endif

! Workout how many neighbours are actually grown this will be used later to
! choose which bulk site we have (fully surrounded or not)
neighbourcount = 0
do \ i = 1, \ \text{information(cagetype,3)}
    tmpx = \text{connections(cagetype)}%\text{net}(i,1)
    tmpy = \text{connections(cagetype)}%\text{net}(i,2)
    tmpz = \text{connections(cagetype)}%\text{net}(i,3)
    neighbourcount = neighbourcount + 
    \text{grid(ch}x+\text{tmpx, ch}y+\text{tmpy, ch}z+\text{tmpz,1)}
endo

! Here we calculate how many Tsites we have (qcount)
qcount = 0
do \ i = 1, \ \text{information(cagetype,1)}
    tmp = 0
    do \ j = 1, \ \text{information(cagetype,4)}
        tmpx = \text{connections(cagetype)}%\text{net(tsites(cagetype)}%\text{net}(i,j,1)
        tmpy = \text{connections(cagetype)}%\text{net(tsites(cagetype)}%\text{net}(i,j,2)
        tmpz = \text{connections(cagetype)}%\text{net(tsites(cagetype)}%\text{net}(i,j,3)
        tmp = tmp + \text{grid(ch}x+\text{tmpx, ch}y+\text{tmpy, ch}z+\text{tmpz,1)}
endo
if(tmp==0) qcount = qcount + 1
endo

\[
\text{st} = \ \text{information(cagetype,2)} + \ \text{qcount}
\]
if(\text{information(cagetype,3)===neighbourcount}) \ \text{st} = \ \text{st} - 1

! If the first layer of the main array is 1 then that site is grown and thus probably
! just got its connectivity updated we add the number of site to the new site to
! get the dissolution site type. If it is zero and also qcount is zero then in a fused
! cage system all of the T-sites have been contributed by neighbouring cages,
! this is indirect growth and therefore the main arrays are updated with grown status.
! If we don't do this then they cause voids inside the crystal when rendering which
! has a good chance of screwing the colouring routines. Failing both of those
! scenario's then we arrive at an ordinary growth site
if (\text{grid(ch}x,ch}y,ch}z,1)==1) then
    new\_st = \text{st}+\text{gstc}
elseif(\text{grid(ch}x,ch}y,ch}z,1)==0.and qcount==0) then
    grid(ch}x,ch}y,ch}z,1) = 1
    new\_st = \text{st}+\text{gstc}
    \text{neighbour\_queue\_count} = \text{neighbour\_queue\_count} + 1
    \text{neighbour\_queue(\text{neighbour\_queue\_count})} = \text{ch}
else
    new\_st = \text{st}
endif

! Check if we need more space to start saving new sites, we always
! need a buffer zone of 10 just incase bad things happen.
if((\text{size(site(new\_st)\%position)})-c(new\_st)) <= 10) then
    \text{call memory\_reallocation(new\_st)}
endif

! Store new site type in grid.
\text{grid(ch}x,ch}y,ch}z,2) = \text{new\_st}
! If the new site type fails this test then it can’t be right so
! it is set to 0 so it can be like an initial element
if (new_st<1 .or. new_st>gstc_tot) then
  grid(ch%x,ch%y,ch%z,3) = 0
else
  ! Otherwise if it passes then things are all good and the normal
  ! addition to the Counters and arrays ensues here.
  c(new_st) = c(new_st)+1
  grid(ch%x,ch%y,ch%z,3) = c(new_st)
  site(new_st)%position(c(new_st)) = ch
endif

! This if statement is only intended for advanced debugging, if it starts outputting then there is
! something wrong.
if(new_st<1 .or. new_st>gstc_tot)then
  print*, "Site Type = ", new_st, "st = ", st, "qcount = ", qcount, "neighbourcount = ", neighbourcount, "cagetype = ", cagetype
  print*, "x = ",ch%x, "y = ", ch%y, "z = ",ch%z
endif
end subroutine neighbour_check

subroutine nucleation(edgex, edgey, edgez, g_event, cagetypes, gstc, gstc_tot)
  use storage_structures
  implicit none
  integer(kind=3): i, neighbour_queue_count, gst, total_prob, sum_prob, cage, tmp
  integer(kind=3), intent(in): edgex, edgey, edgez, cagetypes, gstc, gstc_tot
  real(kind=2), intent(out): g_event
  type(coordinates): :: gs
  real(kind=2): :: r
!
! some times due to dissolution on small chains can create multiple grown cages
! these are eliminated here.
do cage = 1, cagetypes
tmp = information(cage,1) + information(cage,2) + gstc
if(c(tmp)>0) then
  if(c(tmp)==1 .and. (SUM(c(tmp:gstc_tot))==1)) then
    gs = site(tmp)%position(i)
    grid(gs%x,gs%y,gs%z,1) = 0
    grid(gs%x,gs%y,gs%z,2) = 0
    grid(gs%x,gs%y,gs%z,3) = 0
    g_event = g_event - 1
    neighbour_queue_count = 1
    call neighbours(gs,neighbour_queue_count,cagetypes,gstc,gstc_tot)
  endif
  else
    gs%x = (edgex+1)/2; gs%y = (edgey+1)/2; gs%z = (edgez+1)/2
    grid(gs%x,gs%y,gs%z,1) = 1
    grid(gs%x,gs%y,gs%z,2) = tmp
    grid(gs%x,gs%y,gs%z,3) = 1
    site(tmp)%position(1) = gs
    g_event = 1
    neighbour_queue_count = 1
    call neighbours(gs,neighbour_queue_count,cagetypes,gstc,gstc_tot)
  endif
enddo
Appendices – Appendix 2

\[
\begin{align*}
\text{do } i &= 1, c(tmp) \\
\text{gs} &= \text{site(tmp)}\%\text{position}(i) \\
\text{grid(gs\%x,gs\%y,gs\%z,1)} &= 0 \\
\text{grid(gs\%x,gs\%y,gs\%z,2)} &= 0 \\
\text{grid(gs\%x,gs\%y,gs\%z,3)} &= 0 \\
\text{g_event} &= \text{g_event} - 1 \\
\text{neighbour_queue_count} &= 1 \\
\text{call neighbours(gs,neighbour_queue_count,cagetypes,gstc,gstc_tot)} \\
\text{enddo} \\
c(tmp) &= 0 \\
\text{endif} \\
\text{endif} \\
\text{enddo}
\end{align*}
\]

![neighbours(gs,neighbour_queue_count,cagetypes,gstc,gstc_tot)

\begin{align*}
\text{neighbours}(gs,\text{neighbour_queue_count},cagetypes,\text{gstc},\text{gstc_tot}) \\
\text{endif} \\
c(tmp) &= 0 \\
\text{enddo} \\
\text{endif}
\end{align*}

! Get rid of all ungrown nuclei for all cage types

do cage = 1, cagetypes
\[
\begin{align*}
tmp &= \text{information(cage,1)} + \text{information(cage,2)} \\
\text{if}(c(tmp)>0) &\text{ then} \\
\text{do } i &= 1, c(tmp) \\
\text{gs} &= \text{site(tmp)}\%\text{position}(i) \\
\text{grid(gs\%x,gs\%y,gs\%z,1)} &= 0 \\
\text{grid(gs\%x,gs\%y,gs\%z,2)} &= 0 \\
\text{grid(gs\%x,gs\%y,gs\%z,3)} &= 0 \\
\text{enddo} \\
c(tmp) &= 0 \\
\text{endif}
\end{align*}
\]

! Here the nuclei will be chosen is there isn’t one already
\[
\text{if}(\text{SUM(c)==0}) &\text{ then} \\
\text{total_prob} &= 0 \\
\text{do } cage = 1, cagetypes \\
\text{total_prob} &= \text{total_prob} + p(\text{information(cage,1)} + \text{information(cage,2)}) \\
\text{enddo} \\
\text{call random_number(r)} \\
\text{sum_prob} &= 0 \\
\text{do } cage = 1, cagetypes \\
\text{sum_prob} &= \text{sum_prob} + (p(tmp)/\text{total_prob}) \\
\text{if } (\text{sum_prob}>=r) &\text{ then} \\
\text{gst} &= \text{tmp} \\
\text{gs\%x} &= ((\text{edgez}+1)/2); \text{gs\%y} = (\text{edgey}+1)/2; \text{gs\%z} = \\
(\text{edgez}+1)/2 \\
\text{exit} \\
\text{endif} \\
\text{enddo} \\
\text{grid(gs\%x,gs\%y,gs\%z,1)} &= 1 \\
\text{grid(gs\%x,gs\%y,gs\%z,2)} &= \text{gst} \\
\text{grid(gs\%x,gs\%y,gs\%z,3)} &= 1 \\
\text{site(gst)}\%\text{position}(1) &= \text{gs} \\
\text{c(gst)} &= 1 \\
\text{g_event} &= 1 \\
\text{neighbour_queue_count} &= 1 \\
\text{call neighbours(gs,neighbour_queue_count,cagetypes,gstc,gstc_tot)} \\
\text{endif} \\
\text{end subroutine nucleation}
\]
Appendix 3

This is the contents of the file that defines the connectivity of the LTA system.

```
5
6 8 3
+1 0 0 4
-1 0 0 4
0 +1 +1 5
0 +1 -1 5
0 -1 -1 5
0 -1 +1 5
1 3 4
1 3 6
1 6 5
1 4 5
2 3 4
2 3 6
2 6 5
2 4 5
6 -8 3
0 +1 0 4
0 -1 0 4
+1 0 +1 5
+1 0 -1 5
-1 0 -1 5
-1 0 +1 5
1 3 6
1 3 4
1 4 5
1 6 5
2 3 4
2 4 5
2 6 5
6 -8 3
0 0 +1 4
0 0 -1 4
+1 +1 0 5
+1 -1 0 5
-1 +1 0 5
-1 -1 0 5
1 3 5
1 3 4
1 4 6
1 5 6
2 3 5
2 3 4
2 4 6
2 5 6
14 24 3
+1 0 0 1
-1 0 0 1
0 +1 0 2
0 -1 0 2
0 0 +1 3
0 0 -1 3
+1 +1 +1 5
+1 +1 -1 5
-1 +1 -1 5
```
Appendices – Appendix 4

-1  +1  +1  5
+1  -1  +1  5
+1  -1  -1  5
-1  -1  -1  5
-1  -1  +1  5
3   7  10
3   7  8
3   8  9
3  10  9
5   7  10
1   7  8
6   8  9
2  10  9
5   7  11
1  8  12
6   8  12
6   9  13
2   9  13
2  10  14
5  10  14
5  11  14
1  11  12
6   12  13
2   14  13
4   11  14
4   11  12
4   12  13
4   14  13
26  48  3
+2  0  0  5
-2  0  0  5
 0  +2  0  5
0   -2  0  5
0   0   +2  5
0   0   -2  5
+1  +1  +1  4
+1  +1  -1  4
-1  +1  -1  4
-1  +1  +1  4
+1  -1  +1  4
+1  -1  -1  4
-1  -1  -1  4
-1  -1  +1  4
 0   +1  +1  1
+1  +1  0  3
 0   +1  -1  1
 0   +1  0  3
+1  0   +1  2
+1  0   -1  2
-1  0   -1  2
-1  0   +1  2
 0   -1  +1  1
+1  -1  0  3
 0   -1  -1  1
-1  -1  0  3
3   15  7
3   16  7
3   16  8
3   17  8
3   17  9
3  18  9  
3  18  10 
3  15  10 
5  15  7  
1  16  7 
1  16  8 
6  17  8 
6  17  9 
2  18  9 
2  18  10 
5  15  10 
5  19  7  
1  19  7 
1  20  8 
6  20  8 
6  21  9 
2  21  9 
2  22  10 
5  22  10 
5  19  11 
1  19  11 
1  20  12 
6  20  12 
6  21  13 
2  21  13 
2  22  14 
5  22  14 
5  23  11 
1  24  11 
1  24  12 
6  25  12 
6  25  13 
2  26  13 
2  26  14 
5  23  14 
4  23  11 
4  24  11 
4  24  12 
4  25  12 
4  25  13 
4  26  13 
4  26  14 
4  23  14
Appendices – Appendix 4

Appendix 4

This is the contents of the input file that defines the connectivity for the sodalite system.

```
1
14 24 3
+2 0 0 1
-2 0 0 1
0 +2 0 1
0 -2 0 1
0 0 +2 1
0 0 -2 1
+1 +1 +1 1
+1 +1 -1 1
-1 +1 -1 1
-1 +1 +1 1
+1 -1 +1 1
+1 -1 -1 1
-1 -1 -1 1
-1 -1 +1 1
3  7  10
3  7  8
3  8  9
3 10  9
5  7  10
1  7  8
6  8  9
2 10  9
5  7  11
1  7  11
1  8  12
6  8  12
6  9  13
2  9  13
2 10  14
5 10  14
5 11  14
1 11  14
6 12  13
2 14  13
4 11  14
4 11  12
4 12  13
4 14  13
```
Appendix 5

Here are all of the source files for the visualiser program. This program’s source has been split into multiple files to enable easy editing of the source code without having to resort to using the grep command for navigation. These files should be placed into a suitably configured visual C++ Express project linking the win32 and opengl dll’s to compile.

Colouring.h

```c
#pragma once
#include "FileOps.h"

#ifndef COLOURING_H
    // Define the function prototypes
#define COLOURING_H
void colouring(int mode, int startframe, int stopframe);
#endif
```

Colouring.cpp

```c
//------------------------------------------------------------------------------
//------------------------------------------------------------------------------

#include "Colouring.h"

extern OBJECT vertices;
extern FRAMEOBJECT frames;

//------------------------------------------------------------------------------
//------------------------------------------------------------------------------

int layercolour(int mode, double colour)
{
    int cv;
    double colour4;
    int icolour4, realcolour;

    colour4 = colour/8;
    icolour4 = int(colour4);
    realcolour = int((8*(colour4-icolour4));

    if (mode == 1){
        if (realcolour==0){ cv = 4; }
        else if (realcolour==1){ cv = 4; }
        else if (realcolour==2){ cv = 3; }
        else if (realcolour==3){ cv = 3; }
        else if (realcolour==4){ cv = 2; }
        else if (realcolour==5){ cv = 2; }
        else if (realcolour==6){ cv = 1; }
        else if (realcolour==7){ cv = 1; }
    } else {
```

299
if (realcolour == 0) { cv = 4; }
else if (realcolour == 1) { cv = 3; }
else if (realcolour == 2) { cv = 3; }
else if (realcolour == 3) { cv = 2; }
else if (realcolour == 4) { cv = 2; }
else if (realcolour == 5) { cv = 1; }
else if (realcolour == 6) { cv = 1; }
else if (realcolour == 7) { cv = 4; }
}
return (cv);

void colouring(int mode, int startframe, int stopframe) // for cubic systems
{
    int tx, ty, tz, cx, cy, cz;
    int cv, firstframe, lastframe, num_layers_colour;
    double colour, colour2;

    num_layers_colour = 10;

    for (int i = startframe; i < stopframe; i++)
    {
        firstframe = frames.frame[i].StartFrame;
        lastframe = frames.frame[i].EndFrame;

        for (int j = firstframe; j < lastframe; j++)
        {
            tx = vertices.points[j].x;
            ty = vertices.points[j].y;
            tz = vertices.points[j].z;

            cx = abs(tx - frames.frame[i].OffsetX);
            cy = abs(ty - frames.frame[i].OffsetY);
            cz = abs(tz - frames.frame[i].OffsetZ);

            if ((cx + cy + cz) > (frames.frame[i].face111-
                 frames.frame[i].adjustment111-num_layers_colour))
            {
                colour = cx + cy + cz;
                cv = layercolour(mode, colour);
            } else if (((cx + cy)>(frames.frame[i].face110-
                           frames.frame[i].adjustment110-num_layers_colour))
                        || ((cy + cz)>(frames.frame[i].face110-
                           frames.frame[i].adjustment110-num_layers_colour))
                        || ((cx + cz)>(frames.frame[i].face110-
                           frames.frame[i].adjustment110-num_layers_colour))
                        || ((cx + cy + cz) > (frames.frame[i].face111-
                                             frames.frame[i].adjustment111-num_layers_colour)))
            {
                colour2 = max(cx + cy + cz);
                colour = max(colour2, cx + cz);
                colour = cx + cz;
            } else
            {
                colour = cx + cy + cz;
            }
cv = layercolour(mode,colour);

} else if (cx > (frames.frame[i].face100-
frames.frame[i].adjustment100-num_layersColour) ||
  cy > (frames.frame[i].face100-
frames.frame[i].adjustment100-num_layersColour) ||
  cz > (frames.frame[i].face100-
frames.frame[i].adjustment100-num_layersColour)) {
  colour2 = max(cx,cy);
  colour = max(colour2,cz);
  cv = layercolour(mode,colour);
} else {
  cv = 5;
}
vertices.points[j].c = cv;
}
}*/

//------------------------------------------------------------------
//------------------------------------------------------------------
----
/*
void colouring(int mode, int startframe, int stopframe) // adapted
  for LTL
{
  int tx,ty,tz,cx,cy,cz;
  int cv,firstframe,lastframe,num_layersColour;
  double colour,colour2,a_r1,a_r2;
  num_layersColour = 10;
  for(int i=startframe;i<stopframe;i++)
  {
    firstframe = frames.frame[i].StartFrame;
    lastframe = frames.frame[i].EndFrame;
    for(int j=firstframe;j<lastframe;j++)
    {
      tx=vertices.points[j].x;
      ty=vertices.points[j].y;
      tz=vertices.points[j].z;
      cx = abs(tx - frames.frame[i].OffsetX);
      cy = abs(ty - frames.frame[i].OffsetY);
      cz = abs(tz - frames.frame[i].OffsetZ);
      if (cy > (frames.frame[i].GreatestY-
        frames.frame[i].OffsetY-15))
      {
        colour = cy;
        cv = layercolour(mode,colour);
      } else if (cz > (frames.frame[i].GreatestZ-
        frames.frame[i].OffsetZ-15))
      {
void colouring(int mode, int startframe, int stopframe) // adapted
for ERI and OFF due to the switch in y and z coords in the
simulation source
{
    int tx,ty,tz,cx, cy, cz;
    int cv, firstframe, lastframe, num_layers_colour;
    double colour;
    num_layers_colour = 10;

    for (int i=startframe; i<stopframe; i++)
    {
        firstframe = frames.frame[i].StartFrame;
        lastframe = frames.frame[i].EndFrame;

        for (int j=firstframe; j<lastframe; j++)
        {
            tx = vertices.points[j].x;
            ty = vertices.points[j].y;
            tz = vertices.points[j].z;

            cx = abs(tx - frames.frame[i].OffsetX);
            cy = abs(ty - frames.frame[i].OffsetY);
            cz = abs(tz - frames.frame[i].OffsetZ);

            if (cy > (frames.frame[i].GreatestY -
                frames.frame[i].OffsetY-15))
            {
                colour = cy;
                cv = layercolour(mode, colour);
            }
            else if (cz > (frames.frame[i].GreatestZ-
                frames.frame[i].OffsetZ-15))
            {
                colour = cz;
                cv = layercolour(mode, colour);
            }
} else if({(cx + cy)>(frames.frame[1].face110-40-frames.frame[i].adjustment110-num_layers_colour))
{
    colour = cx+cy;
    cv = layercolour(mode,colour);
}
else {
    cv = 4;
}
vertices.points[j].c = cv;
}
}

CreateWindow.h

#include "Resource.h"
#include "Messages.h"

#ifndef CreateWindow_H
    // Define the function prototypes
#define CreateWindow_H

BOOL CreateGLWindow(char* title, int width, int height, int bits);
#endif

CreateWindow.cpp

#include "CreateWindow.h"

extern HDC hDC;
extern HGLRC hRC;
extern HWND hWnd;
extern HINSTANCE hInstance;

BOOL CreateGLWindow(char* title, int width, int height, int bits)
{
    GLuint PixelFormat;
    WNDCLASS wc;
    DWORD dwExStyle;
    DWORD dwStyle;
    RECT WindowRect;
    WindowRect.left=(long)0;
    WindowRect.right=(long)width;
    WindowRect.top=(long)0;
    WindowRect.bottom=(long)height;

    hInstance = GetModuleHandle(NULL);
    wc.style = CS_HREDRAW | CS_VREDRAW | CS_OWNDC;
    wc.lpfWndProc = (WNDPROC)WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_LARGE));
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);


Appendices – Appendix 5

wc.hbrBackground = NULL;
cwc.lpszMenuName = MAKEINTRESOURCE(IDC_TEST2);
cwc.lpszClassName = "OpenGL";

if (!RegisterClass(&wc))
{
    MessageBox(NULL, "Failed To Register The Window Class.", "ERROR", MB_OK | MB_ICONEXCLAMATION);
    return FALSE;
}

dwExStyle = WS_EX_APPWINDOW | WS_EX_WINDOWEDGE;
dwStyle = WS_OVERLAPPEDWINDOW;
AdjustWindowRectEx(&WindowRect, dwStyle, FALSE, dwExStyle);

if (!((hWnd = CreateWindowEx(
    dwExStyle, "OpenGL", title, dwStyle |
    WS_CLIPSIBLINGS | WS_CLIPCHILDREN, 
    0, 0, WindowRect.right-
    WindowRect.left, 
    WindowRect.top, 
    NULL, NULL, 
    hInstance, 
    NULL))

    { 
        KillGLWindow();
        MessageBox(NULL, "Window Creation Error.", "ERROR", MB_OK | MB_ICONEXCLAMATION);
        return FALSE;
    }
}

static PIXELFORMATDESCRIPTOR pfд =
{
    sizeof(PIXELFORMATDESCRIPTOR),
    1,
    PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER,
    PFD_TYPE_RGBA,
    bits,
    0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 
    PFD_MAIN_PLANE,
    0,
    0, 0, 0
};
if (!hDC=GetDC(hWnd))
{
    KillGLWindow();
    MessageBox(NULL,"Can't Create A GL Device
Context.","ERROR",MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

if (!PixelFormat=ChoosePixelFormat(hDC,&pfd))
{
    KillGLWindow();
    MessageBox(NULL,"Can't Find A Suitable
PixelFormat.","ERROR",MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

if (!SetPixelFormat(hDC,PixelFormat,&pfd))
{
    KillGLWindow();
    MessageBox(NULL,"Can't Set The
PixelFormat.","ERROR",MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

if (!hRC=wglCreateContext(hDC))
{
    KillGLWindow();
    MessageBox(NULL,"Can't Create A GL Rendering
Context.","ERROR",MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

if (!wglMakeCurrent(hDC,hRC))
{
    KillGLWindow();
    MessageBox(NULL,"Can't Activate The GL Rendering
Context.","ERROR",MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

ShowWindow(hWnd,SW_SHOW);
SetForegroundWindow(hWnd);
SetFocus(hWnd);
ReSizeGLScene(width, height);
InitGL();

return TRUE;

DisplayLists.h

#include "Opengl.h"
#ifndef DisplayLists_H
    // Define the function prototypes
#define DisplayLists_H

GLvoid BuildListssod();
GLvoid BuildLists2DSod();
GLvoid BuildListsltl();
GLvoid BuildListsoff();
GLvoid BuildListseri();
Appendices – Appendix 5

GLvoid BuildListslta();
#endif

DisplayLists.cpp

//---------------------------------------------------------------
//  Appendix 5
//---------------------------------------------------------------

#include "DisplayLists.h"

GLuint list;  // Storage For The Display List
GLsizei range=0;

//---------------------------------------------------------------

GLvoid BuildListssod()
{
    range = 1;
    list=glGenLists(1);  // Generate 5 Lists
    glNewList(1,GL_COMPILE);
    // Start With The first List

    glBegin(GL_QUADS);
    glNormal3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
    //V1
    glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
    //V2
    glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
    //V3
    glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
    //V4
    glEnd();

    glBegin(GL_QUADS);
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
    //V5
    glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
    //V6
    glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
    //V7
    glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
    //V8
    glEnd();

    glBegin(GL_QUADS);
    glNormal3f(0.0f, -1.0f, 0.0f);
    glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
    //V9
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
    //V10
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V11
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
    //V12
glEnd();

eglBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
    //V13
eglVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
    //V14
eglVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
eglVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
    //V16
eglEnd();

eglBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
    //V17
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
    //V18
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
    //V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
    //V20
glEnd();

eglBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, 1.0f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
    //V21
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
    //V22
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
    //V23
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
    //V24
glEnd();

eglBegin(GL_POLYGON);
glNormal3f(-0.58f, 0.58f, 0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
glVertex3f(-0.0042475f, +0.0000000f, +0.0084955f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
glEnd();
eglBegin(GL_POLYGON);
Appendices – Appendix 5

```c
glNormal3f(0.58f, 0.58f, 0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
glEnd();

glBegin(GL_POLYGON);
glNormal3f(0.58f, 0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0000000f);
glVertex3f(-0.0042475f, +0.0084955f, -0.0000000f);
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
glEnd();

glBegin(GL_POLYGON);
glNormal3f(-0.58f, 0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0000000f);
glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
glVertex3f(-0.0084955f, +0.0042475f, +0.0000000f);
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glEnd();

glBegin(GL_POLYGON);
glNormal3f(-0.58f, -0.58f, 0.58f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
glVertex3f(-0.0084955f, -0.0000000f, -0.0084955f);
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glEnd();
```
glBegin(GL_POLYGON);
  glNormal3f(0.58f, -0.58f, 0.58f);
  glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  glEnd();

glBegin(GL_POLYGON);
  glNormal3f(0.58f, -0.58f, -0.58f);
  glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
  glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
  glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
  glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  glEnd();

glBegin(GL_POLYGON);
  glNormal3f(-0.58f, -0.58f, -0.58f);
  glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
  glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
  glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
  glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
  glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  glEnd();

} // BuildListsltl()
glNewList(1, GL_COMPILE);
  // Start With The first List
  glRotatef(30.0f, 0.0f, 0.0f, 1.0f);

  glBegin(GL_TRIANGLE_FAN);
  glVertex3f(+0.00245f, +0.00000f, +0.00000f);
  //Va
  glNormal3f(-0.92f, 0.35f, 0.17f);
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);
  //V13
  glVertex3f(+0.00265f, +0.00155f, -0.00215f);
  //V1
  glNormal3f(-1.00f, 0.00f, -0.09f);
  glVertex3f(+0.00265f, -0.00155f, -0.00215f);
  //V6
  glNormal3f(-0.92f, -0.35f, 0.17f);
  glVertex3f(+0.00380f, -0.00350f, +0.00000f);
  //V18
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
  //V14
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);
  //V13
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(-0.35f, -0.60f, 0.73f);
  glVertex3f(+0.00265f, -0.00155f, +0.00215f);
  //V1
  glVertex3f(-0.00000f, +0.00310f, -0.00215f);
  //V8
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
  //V14
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);
  //V13
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(-0.35f, -0.59f, -0.73f);
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
  //V14
  glVertex3f(-0.00000f, +0.00310f, +0.00215f);
  //V8
  glVertex3f(+0.00265f, +0.00155f, +0.00215f);
  //V7
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);
  //V13
  glEnd();

  glBegin(GL_TRIANGLE_FAN);
  glVertex3f(-0.001325f, +0.002325f, +0.00000f);
  //Vb
  glNormal3f(0.20f, -0.95f, 0.22f);
  glVertex3f(-0.00495f, +0.00155f, +0.00000f);
  //V15
  glVertex3f(-0.00265f, +0.00155f, -0.00215f);
  //V3
glNormal3f(0.5f, -0.86f, 0.0f);
glVertex3f(-0.00000f, +0.00310f, -0.00215f);
    //V2
glNormal3f(0.73f, -0.65f, 0.22f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
    //V4
glNormal3f(0.73f, -0.65f, -0.22f);
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    //V8
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
    //V9
glNormal3f(0.20f, -0.95f, -0.22f);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
    //V15

glEnd();

glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, 0.73f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V16
glVertex3f(-0.00495f, +0.00155f, +0.00215f);
    //V10

glEnd();


glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, -0.73f);
glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    //V4

glVertex3f(-0.00265f, -0.00155f, +0.00215f);
    //V9

glVertex3f(-0.00265f, -0.00155f, +0.00000f);
    //V16

glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V15

glEnd();


glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, -0.73f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    //V3

glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    //V4

glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V16

glVertex3f(-0.00495f, +0.00155f, +0.00215f);
    //V15

glEnd();


glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, 0.73f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    //V3

glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    //V4

glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V16

glVertex3f(-0.00495f, +0.00155f, +0.00215f);
    //V15

glEnd();


glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.001325f, -0.002325f, +0.00000f);
    //Vc

glNormal3f(0.73f, 0.65f, 0.22f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    //V17

glVertex3f(-0.00000f, -0.00310f, -0.00215f);
    //V5

glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    //V4

glNormal3f(0.20f, 0.95f, 0.22f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V16

glNormal3f(0.20f, 0.95f, -0.22f);
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
    //V10

glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
    //V11
glNormal3f(0.82f, 0.46f, -0.34f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
//V17

glEnd();

glBegin(GL_QUADS);
glNormal3f(-0.34f, 0.60f, 0.72f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
//V5
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
//V6
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
//V18
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
//V17

glEnd();

// 6 bond lines representing 2 coordinate side
growth

glBegin(GL_LINES);    // corrected
    glVertex3f(+0.00110f, +0.00505f, +0.00000f);
    glVertex3f(+0.00025f, +0.00655f, +0.00000f);
    glVertex3f(+0.00025f, +0.00655f, +0.00000f);
    glVertex3f(+0.00535f, +0.00350f, +0.00000f);
    glVertex3f(+0.00535f, +0.00350f, +0.00000f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    glVertex3f(-0.00580f, -0.00305f, +0.00000f);
    glVertex3f(-0.00580f, -0.00305f, +0.00000f);
    glVertex3f(-0.00495f, +0.00155f, +0.00000f);
    glVertex3f(-0.00580f, +0.00305f, +0.00000f);
    glVertex3f(-0.00580f, +0.00305f, +0.00000f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    glVertex3f(+0.00535f, -0.00350f, +0.00000f);
    glVertex3f(+0.00535f, -0.00350f, +0.00000f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);

    glEnd();
glBegin(GL_LINES);  //corrected
    glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    //V17
    glVertex3f(+0.00025f, -0.00655f, +0.00000f);
    //V24
    glEnd();

    //six ring connector
    glBegin(GL_QUAD_STRIP);
    glNormal3f(-0.50f, -0.86f, 0.0f);
    glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V1
    glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V25
    glVertex3f(-0.00000f, +0.00310f, -0.00215f);
    //V2
    glVertex3f(-0.00000f, +0.00310f, -0.00380f);
    //V27
    glNormal3f(0.50f, -0.86f, 0.0f);
    glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    //V3
    glVertex3f(-0.00265f, +0.00155f, -0.00380f);
    //V29
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    //V4
    glVertex3f(-0.00265f, -0.00155f, -0.00380f);
    //V31
    glNormal3f(0.50f, 0.86f, 0.0f);
    glVertex3f(0.00000f, -0.00310f, -0.00215f);
    //V5
    glVertex3f(0.00000f, -0.00310f, -0.00380f);
    //V33
    glNormal3f(-0.50f, 0.86f, 0.0f);
    glVertex3f(+0.00265f, -0.00155f, -0.00215f);
    //V6
    glVertex3f(+0.00265f, -0.00155f, -0.00380f);
    //V35
    glNormal3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V1
    glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V25
    glEnd();

    glBegin(GL_QUAD_STRIP);
    glNormal3f(-0.5f, -0.86f, 0.0f);
    glVertex3f(+0.00265f, +0.00155f, +0.00215f);
    //V7
    glVertex3f(+0.00265f, +0.00155f, +0.00380f);
    //V26
    glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    //V8
    glVertex3f(-0.00000f, +0.00310f, +0.00380f);
    //V28
    glNormal3f(0.5f, -0.86f, 0.0f);
    glVertex3f(-0.00265f, +0.00155f, +0.00215f);
    //V9
    glVertex3f(-0.00265f, +0.00155f, +0.00380f);
    //V30
    glNormal3f(1.0f, 0.0f, 0.0f);

313
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
    //V10
glVertex3f(-0.00265f, -0.00155f, +0.00380f);
    //V32
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
    //V11
glVertex3f(-0.00000f, -0.00310f, +0.00380f);
    //V34
glNormal3f(-0.50f, 0.86f, 0.0f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    //V12
glVertex3f(+0.00265f, -0.00155f, +0.00380f);
    //V36
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
    //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
    //V26

glEnd();
glRotatef(-30.0f, 0.0f, 0.0f, 1.0f);
glEndList();

glNewList(2, GL_COMPILE);
    // Start With The first List
    glRotatef(-30.0f, 0.0f, 0.0f, 1.0f);

    glBegin(GL_TRIANGLE_FAN);
    glVertex3f(+0.00245f, +0.00000f, +0.00000f);
        //Va
    glNormal3f(-0.92f, 0.35f, 0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
        //V13
    glVertex3f(+0.00265f, +0.00155f, -0.00215f);
        //V1
    glNormal3f(-1.00f, 0.00f, -0.09f);
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
        //V6
    glNormal3f(-0.92f, -0.35f, 0.17f);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
        //V18
    glNormal3f(-0.92f, -0.35f, -0.17f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
        //V12
    glNormal3f(-1.00f, 0.00f, 0.09f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
        //V7
    glNormal3f(-0.92f, 0.35f, -0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
        //V13
    glEnd();

glBegin(GL_TRIANGLES);
    glNormal3f(-0.35f, -0.60f, 0.73f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
        //V1
    glVertex3f(-0.00000f, +0.00310f, -0.00215f);
        //V2
    glVertex3f(+0.00110f, +0.00505f, +0.00000f);
        //V14

314
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
//V13

glEnd();

glBegin(GL_QUADS);
  glNormal3f(-0.35f, -0.59f, -0.73f);
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
  //V14
  glVertex3f(-0.00000f, +0.00310f, +0.00215f);
  //V8
  glVertex3f(+0.00265f, +0.00155f, +0.00000f);
  //V7
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);
  //V13

glEnd();

glBegin(GL_TRIANGLES);  
  glVertex3f(-0.001325f, +0.002325f, +0.00000f);
  glNormal3f(0.20f, -0.95f, 0.22f);
  glVertex3f(-0.00495f, +0.00155f, +0.00000f);
  //Vb
  glVertex3f(-0.00265f, -0.00155f, -0.00215f);
  //V3
  glVertex3f(0.5f, -0.86f, 0.0f);
  glVertex3f(-0.00000f, +0.00310f, -0.00215f);
  //V2
  glVertex3f(0.73f, -0.65f, 0.22f);
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
  //V14
  glVertex3f(0.73f, -0.65f, -0.22f);
  glVertex3f(-0.00000f, +0.00310f, +0.00215f);
  //V8
  glVertex3f(0.50f, -0.86f, 0.0f);
  glVertex3f(-0.00265f, +0.00155f, +0.00000f);
  //V9
  glVertex3f(0.20f, -0.95f, -0.22f);
  glVertex3f(-0.00495f, +0.00155f, +0.00000f);
  //V15

glEnd();

glBegin(GL_QUADS);
  glVertex3f(-0.0015325f, +0.002325f, +0.00000f);
  glVertex3f(-0.00265f, +0.00155f, -0.00215f);
  glVertex3f(-0.00265f, -0.00155f, -0.00215f);
  glVertex3f(-0.001325f, +0.002325f, +0.00000f);

glEnd();

glBegin(GL_TRIANGLES);
  glVertex3f(0.68f, 0.0f, 0.73f);
  glVertex3f(-0.00265f, -0.00155f, -0.00215f);
  glVertex3f(-0.00265f, -0.00155f, -0.00215f);
  glVertex3f(-0.001325f, +0.002325f, +0.00000f);

glEnd();

glBegin(GL_TRIANGLES);
  glVertex3f(0.68f, 0.0f, -0.73f);
  glVertex3f(-0.00495f, -0.00155f, +0.00000f);
  glVertex3f(-0.00495f, -0.00155f, +0.00000f);
  glVertex3f(-0.001325f, +0.002325f, +0.00000f);

glEnd();
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
    // V10

glEnd();

glBegin(GL_TRIANGLES_FAN);
    glVertex3f(-0.001325f, -0.002325f, +0.00000f);
    // Vc
    glNormal3f(0.73f, 0.65f, 0.22f);
    glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    // V17
    glVertex3f(-0.00000f, -0.00310f, -0.00215f);
    // V5
    glNormal3f(0.50f, 0.86f, 0.0f);
    glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    // V10
    glNormal3f(0.20f, 0.95f, 0.22f);
    glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    // V16
    glNormal3f(0.20f, 0.95f, -0.22f);
    glVertex3f(-0.00265f, -0.00155f, -0.00215f);
    // V4
    glNormal3f(0.50f, 0.86f, 0.0f);
    glVertex3f(-0.00000f, -0.00310f, +0.00215f);
    // V11
    glNormal3f(0.82f, 0.46f, -0.34f);
    glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    // V17
    glNormal3f(-0.34f, 0.60f, 0.72f);
    glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    // V6
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    // V18
    glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    // V17
    glEnd();

glBegin(GL_QUADS);
    glNormal3f(-0.35f, 0.60f, -0.72f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    // V18
    glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    // V17
    glVertex3f(+0.00000f, -0.00310f, +0.00215f);
    // V11
    glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    // V12
    glEnd();

    glBegin(GL_LINES);
      // corrected
      glVertex3f(+0.00110f, +0.00505f, +0.00000f);
      // V14
      glVertex3f(+0.00025f, +0.00655f, +0.00000f);
      // V19
      glEnd();
glBegin(GL_LINES);  // corrected
  glVertex3f(+0.00380f, +0.00350f, +0.00000f);  //V13
  glVertex3f(+0.00535f, +0.00350f, +0.00000f);  //V20
glEnd();

// six ring connector

glBegin(GL_LINES);  // corrected
  glVertex3f(-0.00495f, -0.00155f, +0.00000f);  //V16
  glVertex3f(-0.00580f, -0.00305f, +0.00000f);  //V21
glEnd();

glBegin(GL_LINES);  // corrected
  glVertex3f(-0.00495f, +0.00155f, +0.00000f);  //V15
  glVertex3f(-0.00580f, +0.00305f, +0.00000f);  //V22
glEnd();

glBegin(GL_LINES);  // corrected
  glVertex3f(+0.00380f, -0.00350f, +0.00000f);  //V18
  glVertex3f(+0.00535f, -0.00350f, +0.00000f);  //V23
glEnd();

glBegin(GL_LINES);  // corrected
  glVertex3f( 0.00265f,  0.00155f, -0.00215f);  //V1
  glVertex3f( 0.00265f,  0.00155f, -0.00380f);  //V25
  glVertex3f( 0.00000f,  0.00310f, -0.00215f);  //V2
  glVertex3f( 0.00000f,  0.00310f, -0.00380f);  //V27
  glVertex3f( 0.50f, -0.86f,  0.0f);  //V3
  glVertex3f(-0.00265f, -0.00155f, -0.00215f);  //V29
  glVertex3f( 1.0f,  0.0f,  0.0f);  //V4
  glVertex3f(-0.00265f, -0.00155f, -0.00380f);  //V31
  glVertex3f( 0.50f,  0.86f,  0.0f);  //V5
  glVertex3f(-0.00000f, -0.00310f, -0.00215f);  //V33
  glVertex3f(-0.50f,  0.86f,  0.0f);  //V6
Appendices – Appendix 5

```c
glVertex3f(+0.00265f, -0.00155f, -0.00380f);
   //V35
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
   //V1
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
   //V25
glEnd();

glBegin(GL_QUAD_STRIP);
glNormal3f(-0.5f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
   //V26
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8
glVertex3f(-0.00000f, +0.00310f, +0.00380f);
   //V28
glNormal3f(0.50f, 0.86f, 0.0f);
   //V9
   //V30
   //V10
   //V32
   //V11
   //V34
   //V12
   //V36
   //V7
   //V26
   glEnd();
glRotatef(30.0f, 0.0f, 0.0f, 1.0f);
   glEndList();
}

GLvoid BuildListsoff()
{
    range = 1;
    list=glGenLists(1);
    // Generate 5 Lists
glNewList(1,GL_COMPILE);
    glRotatef(-30.0f, 0.0f, 0.0f, 1.0f);
    glBegin(GL_TRIANGLE_FAN);
    glVertex3f(+0.00245f, +0.00000f, +0.00000f);
    //Va
```
glNormal3f(-0.92f, 0.35f, 0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13

glVertex3f(+0.00265f, +0.00155f, -0.00215f);
   //V1

glNormal3f(-1.00f, 0.00f, -0.9f);
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
   //V6

glNormal3f(-0.92f, -0.35f, 0.17f);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
   //V18

glNormal3f(-0.92f, -0.35f, -0.17f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
   //V12

glNormal3f(-1.00f, 0.00f, 0.9f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7

glNormal3f(-0.92f, 0.35f, -0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13

gEnd();

glBegin(GL_QUADS);
glNormal3f(-0.35f, -0.60f, 0.73f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
   //V1

gVertex3f(-0.00000f, +0.00310f, -0.00215f);
   //V2

gVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14

gVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13

gEnd();

glBegin(GL_QUADS);
glNormal3f(-0.35f, -0.59f, -0.73f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14

gVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8

gVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7

gVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13

gEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.001325f, +0.002325f, +0.00000f);
   //Vb

glNormal3f(0.20f, -0.95f, 0.22f);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
   //V15

gVertex3f(-0.00265f, +0.00155f, -0.00215f);
   //V3

glNormal3f(0.5f, -0.86f, 0.0f);
glVertex3f(-0.00000f, +0.00310f, -0.00215f);
   //V2

glNormal3f(0.73f, -0.65f, 0.22f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14

glNormal3f(0.73f, -0.65f, -0.22f);
Appendices – Appendix 5

```c
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
//V8
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
//V9
glNormal3f(0.20f, -0.95f, -0.22f);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
//V15
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, 0.73f);
glVertex3f(-0.00000f, +0.00155f, +0.00000f);
//V3
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
//V4
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
//V16
glVertex3f(-0.00495f, +0.00155f, +0.00215f);
//V10
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, -0.73f);
glVertex3f(-0.00495f, -0.00155f, +0.00215f);
//V16
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
//V15
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
//V9
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
//V15
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.001325f, -0.002325f, +0.00000f);
//Vc
glNormal3f(0.73f, 0.65f, 0.22f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
//V17
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
//V5
glNormal3f(0.50f, 0.86f, 0.00f);
glVertex3f(-0.00265f, -0.00155f, -0.00215f);
//V4
glNormal3f(0.20f, 0.95f, 0.22f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);
//V16
glNormal3f(0.20f, 0.95f, -0.22f);
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
//V10
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
//V11
glNormal3f(0.82f, 0.46f, -0.34f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
//V17
glEnd();

glBegin(GL_QUADS);
glNormal3f(-0.34f, 0.60f, 0.72f);
```

320
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
  //V5
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
  //V6
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
  //V18
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
  //V17
glEnd();

// 6 bond lines representing 2 coordinate side growth
glBegin(GL_QUADS);  // [corrected]
glNormal3f(-0.35f, 0.60f, -0.72f);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
  //V18
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
  //V17
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
  //V11
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
  //V12
glEnd();

// six ring connector
glBegin(GL_QUAD_STRIP);
glNormal3f(-0.50f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V1
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V25
glVertex3f(-0.00000f, +0.00310f, -0.00215f);
    //V2
glVertex3f(-0.00000f, +0.00310f, -0.00380f);
    //V27
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    //V3
glVertex3f(-0.00265f, +0.00155f, -0.00380f);
    //V29
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
    //V4
glVertex3f(-0.00000f, -0.00310f, -0.00380f);
    //V31
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
    //V5
glVertex3f(-0.00000f, -0.00310f, -0.00380f);
    //V33
glNormal3f(-0.50f, 0.86f, 0.0f);
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
    //V6
glVertex3f(+0.00265f, -0.00155f, -0.00380f);
    //V35
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V7
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V37
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V8
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V38
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    //V9
glVertex3f(-0.00000f, +0.00310f, +0.00380f);
    //V39
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
    //V10
glVertex3f(-0.00265f, +0.00155f, +0.00380f);
    //V40
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
    //V11

322
glVertex3f(-0.00000f, -0.00310f, +0.00380f);
  //V34
glNormal3f(-0.50f, 0.86f, 0.0f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
  //V12
glVertex3f(+0.00265f, -0.00155f, +0.00380f);
  //V36
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
  //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
  //V26
glEnd();

glRotatef(30.0f, 0.0f, 0.0f, 1.0f);
glEndList();

GLvoid BuildListseri()
{
  range = 2;
  list=glGenLists(2);
  // Generate 5 Lists
  glNewList(1,GL_COMPILE);
  glRotatef(-30.0f, 0.0f, 0.0f, 1.0f);
  glBegin(GL_TRIANGLES);
  glVertex3f(+0.00265f, +0.00000f, +0.00000f);
    //Va
glNormal3f(-0.92f, 0.35f, 0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
    //V13
  glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V1
  glNormal3f(-1.00f, 0.00f, -0.09f);
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
    //V6
  glNormal3f(-0.92f, -0.35f, 0.17f);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    //V18
  glNormal3f(-0.92f, -0.35f, -0.17f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    //V12
  glNormal3f(-1.00f, 0.00f, 0.09f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
    //V7
  glNormal3f(-0.92f, 0.35f, -0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
    //V13
glEnd();

  glBegin(GL_QUADS);
  glVertex3f(+0.00265f, -0.00310f, +0.00380f);
    //V34
  glVertex3f(-0.00000f, +0.00310f, -0.00380f);
    //V2
  glVertex3f(+0.00000f, +0.00310f, -0.00380f);
    //V2
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
    //V14
}
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13
glEnd();

.glBegin(GL_QUADS);
glNormal3f(-0.35f, -0.59f, -0.73f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14
.glVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8
.glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7
.glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13
.glEnd();

.glBegin(GL_QUADS);
glNormal3f(-0.35f, -0.59f, -0.73f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14
.glVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8
.glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7
.glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13
.glEnd();

.glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.001325f, +0.002325f, +0.00000f);
   //Vb
.glNormal3f(0.20f, -0.95f, 0.22f);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
   //V15
.glVertex3f(-0.00265f, +0.00155f, -0.00215f);
   //V3
.glNormal3f(0.5f, -0.86f, 0.0f);
glVertex3f(-0.00000f, +0.00310f, -0.00215f);
   //V2
.glNormal3f(0.73f, -0.65f, 0.22f);
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
   //V14
.glNormal3f(0.73f, -0.65f, -0.22f);
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8
.glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
   //V9
.glNormal3f(0.20f, -0.95f, -0.22f);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
   //V15
.glEnd();

.glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, 0.73f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
   //V4
.glVertex3f(-0.00265f, -0.00155f, -0.00215f);
   //V3
.glVertex3f(-0.00495f, -0.00155f, +0.00000f);
   //V16
.glVertex3f(-0.00495f, +0.00155f, +0.00000f);
   //V15
.glEnd();

.glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, -0.73f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);
   //V16
.glVertex3f(-0.00495f, +0.00155f, +0.00000f);
   //V15
.glVertex3f(-0.00265f, +0.00155f, +0.00215f);
   //V9
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
  //V10
glEnd();

glBegin(GL_TRIANGLES);
glVertex3f(-0.001325f, -0.002325f, +0.00000f);
  //Vc
glNormal3f(0.20f, 0.95f, 0.22f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
  //V17
// 6 bond lines representing 2 coordinate side growth
//  [corrected]
glBegin(GL_LINES);
  glVertex3f(+0.00110f, +0.00505f, +0.00000f);
      //V14
glVertex3f(+0.00110f, +0.00660f, +0.00000f);
      //V19
  glEnd();  // [/corrected]

//  [corrected]
glBegin(GL_LINES);

//  [corrected]
glBegin(GL_QUADS);
  glEnd();

//  [corrected]
glBegin(GL_LINES);

//  [corrected]
glBegin(GL_QUADS);
  glEnd();

//  [corrected]
glBegin(GL_TRIANGLES);
  glEnd();

//  [corrected]
glBegin(GL_TRIANGLES);
  glEnd();

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]

//  [corrected]
Appendices – Appendix 5

```c
// six ring connector

glBegin(GL_QUAD_STRIP);
glNormal3f(-0.5f, -0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
// V5

glVertex3f(-0.00265f, +0.00155f, -0.00380f);
// V29

glNormal3f(0.5f, 0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00000f);
// V31

glNormal3f(0.5f, 0.86f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, -0.00380f);
// V33

glNormal3f(-0.5f, -0.86f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, -0.00380f);
// V35
```
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
   //V1
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
   //V25
glEnd();

glBegin(GL_QUAD_STRIP);
glNormal3f(-0.5f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
   //V26
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
   //V8
glVertex3f(-0.00000f, +0.00310f, +0.00380f);
   //V28
glNormal3f(0.5f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
   //V9
glVertex3f(-0.00265f, +0.00155f, +0.00380f);
   //V30
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
   //V10
glVertex3f(-0.00265f, -0.00155f, +0.00380f);
   //V32
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
   //V11
glVertex3f(-0.00000f, -0.00310f, +0.00380f);
   //V34
glNormal3f(-0.50f, 0.86f, 0.0f);
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
   //V12
glVertex3f(+0.00265f, -0.00155f, +0.00380f);
   //V36
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
   //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
   //V26

glEnd();
glRotatef(30.0f, 0.0f, 0.0f, 1.0f);
glEndList();

.glNewList(2, GL_COMPILE);
glRotatef(30.0f, 0.0f, 0.0f, 1.0f);

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.00245f, +0.00000f, +0.00000f);
   //Va
glNormal3f(-0.92f, 0.35f, 0.17f);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
   //V13
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
   //V1
glNormal3f(-1.00f, 0.00f, -0.09f);
Appendices – Appendix 5

```
void App5() {
    glVertex3f(+0.00265f, -0.00155f, -0.00215f);
    glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    glVertex3f(+0.00380f, +0.00350f, +0.00000f);
    glVertex3f(+0.00265f, +0.00155f, +0.00215f);
    glEnd();

    glBegin(GL_QUADS);
    glVertex3f(+0.00110f, +0.00505f, +0.00000f);
    glVertex3f(+0.00265f, -0.00155f, -0.00215f);
    glVertex3f(+0.00380f, +0.00350f, +0.00000f);
    glVertex3f(+0.00265f, +0.00155f, +0.00215f);
    glEnd();

    glBegin(GL_QUADS);
    glVertex3f(-0.00265f, +0.00155f, +0.00215f);
    glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    glVertex3f(-0.00310f, +0.00000f, +0.00000f);
    glVertex3f(-0.00265f, +0.00155f, +0.00215f);
    glEnd();

    glBegin(GL_QUADS);
    glVertex3f(-0.001325f, +0.002325f, +0.00000f);
    glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    glVertex3f(-0.00310f, +0.00000f, +0.00215f);
    glVertex3f(-0.00265f, +0.00155f, -0.00215f);
    glEnd();

    glBegin(GL_TRIANGLES);
    glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    glVertex3f(-0.00495f, +0.00155f, +0.00000f);
    glVertex3f(-0.001325f, +0.002325f, +0.00000f);
    glVertex3f(-0.00000f, +0.00310f, +0.00215f);
    glVertex3f(-0.00000f, -0.00265f, -0.00000f);
    glVertex3f(-0.00000f, +0.00310f, +0.00000f);
    glEnd();
```

glVertex3f(-0.00495f, +0.00155f, +0.00000f);

// V15

glEnd();

glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, 0.73f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);

// V3

glVertex3f(-0.00265f, -0.00155f, -0.00215f);

// V4

glVertex3f(-0.00495f, -0.00155f, +0.00000f);

// V16

glVertex3f(-0.00495f, +0.00155f, +0.00000f);

// V15

glEnd();

glBegin(GL_QUADS);
glNormal3f(0.68f, 0.0f, -0.73f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);

// V16

glVertex3f(-0.00495f, +0.00155f, +0.00000f);

// V15

glVertex3f(-0.00265f, +0.00155f, +0.00215f);

// V9

glVertex3f(-0.00265f, -0.00155f, +0.00215f);

// V16

glVertex3f(-0.00000f, -0.00310f, -0.00215f);

// V10

glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.001325f, -0.002325f, +0.00000f);

// Vc

glNormal3f(0.73f, 0.65f, 0.22f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);

// V17

glVertex3f(-0.00000f, -0.00310f, -0.00215f);

// V5

glNormal3f(0.50f, 0.86f, 0.00f);
glVertex3f(-0.00265f, -0.00155f, -0.00215f);

// V4

glNormal3f(0.20f, 0.95f, 0.22f);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);

// V16

glNormal3f(0.20f, 0.95f, -0.22f);
glVertex3f(-0.00265f, -0.00155f, +0.00215f);

// V10

glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);

// V11

glNormal3f(0.82f, 0.46f, -0.34f);
glVertex3f(+0.00110f, -0.00505f, +0.00000f);

// V17

glEnd();

glBegin(GL_QUADS);
glNormal3f(-0.34f, 0.60f, 0.72f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);

// V5

glVertex3f(+0.00265f, -0.00155f, -0.00215f);

// V6

glVertex3f(+0.00380f, -0.00350f, +0.00000f);

// V18
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    //V17
glEnd();

glBegin(GL_QUADS);
glNormal3f(-0.35f, 0.60f, -0.72f);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    //V18
glVertex3f(+0.00110f, -0.00505f, +0.00000f);
    //V17
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
    //V11
glVertex3f(+0.00265f, -0.00155f, +0.00215f);
    //V12
glEnd();

// 6 bond lines representing 2 coordinate side growth
glBegin(GL_LINES);   // [corrected]
glVertex3f(+0.00110f, +0.00505f, +0.00000f);
    //V14
glVertex3f(+0.00110f, +0.00660f, +0.00000f);
    //V19
glEnd();    // [/corrected]

glBegin(GL_LINES);
glVertex3f(+0.00380f, +0.00350f, +0.00000f);
    //V13
glVertex3f(+0.00535f, +0.00440f, +0.00000f);
    //V20
glEnd();

glBegin(GL_LINES);
glVertex3f(-0.00495f, -0.00155f, +0.00000f);
    //V16
glVertex3f(-0.00620f, -0.00215f, +0.00000f);
    //V21
glEnd();

glBegin(GL_LINES);
glVertex3f(-0.00495f, +0.00155f, +0.00000f);
    //V15
glVertex3f(-0.00620f, +0.00215f, +0.00000f);
    //V22
glEnd();

glBegin(GL_LINES);
glVertex3f(+0.00380f, -0.00350f, +0.00000f);
    //V18
glVertex3f(+0.00535f, -0.00440f, +0.00000f);
    //V23
glEnd();

// six ring connector
glBegin(GL_QUAD_STRIP);
glNormal3f(-0.50f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
    //V1
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
    //V25
glVertex3f(-0.00000f, +0.00310f, -0.00215f);
     //V2
glVertex3f(-0.00000f, +0.00310f, -0.00380f);
     //V27
glNormal3f(0.50f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, -0.00215f);
     //V3
glVertex3f(-0.00265f, +0.00155f, -0.00380f);
     //V29
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, -0.00215f);
     //V4
glVertex3f(-0.00265f, -0.00155f, -0.00380f);
     //V31
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, -0.00215f);
     //V5
glVertex3f(-0.00000f, -0.00310f, -0.00380f);
     //V33
glNormal3f(-0.50f, 0.86f, 0.0f);
glVertex3f(+0.00265f, -0.00155f, -0.00215f);
     //V6
glVertex3f(+0.00265f, -0.00155f, -0.00380f);
     //V35
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, -0.00215f);
     //V1
glVertex3f(+0.00265f, +0.00155f, -0.00380f);
     //V25
glEnd();

glBegin(GL_QUAD_STRIP);
glNormal3f(-0.5f, -0.86f, 0.0f);
glVertex3f(+0.00265f, +0.00155f, +0.00215f);
     //V7
glVertex3f(+0.00265f, +0.00155f, +0.00380f);
     //V26
glVertex3f(-0.00000f, +0.00310f, +0.00215f);
     //V8
glVertex3f(-0.00000f, +0.00310f, +0.00380f);
     //V28
glNormal3f(0.5f, -0.86f, 0.0f);
glVertex3f(-0.00265f, +0.00155f, +0.00215f);
     //V9
glVertex3f(-0.00265f, +0.00155f, +0.00380f);
     //V30
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.00265f, -0.00155f, +0.00215f);
     //V10
glVertex3f(-0.00265f, -0.00155f, +0.00380f);
     //V32
glNormal3f(0.50f, 0.86f, 0.0f);
glVertex3f(-0.00000f, -0.00310f, +0.00215f);
     //V11
glVertex3f(-0.00000f, -0.00310f, +0.00380f);
     //V34
glNormal3f(-0.50f, 0.86f, 0.0f);

```c
void BuildLists2DSod()
{
    range = 4;
    list = glGenLists(1);
    // Generate 2 Lists
    glNewList(1, GL_COMPILE);
    // Start With The first List
    glColor4f(0.55f, 0.28f, 0.10f, 1.0f);
    glRotatef(-45.0f, 1.0f, 0.0f, 0.0f);
    glBegin(GL_QUADS);
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(-0.0084955f, +0.0042475f, -0.0042475f);
    //V1
    glVertex3f(-0.0084955f, +0.0042475f, +0.0042475f);
    //V2
    glVertex3f(-0.0084955f, -0.0042475f, +0.0042475f);
    //V3
    glVertex3f(-0.0084955f, +0.0042475f, -0.0042475f);
    //V4
    glEnd();
    glBegin(GL_QUADS);
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(+0.0042475f, -0.0084955f, +0.0042475f);
    //V5
    glVertex3f(+0.0042475f, +0.0042475f, +0.0042475f);
    //V6
    glVertex3f(+0.0042475f, -0.0084955f, -0.0042475f);
    //V7
    glVertex3f(+0.0042475f, +0.0042475f, -0.0042475f);
    //V8
    glEnd();
    glBegin(GL_QUADS);
    glNormal3f(0.0f, 1.0f, 0.0f);
    glVertex3f(-0.0000000f, -0.0084955f, -0.0084955f);
    //V9
    glVertex3f(+0.0042475f, -0.0084955f, -0.0084955f);
    //V10
    glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
    //V11
    glVertex3f(-0.0042475f, -0.0084955f, +0.0042475f);
    //V12
    glEnd();
    glBegin(GL_QUADS);
```
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
    //V13
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
    //V14
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
    //V16
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, 1.0f);
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
    //V17
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
    //V18
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
    //V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
    //V20
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
    //V21
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
    //V22
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
    //V23
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
    //V24
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, +0.0042475f, +0.0032475f);
glNormal3f(0.58f, -0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
    //V16
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
    //V1
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
    //V2
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
    //V24
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
    //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, +0.0042475f, +0.0032475f);
glNormal3f(-0.58f, -0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
    //V14
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
  //V5
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V6
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
  //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
  //V15
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, +0.0042475f, -0.0032475f);
  //V23
glNormal3f(-0.58f, -0.58f, 0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
  //V24
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
  //V2
}
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
  //V12
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  //V11
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);

glBegin(GL_TRIANGLES);
glVertex3f(+0.0032475f, -0.0042475f, +0.0032475f);
glNormal3f(-0.58f, 0.58f, -0.58f);
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  //V23
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  //V11
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  //V7
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V6
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22

glEnd();

// Additional code segments follow...
Appendices – Appendix 5

```c
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
//V19

//V1
glEnd();
glRotatef(45.0f, 1.0f, 0.0f, 0.0f);
glEndList();

gNewList(2, GL_COMPILE);
// Start With The first List
glColor4f(0.55f, 0.48f, 0.20f, 1.0f);
glRotatef(-45.0f, 1.0f, 0.0f, 0.0f);

glBegin(GL_QUADS);
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
//V1
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
//V2
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
//V3
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
//V4
glBegin(GL_QUADS);
glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
//V5
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
//V6
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
//V7
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
//V8
glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
//V9
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
//V10
gVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
//V11
gVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
//V12
glEnd();

gBegin(GL_QUADS);
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
//V13
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
//V14
gVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
//V15
gVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
//V16
glEnd();
gBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, 1.0f);
```

336
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
   //V17
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
   //V18
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
   //V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
   //V20
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
   //V21
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
   //V22
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
   //V23
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
   //V24
glEnd();

glBegin(GL_TRIANGLE_FAN);
glNormal3f(-0.58f, -0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
   //V14
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
   //V5
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
   //V6
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
   //V22
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
   //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glEnd();

glBegin(GL_TRIANGLE_FAN);
glNormal3f(-0.58f, -0.58f, -0.58f);
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
   //V22
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
   //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glEnd();

glBegin(GL_TRIANGLE_FAN);

337
Appendices – Appendix 5

```c
void draw_polygon()
{
    glBegin(GL_TRIANGLES);
    glVertex3f(+0.0032475f, +0.0042475f, -0.0032475f);
    glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
    glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
    glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
    glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
    glVertex3f(+0.0000000f, +0.0000000f, +0.0042475f);
    glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
    glVertex3f(+0.0000000f, +0.0084955f, +0.0042475f);
    glVertex3f(-0.0032475f, +0.0000000f, +0.0084955f);
    glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
    glVertex3f(-0.0042475f, +0.0000000f, -0.0084955f);
    glVertex3f(-0.0084955f, +0.0042475f, +0.0000000f);
    glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
    glVertex3f(-0.0000000f, +0.0000000f, +0.0042475f);
    glEnd();
}
```
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  //V11
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  //V10
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V7
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V6
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, -0.0042475f, -0.0032475f);
glNormal3f(-0.58f, 0.58f, 0.58f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V8
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
  //V18
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
  //V12
glVertex3f(+0.0000000f, -0.0042475f, -0.0000000f);
  //V9
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  //V10
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V7
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, -0.0042475f, -0.0032475f);
glNormal3f(0.58f, 0.58f, 0.58f);
glVertex3f(-0.0000000f, -0.0042475f, -0.0084955f);
  //V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
  //V20
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
  //V4
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
  //V3
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
  //V12
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  //V9
glVertex3f(-0.0000000f, -0.0042475f, -0.0084955f);
  //V19
glEnd();
glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

gEndList();
gList(3, GL_COMPILE);
    // Start With The first List
groundColor(0.70f, 0.48f, 0.10f, 1.0f);
gRotatef(-45.0f, 1.0f, 0.0f, 0.0f);

glBegin(GL_QUADS);
gNormal3f(1.0f, 0.00f, 0.00f);
gVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
  //V1
Appendices – Appendix 5

```c
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f); //V2
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f); //V3
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f); //V4
glEnd();
gBegin(GL_QUADS);
gNormal3f(-1.0f, 0.0f, 0.0f);
gVertex3f(+0.0084955f, +0.0042475f, -0.0000000f); //V5
gVertex3f(+0.0084955f, +0.0000000f, +0.0042475f); //V6
gVertex3f(+0.0084955f, -0.0042475f, +0.0000000f); //V7
gVertex3f(+0.0084955f, +0.0000000f, -0.0042475f); //V8
gEnd();
gBegin(GL_QUADS);
gNormal3f(0.0f, 1.0f, 0.0f);
gVertex3f(-0.0000000f, -0.0084955f, -0.0042475f); //V9
gVertex3f(+0.0042475f, -0.0084955f, -0.0000000f); //V10
gVertex3f(-0.0000000f, -0.0084955f, +0.0042475f); //V11
gVertex3f(-0.0042475f, -0.0084955f, +0.0000000f); //V12
gEnd();
gBegin(GL_QUADS);
gNormal3f(0.0f, -1.0f, 0.0f);
gVertex3f(-0.0000000f, +0.0084955f, -0.0042475f); //V13
gVertex3f(+0.0042475f, +0.0084955f, -0.0000000f); //V14
gVertex3f(-0.0000000f, +0.0084955f, +0.0042475f); //V15
gVertex3f(-0.0042475f, +0.0084955f, +0.0000000f); //V16
gEnd();
gBegin(GL_QUADS);
gNormal3f(0.0f, 0.0f, 1.0f);
gVertex3f(-0.0000000f, +0.0042475f, -0.0084955f); //V17
gVertex3f(+0.0042475f, -0.0000000f, -0.0084955f); //V18
gVertex3f(+0.0000000f, -0.0042475f, -0.0084955f); //V19
gVertex3f(-0.0042475f, -0.0000000f, -0.0084955f); //V20
gEnd();
gBegin(GL_QUADS);
gNormal3f(0.0f, 0.0f, -1.0f);
gVertex3f(-0.0000000f, +0.0042475f, +0.0084955f); //V21
```

340
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f); //V22
glVertex3f(+0.0084955f, +0.0042475f, +0.0084955f); //V23
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f); //V24
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, +0.0042475f, +0.0032475f);
glNormal3f(0.58f, -0.58f, -0.58f);
glVertex3f(-0.0084955f, +0.0042475f, +0.0084955f); //V15
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f); //V16
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f); //V21

glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, +0.0042475f, +0.0032475f);
glNormal3f(-0.58f, -0.58f, -0.58f);
//V15
//V14
//V5
//V6
//V22
//V21
//V15
glEnd();

//V17
//V18
//V8
//V5
glVertex3f(+0.0032475f, +0.0042475f, -0.0000000f);
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, +0.0042475f, -0.0000000f);
glNormal3f(0.58f, -0.58f, 0.58f);
glVertex3f(+0.0000000f, +0.0084955f, +0.0042475f);
  //V13
  glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
  //V16
  glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
  //V1
  glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
  //V4
  glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
  //V20
  glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
  //V17
  glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
  //V13
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, +0.0042475f, +0.0032475f);
glNormal3f(-0.58f, 0.58f, -0.58f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  //V23
  glVertex3f(+0.0042475f, +0.0000000f, +0.0084955f);
  //V24
  glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V2
  glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V3
  glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
  //V12
  glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  //V11
  glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  //V22
  glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, -0.0042475f, +0.0032475f);
glNormal3f(-0.58f, 0.58f, -0.58f);
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22
  glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  //V23
  glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  //V11
  glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
  //V10
  glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V7
  glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
  //V6
  glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
  //V22
  glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(+0.0032475f, -0.0042475f, -0.0032475f);
glNormal3f(-0.58f, 0.58f, 0.58f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V7
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
  //V8
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
  //V18
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  //V19
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
  //V7
glEnd();

 glBegin(GL_TRIANGLE_FAN);
  glVertex3f(-0.0032475f, -0.0042475f, -0.0032475f);
  glNormal3f(0.58f, 0.58f, 0.58f);
  glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
  //V19
  glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
  //V20
  glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
  //V4
  glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
  //V3
  glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
  //V12
  glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  //V9
  glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
  //V19
  glEnd();
  glRotatef(45.0f, 1.0f, 0.0f, 0.0f);
 glEndList();

  glNewList(4, GL_COMPILE);
  // Start With The first List
  glColor4f(0.80f, 0.68f, 0.35f, 1.0f);
  glRotatef(-45.0f, 1.0f, 0.0f, 0.0f);
  glBegin(GL_QUADS);
   glNormal3f(1.0f, 0.00f, 0.00f);
   glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
   //V1
   glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
   //V2
   glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
   //V3
   glVertex3f(-0.0084955f, -0.0042475f, -0.0000000f);
   //V4
 glEnd();

  glBegin(GL_QUADS);
   glNormal3f(-1.0f, 0.0f, 0.0f);
   glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
   //V5
   glVertex3f(+0.0084955f, +0.0042475f, +0.0000000f);
   //V6

glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
    //V7
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
    //V8
glBegin(GL_QUADS);
    glEnd();

glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
    //V9
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
    //V10
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
    //V11
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
    //V12
glEnd();

glBegin(GL_QUADS);
    glEnd();

glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
    //V13
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
    //V14
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
    //V15
glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
    //V16
glEnd();

glBegin(GL_QUADS);
    glEnd();

glNormal3f(0.0f, 0.0f, 1.0f);
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
    //V17
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
    //V18
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
    //V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
    //V20
glEnd();

glBegin(GL_QUADS);
    glEnd();

glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
    //V21
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
    //V22
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
    //V23
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
    //V24
glEnd();

glBegin(GL_TRIANGLE_FAN);
    glEnd();
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
   //V1
glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
   //V2
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
   //V24
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
   //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glEnd();

 glBegin(GL_TRIANGLE_FAN);
 glVertex3f(+0.0032475f, +0.0042475f, +0.0032475f);
 glNormal3f(-0.58f, -0.58f, -0.58f);
 glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glVertex3f(+0.0084955f, +0.0084955f, -0.0000000f);
   //V14
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
   //V5
glVertex3f(+0.0042475f, +0.0000000f, +0.0042475f);
   //V6
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
   //V22
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
   //V21
glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
   //V15
glEnd();

 glBegin(GL_TRIANGLE_FAN);
 glVertex3f(+0.0032475f, +0.0042475f, +0.0032475f);
 glNormal3f(-0.58f, -0.58f, 0.58f);
 glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
   //V14
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
   //V13
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
   //V17
glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
   //V18
glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
   //V8
glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
   //V5
glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
glEnd();

 glBegin(GL_TRIANGLE_FAN);
 glVertex3f(-0.0032475f, +0.0042475f, -0.0032475f);
 glNormal3f(0.58f, -0.58f, 0.58f);
 glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
   //V13
glVertex3f(-0.0000000f, +0.0084955f, +0.0000000f);
   //V16
glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
   //V1
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
   //V4
Appendices – Appendix 5

```c
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
//V20
glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
//V17
glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
//V13
glEnd();

glBegin(GL_TRIANGLES);  
glVertex3f(-0.0032475f, -0.0042475f, +0.0032475f);  
glNormal3f(0.58f, 0.58f, -0.58f);  
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);  
//V23

glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);  
//V24

glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
//V2

glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
//V3

glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
//V12

glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
//V11

glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
//V10

glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
//V7

glVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
//V8

glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
//V18

glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
//V19

glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
//V9

glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
//V10
```
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
//V7
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(-0.0032475f, -0.0042475f, -0.0032475f);
glNormal3f(0.58f, 0.58f, 0.58f);
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
//V19
glVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
//V20
glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
//V3
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
//V12
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
//V9
glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
//V19
glEnd();
glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

glEndList();

//------------------------------------------------------------------
//---------------------------------------------------------------

GLvoid BuildLists1ta()
{
  range = 5;
  list=glGenLists(5);
      // Generate 5 Lists
  glNewList(1, GL_COMPILE);
      // Compile the SOD cage
  glBegin(GL_QUADS);
  glNormal3f(-1.0f, 0.0f, 0.0f);
  glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
      //V1
  glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
      //V2
  glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
      //V3
  glVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
      //V4
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(1.0f, 0.0f, 0.0f);
  glVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
      //V5
  glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
      //V6
  glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
      //V7
  glVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
      //V8

  glEndList();
}
 Appendix 5

```gl
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);
  // V9
  glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  // V10
  glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
  // V11
  glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
  // V12
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(0.0f, 1.0f, 0.0f);
  glVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
  // V13
  glVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
  // V14
  glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
  // V15
  glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
  // V16
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, -1.0f);
  glVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
  // V17
  glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
  // V18
  glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
  // V19
  glVertex3f(-0.0042475f, -0.0084955f, -0.0000000f);
  // V20
  glEnd();

  glBegin(GL_QUADS);
  glNormal3f(0.0f, 0.0f, 1.0f);
  glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
  // V21
  glVertex3f(+0.0042475f, -0.0084955f, +0.0000000f);
  // V22
  glVertex3f(+0.0000000f, -0.0042475f, -0.0084955f);
  // V23
  glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
  // V24
  glEnd();

  glBegin(GL_POLYGON);
  glNormal3f(-0.58f, 0.58f, 0.58f);
  glVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
  glVertex3f(-0.0042475f, +0.0084955f, +0.0000000f);
  glVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
  glVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
```

348
glVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
glVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
glEnd();
gBegin(GL_POLYGON);
gNormal3f(0.58f, 0.58f, 0.58f);
gVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
gVertex3f(+0.0042475f, +0.0084955f, -0.0000000f);
gVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
gVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);
gVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
gVertex3f(-0.0000000f, +0.0042475f, +0.0084955f);
gEnd();

glBegin(GL_POLYGON);
gNormal3f(-0.58f, 0.58f, 0.58f);
gVertex3f(+0.0042475f, +0.0084955f, -0.0042475f);
gVertex3f(-0.0000000f, +0.0084955f, -0.0042475f);
gVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
gVertex3f(+0.0042475f, -0.0000000f, -0.0084955f);
gVertex3f(+0.0084955f, +0.0000000f, -0.0042475f);
gVertex3f(+0.0084955f, +0.0042475f, -0.0000000f);
gEnd();

glBegin(GL_POLYGON);
gNormal3f(-0.58f, -0.58f, -0.58f);
gVertex3f(-0.0042475f, +0.0084955f, +0.0042475f);
gVertex3f(-0.0000000f, +0.0084955f, +0.0042475f);
gVertex3f(-0.0084955f, +0.0042475f, -0.0000000f);
gVertex3f(-0.0084955f, +0.0000000f, -0.0042475f);
gVertex3f(-0.0042475f, -0.0000000f, -0.0084955f);
gVertex3f(-0.0000000f, +0.0042475f, -0.0084955f);
gEnd();

glBegin(GL_POLYGON);
gNormal3f(-0.58f, 0.58f, 0.58f);
gVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
gVertex3f(-0.0042475f, -0.0000000f, +0.0084955f);
gVertex3f(-0.0084955f, -0.0000000f, +0.0042475f);
gVertex3f(-0.0084955f, +0.0000000f, +0.0042475f);
glVertex3f(-0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(-0.0042475f, -0.0084955f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glEnd();

 glBegin(GL_POLYGON);
glVertex3f(+0.0042475f, -0.0000000f, +0.0084955f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);

 glEnd();

 glBegin(GL_POLYGON);
 glNormal3f(0.58f, -0.58f, 0.58f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glVertex3f(+0.0042475f, -0.0084955f, -0.0000000f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(+0.0084955f, +0.0000000f, +0.0042475f);

 glEnd();

 glBegin(GL_POLYGON);
 glNormal3f(-0.58f, -0.58f, -0.58f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glVertex3f(+0.0042475f, -0.0042475f, 0.0000000f);
glVertex3f(-0.0000000f, -0.0042475f, -0.0084955f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);

 glEnd();

 glBegin(GL_POLYGON);
 glNormal3f(-0.58f, -0.58f, -0.58f);
glVertex3f(+0.0000000f, -0.0042475f, +0.0084955f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, +0.0042475f);
glVertex3f(+0.0042475f, -0.0042475f, 0.0000000f);
glVertex3f(-0.0000000f, -0.0042475f, -0.0084955f);
glVertex3f(+0.0084955f, -0.0042475f, +0.0000000f);
glVertex3f(-0.0000000f, -0.0084955f, -0.0042475f);

 glEnd();

 glutEndList();

glNewList(2,GL_COMPILE);
// Compile the D4R cage

 glBegin(GL_QUADS);
glNormal3f(0.0f, 1.0f, 1.0f);
glVertex3f(-0.003003f, -0.0000000f, +0.0042475f);
// V1

glVertex3f(+0.003003f, -0.0000000f, +0.0042475f);
// V2

glVertex3f(+0.003003f, +0.0042475f, +0.0000000f);
// V4

glVertex3f(-0.003003f, +0.0042475f, +0.0000000f);
// V3

glNormal3f(0.0f, 1.0f, -1.0f);
glVertex3f(-0.003003f, +0.0042475f, +0.0000000f);
// V3

glVertex3f(+0.003003f, +0.0042475f, +0.0000000f);
// V4

glVertex3f(+0.003003f, +0.0000000f, -0.0042475f);
// V6

glVertex3f(-0.003003f, +0.0000000f, -0.0042475f);
// V5

glNormal3f(0.0f, -1.0f, -1.0f);
glVertex3f(-0.003003f, -0.0042475f, +0.0000000f);
// V5

glVertex3f(+0.003003f, -0.0042475f, +0.0000000f);
// V6

glVertex3f(+0.003003f, -0.0000000f, -0.0042475f);
// V8

glVertex3f(-0.003003f, -0.0042475f, -0.0000000f);
// V7

glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(+0.003003f, +0.0042475f, +0.0000000f);
// V4

glVertex3f(+0.003003f, +0.0000000f, -0.0042475f);
// V6

glVertex3f(+0.003003f, -0.0042475f, -0.0000000f);
// V8

glVertex3f(+0.003003f, -0.0000000f, +0.0042475f);
// V10

glVertex3f(-0.003003f, +0.0000000f, +0.0042475f);
// V9

glNormal3f(-1.0f, 0.0f, 0.0f);
glVertex3f(+0.003003f, +0.0000000f, +0.0042475f);
// V4

glVertex3f(-0.003003f, +0.0042475f, +0.0000000f);
// V6

glVertex3f(-0.003003f, +0.0000000f, -0.0042475f);
// V8

glVertex3f(-0.003003f, -0.0042475f, -0.0000000f);
// V10

// Compile the D4R cage

// V1

glBegin(GL_QUADS);

glEnd();
gEndList();

gNewList(3, GL_COMPILE);
// Compile the D4R cage

gBegin(GL_QUADS);

351
glNormal3f(-1.0f, 0.0f, 1.0f);
glVertex3f(-0.0000000f, -0.003003f, +0.0042475f);
  //V1
glVertex3f(+0.0000000f, +0.003003f, +0.0042475f);
  //V2
glVertex3f(-0.0042475f, +0.003003f, +0.0000000f);
  //V4
glVertex3f(-0.0042475f, -0.003003f, +0.0000000f);
  //V3
glNormal3f(-1.0f, 0.0f, -1.0f);
glVertex3f(-0.0042475f, -0.003003f, +0.0000000f);
  //V3
glVertex3f(-0.0042475f, +0.003003f, +0.0000000f);
  //V4
glVertex3f(+0.0000000f, +0.003003f, -0.0042475f);
  //V6
glVertex3f(-0.0000000f, -0.003003f, -0.0042475f);
  //V5
glNormal3f(1.0f, 0.0f, -1.0f);
glVertex3f(-0.0042475f, -0.003003f, -0.0042475f);
  //V5
glVertex3f(+0.0000000f, +0.003003f, -0.0042475f);
  //V6
glVertex3f(+0.0042475f, +0.003003f, -0.0000000f);
  //V8
glVertex3f(+0.0042475f, -0.003003f, -0.0000000f);
  //V7
glNormal3f(1.0f, 0.0f, 1.0f);
glVertex3f(+0.0042475f, -0.003003f, -0.0000000f);
  //V7
glVertex3f(+0.0042475f, +0.003003f, -0.0000000f);
  //V8
glVertex3f(+0.0000000f, +0.003003f, +0.0042475f);
  //V10
glVertex3f(-0.0000000f, -0.003003f, +0.0042475f);
  //V9
glNormal3f(0.0f, 1.0f, 0.0f);
glVertex3f(+0.0000000f, +0.003003f, +0.0042475f);
  //V10
glVertex3f(+0.0042475f, +0.003003f, -0.0000000f);
  //V8
glVertex3f(+0.0000000f, +0.003003f, -0.0042475f);
  //V6
glVertex3f(-0.0042475f, +0.003003f, +0.0000000f);
  //V4
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(+0.0000000f, -0.003003f, +0.0042475f);
  //V10
glVertex3f(+0.0042475f, -0.003003f, -0.0000000f);
  //V8
glVertex3f(+0.0000000f, -0.003003f, -0.0042475f);
  //V6
glVertex3f(-0.0042475f, -0.003003f, +0.0000000f);
  //V4
glEnd();
glEndList();

glNewList(4, GL_COMPILE);
  // Start With The fifth List

glBegin(GL_QUADS);
glNormal3f(-1.0f, 1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0042475f, -0.003003f);
//V1
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V2
glVertex3f(-0.0042475f, +0.0000000f, +0.003003f);
//V4
glVertex3f(-0.0042475f, -0.0000000f, -0.003003f);
//V3
glNormal3f(-1.0f, -1.0f, 0.0f);
glVertex3f(-0.0000000f, +0.0042475f, -0.003003f);
//V3
glVertex3f(-0.0042475f, +0.0000000f, +0.003003f);
//V4
glVertex3f(+0.0000000f, -0.0042475f, +0.003003f);
//V6
glVertex3f(+0.0042475f, -0.0000000f, -0.003003f);
//V5
glNormal3f(1.0f, -1.0f, 0.0f);
glVertex3f(+0.0000000f, -0.0042475f, +0.003003f);
//V5
glVertex3f(+0.0042475f, +0.0000000f, +0.003003f);
//V6
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V8
glVertex3f(+0.0042475f, -0.0000000f, -0.003003f);
//V7
glNormal3f(1.0f, 1.0f, 0.0f);
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V8
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V10
glVertex3f(-0.0000000f, +0.0042475f, -0.003003f);
//V9
glNormal3f(0.0f, 0.0f, 1.0f);
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V10
glVertex3f(+0.0000000f, +0.0042475f, +0.003003f);
//V8
glVertex3f(-0.0042475f, +0.0000000f, +0.003003f);
//V6
glVertex3f(-0.0042475f, +0.0000000f, +0.003003f);
//V4
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(+0.0000000f, +0.0042475f, -0.003003f);
//V10
glVertex3f(+0.0042475f, +0.0000000f, -0.003003f);
//V8
glVertex3f(+0.0000000f, -0.0042475f, -0.003003f);
//V6
glVertex3f(-0.0042475f, +0.0000000f, -0.003003f);
//V4
glEnd();
gEndList();

glNewList(5, GL_COMPILE);
// Compile the LTA Cage

glBegin(GL_QUADS);
Appendices – Appendix 5

```c
//V16
//V1
//V16
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
//V17
```

```c
//V21
//V21
//V21
//V21
```
Appendices – Appendix 5

```
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f,
+0.0042475f+0.002995f);  //V6
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f,
+0.0084955f+0.002995f);  //V22
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f,
+0.0084955f+0.002995f);  //V22
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f,
+0.0042475f+0.002995f);  //V6
glEnd();

glBegin(GL_QUADS);
glNormal3f(1.0f, 0.0f, -1.0f);
glVertex3f(+0.0084955f+0.002995f, -0.0000000f+0.002995f, -
0.0084955f-0.002995f);  //V18
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f, -
0.0042475f-0.002995f);  //V8
glVertex3f(+0.0084955f+0.002995f, +0.0000000f-0.002995f, -
0.0042475f-0.002995f);  //V8
glVertex3f(+0.0042475f+0.002995f, -0.0000000f-0.002995f, -
0.0084955f-0.002995f);  //V18
glEnd();

glBegin(GL_QUADS);
glNormal3f(-1.0f, 0.0f, -1.0f);
glVertex3f(-0.0084955f-0.002995f, +0.0000000f+0.002995f, -
0.0042475f-0.002995f);  //V4
glVertex3f(-0.0084955f-0.002995f, -0.0000000f+0.002995f, -
0.0042475f-0.002995f);  //V20
glVertex3f(-0.0084955f-0.002995f, -0.0000000f-0.002995f, -
0.0042475f-0.002995f);  //V20
glVertex3f(-0.0042475f-0.002995f, -0.0000000f-0.002995f, -
0.0084955f-0.002995f);  //V4
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, -1.0f, 1.0f);
glVertex3f(-0.0000000f-0.002995f, -0.0084955f-0.002995f,
+0.0042475f+0.002995f);  //V11
glVertex3f(+0.0000000f-0.002995f, -0.0042475f-0.002995f,
+0.0084955f+0.002995f);  //V23
glVertex3f(+0.0000000f+0.002995f, -0.0042475f+0.002995f,
+0.0084955f+0.002995f);  //V23
glVertex3f(-0.0000000f+0.002995f, -0.0084955f-0.002995f,
+0.0042475f+0.002995f);  //V11
glEnd();

glBegin(GL_QUADS);
glNormal3f(1.0f, -1.0f, 0.0f);
glVertex3f(+0.0000000f-0.002995f, -0.0084955f-0.002995f, -
0.0000000f+0.002995f);  //V10
glVertex3f(+0.0084955f+0.002995f, -0.0042475f-0.002995f,
+0.0000000f+0.002995f);  //V7
glVertex3f(+0.0084955f+0.002995f, -0.0042475f-0.002995f,
+0.0000000f-0.002995f);  //V7
glVertex3f(+0.0042475f+0.002995f, -0.0084955f-0.002995f,
-0.0000000f-0.002995f);  //V10
glEnd();

glBegin(GL_QUADS);
glNormal3f(0.0f, -1.0f, -1.0f);
```
Appendices – Appendix 5

```c
glVertex3f(+0.0000000f+0.002995f, -0.0042475f-0.002995f, -
0.0084955f-0.002995f);  //V19
glVertex3f(-0.0000000f+0.002995f, -0.0084955f-0.002995f, -
0.0042475f-0.002995f);  //V9
glVertex3f(-0.0000000f-0.002995f, -0.0084955f-0.002995f, -
0.0042475f-0.002995f);  //V9
glVertex3f(+0.0000000f-0.002995f, -0.0042475f-0.002995f, -
0.0084955f-0.002995f);  //V19
glEnd();

glBegin(GL_QUADS);
glNormal3f(-1.0f, -1.0f, 0.0f);
glVertex3f(-0.0084955f-0.002995f, -0.0042475f-0.002995f,
+0.0000000f-0.002995f);  //V3
glVertex3f(-0.0042475f-0.002995f, -0.0084955f-0.002995f,
+0.0000000f-0.002995f);  //V3
glVertex3f(-0.0042475f-0.002995f, -0.0084955f-0.002995f,
+0.0000000f+0.002995f);  //V12
glVertex3f(-0.0084955f-0.002995f, -0.0042475f-0.002995f,
+0.0000000f+0.002995f);  //V12
glEnd();

glBegin(GL_POLYGON);
glNormal3f(-0.58f, 0.58f, 0.58f);
glVertex3f(-0.0000000f-0.002995f, +0.0084955f+0.002995f,
+0.0042475f+0.002995f);  //V15
glVertex3f(-0.0042475f-0.002995f, +0.0084955f+0.002995f,
+0.0000000f+0.002995f);  //V16
glVertex3f(-0.0084955f-0.002995f, +0.0042475f+0.002995f,
-0.0000000f+0.002995f);  //V1
glVertex3f(-0.0084955f-0.002995f, +0.0000000f+0.002995f,
+0.0042475f+0.002995f);  //V2
glVertex3f(-0.0042475f-0.002995f, -0.0000000f+0.002995f,
+0.0084955f+0.002995f);  //V24
glVertex3f(-0.0000000f-0.002995f, +0.0042475f+0.002995f,
+0.0084955f+0.002995f);  //V21
glEnd();

glBegin(GL_POLYGON);
glNormal3f(0.58f, 0.58f, 0.58f);
glVertex3f(-0.0000000f+0.002995f, +0.0084955f+0.002995f,
+0.0042475f+0.002995f);  //V15
glVertex3f(+0.0042475f+0.002995f, +0.0084955f+0.002995f,
-0.0000000f+0.002995f);  //V14
glVertex3f(+0.0084955f+0.002995f, +0.0042475f+0.002995f,
-0.0000000f+0.002995f);  //V5
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f,
+0.0042475f+0.002995f);  //V6
glVertex3f(+0.0042475f+0.002995f, -0.0000000f+0.002995f,
+0.0084955f+0.002995f);  //V22
glVertex3f(-0.0000000f+0.002995f, +0.0042475f+0.002995f,
+0.0084955f+0.002995f);  //V21
glEnd();

glBegin(GL_POLYGON);
glNormal3f(0.58f, 0.58f, -0.58f);
glVertex3f(+0.0042475f+0.002995f, +0.0084955f+0.002995f,
-0.0000000f-0.002995f);  //V14
glVertex3f(-0.0042475f+0.002995f, +0.0084955f+0.002995f,
-0.0000000f-0.002995f);  //V13
```
glVertex3f(-0.000000f+0.002995f, +0.0042475f+0.002995f, -0.0084955f+0.002995f); //V1
glVertex3f(+0.0042475f+0.002995f, -0.0000000f+0.002995f, -0.0084955f+0.002995f); //V17
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f, -0.0084955f+0.002995f); //V18
glVertex3f(+0.0084955f+0.002995f, +0.0000000f+0.002995f, -0.0084955f+0.002995f); //V8
glVertex3f(+0.0084955f+0.002995f, +0.0042475f+0.002995f, -0.0000000f+0.002995f); //V5
glEnd();

glBegin(GL_POLYGON);
glNormal3f(-0.58f, 0.58f, -0.58f);
glVertex3f(-0.0000000f-0.002995f, +0.0084955f+0.002995f, -0.0084955f+0.002995f); //V13
glVertex3f(-0.0084955f-0.002995f, +0.0084955f+0.002995f, -0.0084955f+0.002995f); //V16
glVertex3f(-0.0084955f-0.002995f, +0.0000000f+0.002995f, -0.0084955f+0.002995f); //V1

glBegin(GL_POLYGON);
glNormal3f(-0.58f, -0.58f, 0.58f);
glVertex3f(+0.0000000f-0.002995f, -0.0042475f-0.002995f, -0.0084955f+0.002995f); //V23
glVertex3f(+0.0042475f-0.002995f, -0.0000000f-0.002995f, -0.0084955f+0.002995f); //V24
glVertex3f(+0.0084955f-0.002995f, +0.0000000f-0.002995f, -0.0084955f+0.002995f); //V2

glBegin(GL_POLYGON);
glNormal3f(0.58f, -0.58f, 0.58f);
glVertex3f(+0.0042475f+0.002995f, -0.0000000f-0.002995f, -0.0084955f+0.002995f); //V22
glVertex3f(+0.0084955f+0.002995f, -0.0042475f-0.002995f, -0.0084955f+0.002995f); //V23
glVertex3f(+0.0084955f+0.002995f, -0.0084955f-0.002995f, -0.0084955f+0.002995f); //V11

glBegin(GL_POLYGON);
glNormal3f(-0.58f, -0.58f, -0.58f);
goToDentifyFromm(0.0042475f+0.002995f, -0.0000000f-0.002995f, -0.0084955f+0.002995f); //V10
glVertex3f(+0.0084955f+0.002995f, -0.0042475f-0.002995f, -0.0084955f+0.002995f); //V7

Appendices – Appendix 5

Appendix 5

357
Appendices – Appendix 5

```cpp
void glDrawElements(GLenum mode, GLsizei count, GLenum type, const void* indices)
{
    // Draw elements
}
```
Appendices – Appendix 5

```c
glVertex3f(-0.0084955f, -0.002995f, +0.0000000f, -0.002995f, -
0.0042475f, -0.002995f);  //V8
glVertex3f(-0.0084955f, -0.002995f, -0.0042475f, -0.002995f, -
+0.0000000f, -0.002995f);  //V7
glVertex3f(-0.0084955f, -0.002995f, -0.0042475f, -0.002995f, +
+0.0000000f, +0.002995f);  //V7
glVertex3f(-0.0084955f, -0.002995f, +0.0000000f, -0.002995f, +
+0.0042475f, +0.002995f);  //V6
glEnd();

glBegin(GL_POLYGON);
  glNormal3f(0.0f, 1.0f, 0.0f);
  glVertex3f(-0.0000000f, +0.002995f, +0.0084955f, +0.002995f, -
0.0042475f, -0.002995f);  //V13
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, -0.002995f);  //V14
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, +0.002995f);  //V14
  glVertex3f(-0.0000000f, +0.002995f, +0.0084955f, +0.002995f, +
+0.0042475f, +0.002995f);  //V15
  glVertex3f(-0.0000000f, -0.002995f, +0.0084955f, +0.002995f, +
+0.0042475f, +0.002995f);  //V15
  glVertex3f(-0.0000000f, -0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, -0.002995f);  //V16
  glVertex3f(-0.0000000f, -0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, +0.002995f);  //V16
  glVertex3f(-0.0000000f, -0.002995f, +0.0084955f, +0.002995f, -
0.0042475f, -0.002995f);  //V13
glEnd();

glBegin(GL_POLYGON);
  glNormal3f(0.0f, -1.0f, 0.0f);
  glVertex3f(-0.0000000f, +0.002995f, -0.0084955f, -0.002995f, -
0.0042475f, -0.002995f);  //V13
  glVertex3f(+0.0042475f, +0.002995f, -0.0084955f, -0.002995f, -
+0.0000000f, -0.002995f);  //V14
  glVertex3f(+0.0042475f, +0.002995f, -0.0084955f, -0.002995f, -
+0.0000000f, +0.002995f);  //V14
  glVertex3f(-0.0000000f, +0.002995f, -0.0084955f, -0.002995f, +
+0.0042475f, +0.002995f);  //V15
  glVertex3f(-0.0000000f, -0.002995f, -0.0084955f, -0.002995f, +
+0.0042475f, +0.002995f);  //V15
  glVertex3f(-0.0000000f, -0.002995f, -0.0084955f, -0.002995f, -
+0.0000000f, -0.002995f);  //V16
  glVertex3f(-0.0000000f, -0.002995f, -0.0084955f, -0.002995f, -
+0.0000000f, +0.002995f);  //V16
  glVertex3f(-0.0000000f, -0.002995f, -0.0084955f, -0.002995f, -
0.0042475f, -0.002995f);  //V13
glEnd();

glBegin(GL_POLYGON);
  glNormal3f(0.0f, 0.0f, 1.0f);
  glVertex3f(-0.0000000f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0042475f, +0.002995f);  //V21
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, -0.002995f);  //V22
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0000000f, +0.002995f);  //V22
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
+0.0084955f, +0.002995f);  //V23
  glVertex3f(+0.0042475f, +0.002995f, +0.0084955f, +0.002995f, -
-0.0084955f, +0.002995f);  //V23
```
Appendices – Appendix 5

```c
glVertex3f(+0.0000000f-0.002995f, -0.0042475f-0.002995f, +0.0084955f+0.002995f);
    //V23
glVertex3f(-0.0042475f-0.002995f, -0.0000000f-0.002995f, +0.0084955f+0.002995f);
    //V24
glVertex3f(-0.0042475f-0.002995f, -0.0000000f+0.002995f, +0.0084955f+0.002995f);
    //V24
glVertex3f(-0.0000000f-0.002995f, +0.0042475f+0.002995f, +0.0084955f+0.002995f);
    //V21
glEnd();

glBegin(GL_POLYGON);
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f(-0.0000000f+0.002995f, +0.0042475f+0.002995f, -0.0084955f-0.002995f);
    //V21
glVertex3f(+0.0042475f+0.002995f, -0.0000000f+0.002995f, -0.0084955f-0.002995f);
    //V22
glVertex3f(+0.0042475f+0.002995f, -0.0000000f-0.002995f, -0.0084955f-0.002995f);
    //V22
glVertex3f(+0.0000000f+0.002995f, -0.0042475f-0.002995f, -0.0084955f-0.002995f);
    //V23
glVertex3f(+0.0000000f-0.002995f, -0.0042475f-0.002995f, -0.0084955f-0.002995f);
    //V23
glVertex3f(-0.0042475f-0.002995f, -0.0000000f-0.002995f, -0.0084955f-0.002995f);
    //V24
glVertex3f(-0.0042475f-0.002995f, -0.0000000f+0.002995f, -0.0084955f-0.002995f);
    //V24
glVertex3f(-0.0000000f-0.002995f, +0.0042475f+0.002995f, -0.0084955f-0.002995f);
    //V21
glEnd();
glEndList();
```

FileOps.h

```c
#include <stdio.h>
#include "Colouring.h"
#include "DisplayLists.h"

#ifndef FileOps_H
#define FileOps_H

typedef struct
{
    short int x, y, z;
    short int c;
} VERTEX;

typedef struct
{
    VERTEX *points;
} OBJECT;

typedef struct
{
    int StartFrame, EndFrame,
    int OffsetX, OffsetY, OffsetZ,
    int GreatestX, GreatestY, GreatestZ,
    int LeastX, LeastY, LeastZ,
    int face100, face110, face111,
    float adjustment100, adjustment110, adjustment111;
    float ZoomInterval;
}
```

360
typedef struct {
    bool dataloaded;
    int framenum, currentframe, ColourMode, lines;
    FRAMECOUNTERS *frame;
} FRAMEOBJECT;

void objload(OBJECT *k, FRAMEOBJECT *j);
void FileOpen(HWND hwnd);
void FileSave(HWND hwnd);
void saveallframes();
void freerectangles(OBJECT *k);
void freeframes(FRAMEOBJECT *j);
void processfiles(HWND hwnd);

#endif

FileOps.cpp

#include "FileOps.h"
OBJECT vertices;
FRAMEOBJECT frames;

char szFileName[MAX_PATH] = "";
char szFilePath[MAX_PATH] = "";
char szFilePathName[MAX_PATH] = "";
extern int wwidth;
extern int wheight;
extern GLuint list;
extern GLsizei range;
extern bool redraw;

void allocatevertices(OBJECT *k, int n)
{
    k->points = (VERTEX*)malloc(sizeof(VERTEX)*n);
}

void allocateframes(FRAMEOBJECT *j, int n)
{
    j->frame = (FRAMECOUNTERS*)malloc(sizeof(FRAMECOUNTERS)*n);
}

void reallocatevertices(OBJECT *k, int n)
{
    k->points = (VERTEX*)realloc(k->points, sizeof(VERTEX)*n);
}

void reallocateframes(FRAMEOBJECT *j, int n)
{
    j->frame = (FRAMECOUNTERS*)realloc(j->frame, sizeof(FRAMECOUNTERS)*n);
}
void freevertices(OBJECT *k)  
{  
  free(k->points);  
}  

void freeframes(FRAMEOBJECT *j)  
{  
  free(j->frame);  
}  

void readstr(FILE *f, char *string)  
{  
  do  
  {  
    fgets(string, 255, f);  
  }  
  while ((string[0] == '/') || (string[0] == '\n'));  
  return;  
}  

void removeextension()  
{  
  for(unsigned int i = 0; i < (strlen(szFileName) - 4); i++) {  
    szFilePathName[i] = szFileName[i];  
  }  
  szFilePathName[strlen(szFileName) - 4] = '\0';  
}  

void extractpath()  
{  
  for(unsigned int i = 0; i < (strlen(szFileName) - 4); i++) {  
    szFilePath[i] = szFileName[i];  
  }  
  for(int i = (strlen(szFilePath)-1); i > 0; i--) {  
    if(szFilePath[i] == '\\') {  
      szFilePath[i+1] = '\0';  
      break;  
    }  
  }  
}  

void objload(OBJECT *k, FRAMEOBJECT *j)  
{  
  freevertices(&vertices);  
  freeframes(&frames);  
  glDeleteLists(list, range);  
  int a=0, b=0, c=0;  
  int rx, ry, rz;  
  FILE *filein;  
  char oneline[255], ch='\0';  
  int grtx=0, gtry=0, gtrz=0, lstx=10000, lsty=10000, lstz=10000,  
  offsetx=0, offsety=0, offsetz=0, firstframe=0,  
  lastframe=0,  
  tx, ty, tz, cx, cy, cz;
GLfloat zoominterval = 0.0f;
frames.lines = 0;
frames.ColourMode = 1;
frames.currentframe = 0;
frames.dataloaded = true;
allocatevertices(k, i400000000);
// allocatevertices(k, 1000000);
allocateframes(j, 1);
fopen_s(& filein, szFileName, "rt");
j -> frame[c].StartFrame = a;
while((ch = getc(filein)) != EOF) {
    readstr(filein, oneline);
    sscanf_s(oneline, "%i %i %i", &rx, &ry, &rz);
    if(rx != 0 && ry != 0) {
        if(rx > grtx) grtx = rx;
        if(rx < lstx) lstx = rx;
        if(ry > grty) grty = ry;
        if(ry < lsty) lsty = ry;
        if(rz > grtz) grtz = rz;
        if(rz < lstz) lstz = rz;

        k->points[a].x = rx;
        k->points[a].y = ry;
        k->points[a].z = rz;

        // if(b == 999999) {
        //    reallocatevertices(k, a+1000000);
        //    b = 0;
        // }

        a++;
        b++;
    } else if(rx == 0 && ry == 0) {
        j -> frame[c].ZoomInterval = float(grtx)/5000.0f;
        j -> frame[c].OffsetX = int(float(grtx+lstx)*0.5);
        j -> frame[c].OffsetY = int(float(grty+lsty)*0.5);
        j -> frame[c].OffsetZ = int(float(grtz+lstz)*0.5);
        j -> frame[c].GreatestX = grtx;
        j -> frame[c].GreatestY = grty;
        j -> frame[c].GreatestZ = grtz;
        j -> frame[c].LeastX = lstx;
        j -> frame[c].LeastY = lsty;
        j -> frame[c].LeastZ = lstz;
        j -> frame[c].EndFrame = a;
        j -> frame[c].adjustment100 = 0;
        j -> frame[c].adjustment110 = 0;
        j -> frame[c].adjustment111 = 0;
        j -> frame[c].face100 = 0;
        j -> frame[c].face110 = 0;
        j -> frame[c].face111 = 0;
    }
}

363
c++;
reallocateframes(j, c+1);

j -> frame[c].StartFrame = a;
grtx=0, grty=0, grtz=0,
lstx=10000, lsty=10000, lstz=10000,
zoominterval=0.0f;
}
}
reallocateframes(j, c);
j -> framenum = c;
// reallocatevertices(k, a);

for(int i=0;i<frames.framenum;i++)
{
  firstframe = frames.frame[i].StartFrame;
  lastframe = frames.frame[i].EndFrame;

  for(int j=firstframe;j<lastframe;j++)
  {
    tx=vertices.points[j].x;
    ty=vertices.points[j].y;
    tz=vertices.points[j].z;

    cx = abs(tx - /*501*/ frames.frame[i].OffsetX);
    cy = abs(ty - /*501*/ frames.frame[i].OffsetY);
    cz = abs(tz - /*501*/ frames.frame[i].OffsetZ);

    if((cx==cy) && (cx==cz)) { frames.frame[i].face111 = 3*cx; }
    if((cx==cy) && (cz==0)) { frames.frame[i].face110 = 2*cx; }
    if((cy==0) && (cz==0)) { frames.frame[i].face100 = cx; }
  }
  //-----------------------------------------------------
  // Coloring crystals is a pain, this loop here is added
to compensate for
  // what happens when the <110> vector intersects the
  // surface at a layer
  // that cannot be accounted for by x==y && z==0, a
  // better option could
  // utilise the same loop above or at least run through
  // these two loops
  // once only.
  //-----------------------------------------------------

  if(frames.frame[i].face110<1){

for(int j=firstframe;j<lastframe;j++)
{
    tx=vertices.points[j].x;
    ty=vertices.points[j].y;
    tz=vertices.points[j].z;

cx = abs(tx - frames.frame[i].OffsetX);
cy = abs(ty - frames.frame[i].OffsetY);
cz = abs(tz - frames.frame[i].OffsetZ);

if((cx-1==cy) && (cz==0))
{
    frames.frame[i].face110 = 2*cx;
}

if(frames.frame[i].face111<3){
    for(int j=firstframe;j<lastframe;j++)
{
    tx=vertices.points[j].x;
    ty=vertices.points[j].y;
    tz=vertices.points[j].z;

    cx = abs(tx - frames.frame[i].OffsetX);
cy = abs(ty - frames.frame[i].OffsetY);
cz = abs(tz - frames.frame[i].OffsetZ);

    if((cx-1==cy) && (cz==0))
    {
        frames.frame[i].face111 = 2*cx;
    }
    if((cx==cy) && (cx-1==cz))
    {
        frames.frame[i].face111 = 3*cx;
    }
    if((cx-1==cy) && (cx-1==cz))
    {
        frames.frame[i].face111 = 3*cx;
    }
    if((cx-1==cy) && (cx-1==cz))
    {
        frames.frame[i].face111 = 3*cx;
    }
    }
}
}
fclose(filein);
// BuildLists1ta();
// BuildListssod();
BuildLists1tl();
colouring(frames.ColourMode,0,c);
redraw = true;
}

void FileOpen(HWND hwnd)
{
    OPENFILENAME ofn;
    ZeroMemory(&ofn, sizeof(ofn));
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwnd;
ofn.lpstrFilter = "Simulations 3D (*.xyz)\0*.xyz\0Simulations 2D (*.xy)\0*.xy\0";
ofn.lpstrFile = szFileName;
ofn.nMaxFile = MAX_PATH;
Appendices – Appendix 5

```c
ofn.Flags = OFN_EXPLORER | OFN_FILEMUSTEXIST |
OFN_HIDEREADONLY;
ofn.lpstrDefExt = NULL;
GetOpenFileName(&ofn);
}

void objsave(int windowWidth, int windowHeight, char* filename)
{
    FILE *fileptr;
    byte* bmpBuffer = (byte*)malloc(windowWidth*windowHeight*3);
    byte tmp;
    if (!bmpBuffer)
        return;
    DrawScene();
    glReadPixels(0, 0, windowWidth-1, windowHeight-1, GL_RGB,
GL_UNSIGNED_BYTE, bmpBuffer);
    fopen_s(&fileptr, filename, "wb");
    BITMAPFILEHEADER bitmapFileHeader;
    BITMAPINFOHEADER bitmapInfoHeader;
    bitmapFileHeader.bfType = 0x4D42; //"BM"
    bitmapFileHeader.bfSize = windowWidth*windowHeight*3;
    bitmapFileHeader.bfReserved1 = 0;
    bitmapFileHeader.bfReserved2 = 0;
    bitmapFileHeader.bfOffBits = sizeof(BITMAPFILEHEADER) +
sizeof(BITMAPINFOHEADER);
    bitmapInfoHeader.biSize = sizeof(BITMAPINFOHEADER);
    bitmapInfoHeader.biWidth = windowWidth-1;
    bitmapInfoHeader.biHeight = windowHeight-1;
    bitmapInfoHeader.biPlanes = 1;
    bitmapInfoHeader.biBitCount = 24;
    bitmapInfoHeader.biCompression = BI_RGB;
    bitmapInfoHeader.biSizeImage = 0;
    bitmapInfoHeader.biXPelsPerMeter = 0;
    bitmapInfoHeader.biYPelsPerMeter = 0;
    bitmapInfoHeader.biClrUsed = 0;
    bitmapInfoHeader.biClrImportant = 0;
    for (int j = 2; j < windowWidth*windowHeight*3; j+=3)
    {
        tmp = bmpBuffer[j-2];
        bmpBuffer[j-2] = bmpBuffer[j];
        bmpBuffer[j] = tmp;
    }
    fwrite(&bitmapFileHeader, sizeof(BITMAPFILEHEADER), 1,
fileptr);
    fwrite(&bitmapInfoHeader, sizeof(BITMAPINFOHEADER), 1,
fileptr);
    fwrite(bmpBuffer, windowWidth*windowHeight*3, 1, fileptr);
    fclose(fileptr);
    free(bmpBuffer);
}

void FileSave(HWND hwnd)
```

366
{  
    OPENFILENAME ofn;
    char svFileName[MAX_PATH] = "";
    ZeroMemory(&ofn, sizeof(ofn));
    ofn.lStructSize = sizeof(OPENFILENAME);
    ofn hwndOwner = hwnd;
    ofn.lpstrFilter = "Bitmap (*.bmp)\0*.bmp\0";
    ofn.lpstrFile = svFileName;
    ofn.nMaxFile = MAX_PATH;
    ofn.Flags = OFN_EXPLORER | OFN_FILEMUSTEXIST |
    OFN_HIDEREADONLY;
    ofn.lpstrDefExt = ".bmp";
    if(GetSaveFileName(&ofn))
    {
        objsave(wwidth,wheight,svFileName);
    }
}

void saveallframes()
{
    char temp[MAX_PATH];
    if(frames.dataloaded) {
        removeextension();
        for(int i = 0;i < frames.framenum;i++)
        {
            frames.currentframe = i;
            sprintf_s(temp,MAX_PATH,"%s-frame%d.bmp\0",szFilePathName,i+1);
            objsave(wwidth,wheight,temp);
        }
    } else {
        MessageBox(NULL,"Load up some data first","Oh come on!",MB_OK|MB_ICONEXCLAMATION);
    }
}

void processfiles(HWND hWnd)
{
    WIN32_FIND_DATA FindFileData;
    HANDLE hFind = INVALID_HANDLE_VALUE;
    char dirpathscan[MAX_PATH];
    FileOpen(hWnd);
    extractpath();
    sprintf_s(dirpathscan,MAX_PATH,%"\s*.xyz","szFilePath);  
    hFind = FindFirstFile(dirpathscan, &FindFileData);
    if(hFind != INVALID_HANDLE_VALUE) {
        do {
            if((FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 1) {
                sprintf_s(szFileName,MAX_PATH,%"\s%s","szFilePath,FindFileData.cFileName);
                objload(&vertices,&frames);
            }
        }while(FindNextFile(hFind,&FindFileData));
Main.cpp

#include "main.h"

HDC hDC=\null;
HGLRC hRC=\null;
HWND hWnd=\null;
HINSTANCE hInstance;

bool keys[256];
bool redraw = true;

bool RMouseDrag = 0, MouseDrag = 0;
int MouseLastX, MouseLastY, MouseLastZ;
int MouseDeltaX, MouseDeltaY, MouseDeltaZ;
int stride = 1;
float RotationX = 30.0f, RotationY = -45.0f, RotationZ = 0.0f;
GLfloat transx = 0.0f, transy = 0.0f, transz = -14.0f;

extern FRAMEOBJECT frames;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg;
    BOOL done = FALSE;

    int desktopw = GetSystemMetrics(SM_CXFULLSCREEN) - 15;
    int desktoph = GetSystemMetrics(SM_CYFULLSCREEN) - 15;

    // Create Our OpenGL Window
    if (!CreateGLWindow("Crystal Growth Visualiser", desktopw, desktoph, 16))
    {
        return 0;
    }

    while(!done)
    {
        if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
        {
            if (msg.message==WM_QUIT)
            {
                done=TRUE;
            } else {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        } else {
if (keys[VK_ESCAPE])
{
    done=TRUE;
} else {
    if(redraw){
        DrawScene();
        SwapBuffers(hDC);
        redraw = false;
    }
    if (keys[VK_CAPITAL])
    { redraw = true; }
    if (keys['A'])
    { transx -= frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['D'])
    { transx += frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['W'])
    { transy += frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['S'])
    { transy -= frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['Z'])
    { transz += frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['X'])
    { transz -= frames.frame[frames.currentframe].ZoomInterval; redraw = true; }
    if (keys['K'])
    {
        frames.frame[frames.currentframe].adjustment111++;
        colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
        redraw = true;
    }
    if (keys['M'])
    {
        frames.frame[frames.currentframe].adjustment111--;
        colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
        redraw = true;
    }
    if (keys['J'])
    {
        frames.frame[frames.currentframe].adjustment110++;
        colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
        redraw = true;
    }
    if (keys['N'])
    {
        frames.frame[frames.currentframe].adjustment110--;
    }
Appendices – Appendix 5

```c

    colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
    redraw = true;
    }
    if (keys['H'])
    {
        frames.frame[frames.currentframe].adjustment100++;
    }
    colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
    redraw = true;
    }
    if (keys['B'])
    {
        frames.frame[frames.currentframe].adjustment100--;
    }
    colouring(frames.ColourMode,frames.currentframe,frames.currentframe+1);
    redraw = true;
    }
    if (keys['1'])
    { stride = 1; redraw = true; }
    if (keys['2'])
    { stride = 2; redraw = true; }
    if (keys['3'])
    { stride = 3; redraw = true; }
    if (keys['4'])
    { stride = 4; redraw = true; }
    if (keys[VK_LEFT])
    { if(frames.currentframe > 0) { frames.currentframe--; redraw =
        true; }
    }
    if (keys[VK_RIGHT])
    { if(frames.currentframe <
        (frames.framenum)-1) { frames.currentframe++; redraw =
        true; }
    }
    if(MouseDrag)
    { 
        RotationX += (float)MouseDeltaY * 0.5f;
        RotationY += (float)MouseDeltaX * 0.5f;
        redraw = true;
    }
    if(RMouseDrag)
    { 
        RotationZ += (float)MouseDeltaZ * 0.5f;
        redraw = true;
    }
    }
    WaitMessage();
```

370
Appendices – Appendix 5

}  
// Shutdown
KillGLWindow();
return (msg.wParam);
}

Messages.h

#include "Resource.h"
#include "FileOps.h"

#ifndef Messages_H
#define Messages_H

INT_PTR CALLBACK About(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam);
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

#endif

Messages.cpp

#include "Messages.h"

HINSTANCE hInst;

extern OBJECT vertices;
extern FRAMEOBJECT frames;

extern bool keys[256];
extern bool redraw;
extern bool MouseDrag, RMouseDrag;
extern int MouseLastX, MouseLastY, MouseLastZ;
extern int MouseDeltaX, MouseDeltaY, MouseDeltaZ;

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (uMsg)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;

    switch (uMsg)
    {
    case WM_COMMAND:
    {
        wmId = LOWORD(wParam);
        wmEvent = HIWORD(wParam);
        // Parse the menu selections:
        switch (wmId)
        {
        case IDM_OPEN:
            FileOpen(hWnd);
            objload(&vertices,&frames);
            break;
        case IDM_SAVE:
            FileSave(hWnd);
            break;
        case IDM_SAVEALL:
            saveallframes();
            break;
        case IDM_PROCESSALL:
            processfiles(hWnd);
            break;
        case IDM_EXIT:
            PostQuitMessage(0);
            break;
        case IDM_LAYERS_M1:
            frames.ColourMode = 1;
            colouring(1,0,frames.framenum);
            redraw = true;
            break;
        case IDM_LAYERS_M2:
            frames.ColourMode = 2;
            colouring(2,0,frames.framenum);
            redraw = true;
            break;
        case IDM_CAGES:
            frames.ColourMode = 3;
            redraw = true;
            break;
        case IDM_LINES:
            if((frames.lines == 0){ frames.lines = 1; } 
            else if((frames.lines == 1){ frames.lines = 
            0; })
                redraw = true;
                break;
        case IDM_ABOUT:
            DialogBox(hInst, 
            MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
            break;
        }
        break;
    }
    
    case WM_SYSCOMMAND:
    {
    }
}
switch (wParam)
{
    case SC_SCREENSSAVE:
    case SC_MONITORPOWER:
        return 0;
}
break;
}
case WM_CLOSE:
{
    PostQuitMessage(0);
    return 0;
}
case WM_KEYDOWN:
{
    keys[wParam] = TRUE;
    return 0;
}
case WM_KEYUP:
{
    keys[wParam] = FALSE;
    return 0;
}
case WM_SIZE:
{
    ResizeGLScene(LOWORD(lParam), HIWORD(lParam));
    return 0;
}
case WM_LBUTTONDOWN:
    MouseDrag = 1;
    MouseLastX = LOWORD(lParam);
    MouseLastY = HIWORD(lParam);
    SetCapture(hWnd);
    break;

case WM_RBUTTONDOWN:
    RMouseDrag = 1;
    MouseLastZ = LOWORD(lParam);
    SetCapture(hWnd);
    break;

case WM_MOUSEMOVE:
    if (MouseDrag)
    {
        MouseDeltaX = LOWORD(lParam) - MouseLastX;
        MouseDeltaY = HIWORD(lParam) - MouseLastY;
        MouseLastX = LOWORD(lParam);
        MouseLastY = HIWORD(lParam);
    } else if (RMouseDrag) {
        MouseDeltaZ = HIWORD(lParam) - MouseLastZ;
        MouseLastZ = HIWORD(lParam);
    }
    break;

case WM_LBUTTONUP:
    MouseDrag = 0;
Appendices – Appendix 5

    ReleaseCapture();
    break;
    case WM_RBUTTONDOWN:
        RMouseDrag = 0;
        ReleaseCapture();
        break;
    }

    // Pass All Unhandled Messages To DefWindowProc
    return DefWindowProc(hWnd, uMsg, wParam, lParam);
}

Opengl.h

#include <windows.h>
#include <gl\gl.h>
#include <gl\glu.h>
#include <math.h>
#pragma once
#include "FileOps.h"

#ifndef OPENGL_H
#define OPENGL_H

void ReSizeGLScene(GLsizei width, GLsizei height);
void KillGLWindow();
void InitGL();
void DrawScene();
#endif

Opengl.cpp

#include "Opengl.h"

extern HDC hDC;
extern HGLRC hRC;
extern HWND hWnd;
extern HINSTANCE hInstance;
extern OBJECT vertices;
extern FRAMEOBJECT frames;
extern GLfloat transx, transy, transz, RotationX, RotationY, RotationZ;
extern bool keys[256];
extern bool redraw, dataloaded, MouseDrag, RMouseDrag;
extern int stride;

int wwidth, wheight;
GLuint listid;

void BuildFont()
{
    HFONT font;
    HFONT prevfont;

    listid = glGenLists(96);

    font = CreateFont(-12, 0, 0, 0, FW_BOLD, FALSE, FALSE, FALSE,
        ANSI_CHARSET, OUT_TT_PRECIS, CLIP_DEFAULT_PRECIS, ANTIALIASED_QUALITY,
        FF_DONTCARE, DEFAULT_PITCH, "Courier New");
prevfont = (HFONT)SelectObject(hDC, font);
wglUseFontBitmaps(hDC, 32, 96, listid);
SelectObject(hDC, prevfont);
DeleteObject(font);
}

void KillFont()
{
    glDeleteLists(listid, 96);
}

void glPrint(const char *textstring, ...)
{
    char TextString[256];
    va_list ArgList;

    if (textstring == NULL)
        return;

    va_start(ArgList, textstring);
    vsprintf_s(TextString, 256, textstring, ArgList);
    va_end(ArgList);
    glPushAttrib(GL_LIST_BIT);
    glListBase(listid - 32);
    glCallLists(strlen(TextString), GL_UNSIGNED_BYTE, TextString);
    glPopAttrib();
}

void PerspectiveMatrix(GLsizei width, GLsizei height)
{
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, (GLfloat)width/(GLfloat)height, 0.1f, 100.0f);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void ReSizeGLScene(GLsizei width, GLsizei height)
{
    if (height == 0)
    {
        height = 1;
    }
    wwidth = width, wheight = height - 20;
    PerspectiveMatrix(width, height);
    redraw = true;
}

void KillGLWindow()
{
    freevertices(&vertices);
}
freeframes(&frames);
KillFont();

if (hRC)
{
    if (!wglMakeCurrent(NULL,NULL))
    {
        MessageBox(NULL,"Release Of DC And RC Failed.","SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
    }

    if (!wglDeleteContext(hRC))
    {
        MessageBox(NULL,"Release Rendering Context Failed.","SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
    }

    hRC=NULL;
}

if (hDC && !ReleaseDC(hWnd,hDC))
{
    MessageBox(NULL,"Release Device Context Failed.","SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
    hDC=NULL;
}

if (hWnd && !DestroyWindow(hWnd))
{
    MessageBox(NULL,"Could Not Release hWnd.","SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
    hWnd=NULL;
}

if (!UnregisterClass("OpenGL",hInstance))
{
    MessageBox(NULL,"Could Not Unregister Class.","SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
    hInstance=NULL;
}

void InitGL()
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
glClearDepth(1.0f);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_COLOR_MATERIAL);
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);

    // glEnable (GL_BLEND);

    // glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    // Light Parameters for light0
    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat diffuseLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat specularLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat position[] = { 0.0f, 1.0f, 1.0f, 0.0f };
// Assign created components to GL_LIGHT0
GLfloat colourcages = 0.0f;
void BuildFont
{
    BuildFont();
}

void SOD3D(int firstframe, int lastframe, int offsetx, int offsety, int offsetz)
{
    int tx, ty, tz, cv;
    GLfloat colourcages[3][4] =
    {{1.0f, 1.0f, 0.0f, 1.0f},
    {0.65f, 0.0f, 0.0f, 1.0f},
    {0.0f, 0.0f, 0.65f, 1.0f},
    {0.0f, 0.0f, 0.0f, 0.18f}};
    BuildFont();
    BuildFont();
    BuildFont();

    if (MouseDrag==false && RMouseDrag == false &&
        keys['A']==false && keys['W']==false &&
        keys['S']==false && keys['X']==false &&
        !GetKeyState(VK_CAPITAL))
    {
        for(int i=firstframe;i<lastframe;i++)
        {
            tx=vertices.points[i].x;
            ty=vertices.points[i].y;
            tz=vertices.points[i].z;
            cv=vertices.points[i].c;

            if(frames.ColourMode <= 2)
                glColor4fv(colourlayers[cv-1]);
            if(frames.ColourMode == 3)
                glColor4fv(colourcages[cv-1][cv-1]);

                glTranslatef((tx-offsetx)*0.008495f,(ty-offsety)*0.008495f,(-tz+offsetz)*0.008495f);
                glCallList(1);
                glTranslatef((-tx+offsetx)*0.008495f,(-ty+offsety)*0.008495f,(-tz+offsetz)*0.008495f);
            }
        } else {
            glBegin(GL_POINTS);
            for(int i=firstframe;i<lastframe;i++)
            {
                tx=vertices.points[i].x;
                ty=vertices.points[i].y;
                tz=vertices.points[i].z;
                cv=vertices.points[i].c;

                if(frames.ColourMode <= 2)
                    glColor4fv(colourlayers[cv-1]);
                if(frames.ColourMode == 3)
                    glColor4fv(colourcages[cv-1][cv-1]);
            }
        }
    }
}
Appendices – Appendix 5

```cpp
void LTL3D(int firstframe, int lastframe, int offsetx, int offsety, int offsetz)
{
    GLfloat colourcages,
    colourlayers;
    GLuint
tx, ty, tz;
    GLfloat
    cv;
    uint cv, maxver = 0;
    GLfloat colourlayers[5][4] = {
        {0.80f, 0.68f, 0.35f, 1.0f},
        {0.70f, 0.48f, 0.10f, 1.0f},
        {0.55f, 0.28f, 0.10f, 1.0f},
        {0.40f, 0.18f, 0.05f, 1.0f},
        {0.40f, 0.18f, 0.05f, 1.0f}
    };
    GLfloat colourcages[5][5][4] = {
        {{1.0f, 0.0f, 0.0f, 1.0f},
         {0.88f, 0.0f, 0.0f, 1.0f},
         {0.76f, 0.0f, 0.0f, 1.0f}},
        {{0.65f, 0.0f, 0.0f, 1.0f},
         {0.40f, 0.18f, 0.05f, 1.0f}},
        {{0.0f, 1.0f, 0.0f, 1.0f},
         {0.0f, 0.88f, 0.0f, 1.0f},
         {0.0f, 0.76f, 0.0f, 1.0f}},
        {{0.0f, 0.0f, 1.0f, 1.0f},
         {0.0f, 0.0f, 0.0f, 1.0f}},
        {{0.0f, 0.0f, 0.65f, 1.0f},
         {0.40f, 0.18f, 0.05f, 1.0f}}
    };

    if(MouseDrag == false && RMouseDrag == false &&
        keys['A'] == false &&
        keys['D'] == false &&
        keys['W'] == false &&
        keys['S'] == false &&
        keys['Z'] == false &&
        keys['X'] == false &&
        !GetKeyState(VK_CAPITAL)) {
        for(int i = firstframe; i < lastframe; i++)
        {
            tx = vertices.points[i].x;
            ty = vertices.points[i].y;
            tz = vertices.points[i].z;
            cv = vertices.points[i].c;

            int j = tx;
            int k = tz;

            if (k % 2 == 0) {
                if (j % 2 == 0) {
                    glVertex3f((tx - offsetx) * 0.008495f,
                                (ty - offsety) * 0.008495f,
                                (tz - offsetz) * 0.008495f);
                    glEnd();
                }
            }
        }
    }

    if (firstframe != lastframe) {
        glVertex3f((tx - offsetx) * 0.00905f,
                    (ty - offsety) * 0.00760f);
        glEnd();
    }

    if (firstframe != lastframe) {
        glVertex3f((tx - offsetx) * 0.00905f,
                    (ty - offsety) * 0.00760f);
        glEnd();
    }

    glVertex3f((tx - offsetx) * 0.00905f,
                (ty - offsety) * 0.00760f);
    glEnd();
}
```

378
void OFF3D(int firstframe, int lastframe, int offsetx, int offsety, int offsetz)
    // here is where the drawing begins
{
    GLfloat tx, ty, tz;
    GLuint cv, maxver=0;

    glBegin(GL_POINTS);
    for(int i=firstframe; i<lastframe; i++)
    {
        tx=vertices.points[i].x;
        ty=vertices.points[i].y;
        tz=vertices.points[i].z;
        cv=vertices.points[i].c;

        if(frames.ColourMode <= 2)
            glColor4fv(colourlayers[cv-1]);
        if(frames.ColourMode == 3)
            glColor4fv(colourcages[2][cv-1]);

        int j = tx;
        int k = tz;
        if (k % 2 == 0){
            if (j % 2 == 0){
                glVertex3f((tx-offsetx)*0.00905f,((tz-offsetz)*(0.01590f)+0.00530f),(ty-offsety)*0.00760f);
            }else{
                glVertex3f((tx-offsetx)*0.00905f,((tz-offsetz)*(0.01590f)),(ty-offsety)*0.00760f);
            }
        }else{
            if (j % 2 == 0){
                glVertex3f((tx-offsetx)*0.00905f,((tz-offsetz)*(0.01590f)-0.00530f),(ty-offsety)*0.00760f);
            }else{
                glVertex3f((tx-offsetx)*0.00905f,((tz-offsetz)*(0.01590f)),(ty-offsety)*0.00760f);
            }
        }

    }
    glEnd();
    glEndList();
}
Appendices – Appendix 5

GLfloat colourlayers[5][4] = 
{{0.80f,0.68f,0.35f,1.0f},{0.70f,0.48f,0.10f,1.0f},{0.55f,0.28f,0.10f,1.0f},{0.40f,0.18f,0.05f,1.0f},{0.40f,0.18f,0.05f,1.0f}};

GLfloat colourcages[5][4] = 
{{(1.0f,0.0f,0.0f,1.0f), (0.88f,0.0f,0.0f,1.0f), (0.76f,0.0f,0.0f,1.0f), (0.40f,0.18f,0.05f,1.0f), (0.65f,0.0f,0.0f,1.0f)},
{(1.0f,0.0f,0.0f,1.0f), (0.88f,0.0f,0.0f,1.0f), (0.76f,0.0f,0.0f,1.0f), (0.40f,0.18f,0.05f,1.0f), (0.65f,0.0f,0.0f,1.0f)},
{(1.0f,0.0f,0.0f,1.0f), (0.88f,0.0f,0.0f,1.0f), (0.76f,0.0f,0.0f,1.0f), (0.40f,0.18f,0.05f,1.0f), (0.65f,0.0f,0.0f,1.0f)}};

GlBegin // Start the actual rendering of the crystal here
if(MouseDrag==false && RMouseDrag == false && keys['A']==false && keys['D']==false && keys['W']==false && keys['S']==false && keys['Z']==false && keys['X']==false && GetKeyState(VK_CAPITAL)) {
  for(int i=firstframe;i<lastframe;i++) // loop
    through all the vertices
    
    ty=vertices.points[i].y; // make

    tz=vertices.points[i].z; // make

    cv=vertices.points[i].c;

    if(frames.ColourMode <= 2)
      glColor4fv(colourcages[2][cv-1]);
    if(frames.ColourMode == 3)
      glColor4fv(colourcages[3][cv-1]);

    glTranslatef(((tx-offsetx)*0.00670f, ((ty-offsety)*0.01150f), (tz-offsetz)*0.00760f);
    glCallList(1);
    glTranslatef((-tx+offsetx)*0.00670f, (-ty+offsety)*0.01150f), (-tz+offsetz)*0.00760f);
  }
else{
  glBegin(GL_POINTS);
  for(int i=firstframe;i<lastframe;i++) // loop
    through all the vertices
  
    tx=vertices.points[i].x; // make

    temp X variable equal from helper's X variable
    ty=vertices.points[i].y; // make

    temp Y variable equal from helper's Y variable
    tz=vertices.points[i].z; // make

    temp Z variable equal from helper's Z variable
    cv=vertices.points[i].c;
if(frames.ColourMode <= 2)
glColor4fv(colourlayers[cv-1]);
if(frames.ColourMode == 3)
glColor4fv(colourcages[cv-1]);

glVertex3f((tx-offsetx)*0.00670f,(ty-offsety)*0.01150f,(tz-offsetz)*0.00760f);
}
glEnd();
}

// Start the actual rendering of the crystal here
if(MouseDrag==false && RMouseDrag == false && keys['A']==false && keys['D']==false && keys['W']==false && keys['S']==false && keys['Z']==false && keys['X']==false && !GetKeyState(VK_CAPITAL)) {
    for(int i=firstframe;i<lastframe;i++)
        // loop through all the vertices
        {
            tx=vertices.points[i].x;
            // make temp X variable equal from helper's X variable
            ty=vertices.points[i].y;
            // make temp Y variable equal from helper's Y variable
            tz=vertices.points[i].z;
            // make temp Z variable equal from helper's Z variable
            cv=vertices.points[i].c;

            if(frames.ColourMode <= 2)
glColor4fv(colourlayers[cv-1]);
            if(frames.ColourMode == 3)
glColor4fv(colourcages[cv-1]);
}

Appendices – Appendix 5
j = int(tz);

    if (j % 2 == 0){
        glTranslatef((tx-offsetx)*0.00670,((ty-offsety)*(0.01150)),(tz-offsetz)*0.00760);
        glCallList(1);
    } else{
        glTranslatef((tx-offsetx)*0.00670,(ty-offsety)*(0.01150),(-tz+offsetz)*0.00760);
    }
}
    glEnd();
    glBegin(GL_POINTS);
    for(int i=firstframe;i<lastframe;i++) // loop through all the vertices
    {

        tx=vertices.points[i].x; // make temp X variable equal from helper’s X variable
        ty=vertices.points[i].y; // make temp Y variable equal from helper’s Y variable
        tz=vertices.points[i].z; // make temp Z variable equal from helper’s Z variable
        cv=vertices.points[i].c;

        if(frames.ColourMode <= 2)
            glColor4fv(colourlayers[cv-1]);
        if(frames.ColourMode == 3)
            glColor4fv(colourcages[2][cv-1]);

        glVertex3f((tx-offsetx)*0.00670f,(ty-offsety)*0.01150f,(tz-offsetz)*0.00760f);
    }
    glEnd();
}
void SOD2D(int firstframe, int lastframe, int offsetx, int offsety, int offsetz)
{
    int tx, ty, tz, cv;

    if(MouseDrag==false & & RMouseDrag == false & & keys['A']==false & & keys['D']==false & & keys['W']==false & & keys['S']==false & & keys['Z']==false & & keys['X']==false & & !GetKeyState(VK_CAPITAL)) { // GetKeyState(VK_CAPITAL)) {
        for(int i=firstframe;i<lastframe;i++)
        {
            tx=vertices.points[i].x;
            ty=vertices.points[i].y;
            tz=vertices.points[i].z;
        }
cv=vertices.points[i].c;

glTranslatef((tx-offsetx)*0.01201f,(ty-offsety)*0.01201f,(tz-offsetz)*0.01201f);
if (tx==1||ty==5||tz==1)
    glCallList(1);
else if (tx==2||ty==6)
    glCallList(2);
else if (tx==3||ty==7)
    glCallList(3);
else if (tx==4||ty==8)
    glCallList(4);

else
    glBegin(GL_POINTS);
    for(int i=firstframe;i<lastframe;i+=stride)
        { 
        tx=vertices.points[i].x;
        ty=vertices.points[i].y;
        tz=vertices.points[i].z;
        //
        cv=vertices.points[i].c;
            if (tx==1||ty==5||tz==1)
                glColor4f(0.55f,0.28f,0.10f,1.0f);
            else if (tx==2||ty==6)
                glColor4f(0.55f,0.48f,0.20f,1.0f);
            else if (tx==3||ty==7)
                glColor4f(0.70f,0.48f,0.10f,1.0f);
            else if (tx==4||ty==8)
                glColor4f(0.80f,0.68f,0.35f,1.0f);
        }
    glEnd();
    }
}

void LTA3D(int firstframe, int lastframe, int offsetx, int offsety, int offsetz)
{
    int tx, ty, tz, cv;
    GLfloat colourlayers[3][5][4] =
        {{{0.80f,0.68f,0.35f,1.0f}},{0.70f,0.48f,0.10f,1.0f},{0.55f,0.28f,0.10f,1.0f}},{0.40f,0.18f,0.05f,1.0f}},{0.40f,0.18f,0.05f,1.0f}};
    GLfloat colourcages[3][5][4] =
        {{{1.0f,0.0f,0.0f,1.0f},{0.88f,0.0f,0.0f,1.0f}},{0.76f,0.0f,0.0f,1.0f}},{0.65f,0.0f,0.0f,1.0f}},{0.40f,0.18f,0.05f,1.0f}};
        {{{0.0f,1.0f,0.0f,1.0f}},{0.0f,0.88f,0.0f,1.0f}},{0.0f,0.76f,0.0f,1.0f}},{0.0f,0.65f,0.0f,1.0f}},{0.40f,0.18f,0.05f,1.0f}};
        {{{0.0f,0.0f,1.0f,1.0f}},{0.0f,0.0f,0.88f,1.0f}},{0.0f,0.0f,0.76f,1.0f}},{0.0f,0.0f,0.65f,1.0f}},{0.40f,0.18f,0.05f,1.0f}};
```c
if(MouseDrag==false && RMouseDrag == false && keys['A']==false && keys['D']==false && keys['W']==false && keys['S']==false && keys['Z']==false && keys['X']==false && !GetKeyState(VK_CAPITAL)) {
    for(int i=firstframe;i<lastframe;i++) {
        tx=vertices.points[i].x;
        ty=vertices.points[i].y;
        tz=vertices.points[i].z;
        cv=vertices.points[i].c;

        if((int(tx) % 2 == 0) && (int(ty) % 2 != 0) && (int(tz) % 2 != 0)) {
            if(faces.ColourMode == 3)
            glColor4fv(colourlayers[cv-1]);

            glBegin(GL_TRIANGLES);
            glVertex2f((tx-offsetx)*0.01149f,ty-offsety)*0.01149f, (ty-offsety)*0.01149f, (-ty+offsetz)*0.01149f);
            glEnd();
        } else if((int(tx) % 2 == 0) && (int(ty) % 2 == 0) && (int(tz) % 2 == 0)) {
            if(faces.ColourMode == 3)
            glColor4fv(colourlayers[cv-1]);

            glBegin(GL_TRIANGLES);
            glVertex2f((tx-offsetx)*0.01149f,ty-offsety)*0.01149f, (ty-offsety)*0.01149f, (-ty+offsetz)*0.01149f);
            glEnd();
        } else if((int(tx) % 2 == 0) && (int(ty) % 2 == 0) && (int(tz) % 2 == 0)) {
            if(faces.ColourMode == 3)
            glColor4fv(colourlayers[cv-1]);

            glBegin(GL_TRIANGLES);
            glVertex2f((tx-offsetx)*0.01149f,ty-offsety)*0.01149f, (ty-offsety)*0.01149f, (-ty+offsetz)*0.01149f);
            glEnd();
        } else {
            if(faces.ColourMode == 3)
            glColor4fv(colourlayers[cv-1]);

            glBegin(GL_TRIANGLES);
            glVertex2f((tx-offsetx)*0.01149f,ty-offsety)*0.01149f, (ty-offsety)*0.01149f, (-ty+offsetz)*0.01149f);
            glEnd();
        }
    }
}
```

glBegin(GL_POINTS);
for(int i=firstframe;i<lastframe;i++)
{
    tx=vertices.points[i].x;
    ty=vertices.points[i].y;
    tz=vertices.points[i].z;
    cv=vertices.points[i].c;

    if(frames.ColourMode <= 2)
        glColor4fv(colourlayers[cv-1]);

    if((int(tx) % 2 == 0) && (int(ty) % 2 != 0) &&
        (int(tz) % 2 != 0))
        if(frames.ColourMode == 3)
            glColor4fv(colourcages[1][cv-1]);
    else if((int(ty) % 2 == 0) && (int(tz) % 2 != 0) &&
                  (int(tz) % 2 == 0))
        if(frames.ColourMode == 3)
            glColor4fv(colourcages[0][cv-1]);
    else if((int(tx) % 2 != 0) && (int(ty) % 2 == 0) &&
                  (int(tz) % 2 == 0))
        if(frames.ColourMode == 3)
            glColor4fv(colourcages[0][cv-1]);
    else
        if(frames.ColourMode == 3)
            glEnd();
        glEnable(GL_LIGHTING);
        glEnd();
}

void HUD(int lstx, int lsty, int lstz, int gtx, int gty, int gtz, 
int offsetx, int offsety, int offsetz)
{
    int sizex, sizey, sizez;

    // The X-axis line for measuring
    glBegin(GL_QUADS);
        glColor3f(1.0f,1.0f,1.0f);
        glNormal3f(0.0f,0.0f,-1.0f);
        glVertex3f(lstx-offsetx)*0.008495f,(lsty-offsety)*0.008495f,(gtz-offsetz)*0.008495f);
        glVertex3f((gtx-offsetx)*0.08495f,(gtz-offsetz)*0.008495f);
        glVertex3f((gtx-offsetx)*0.008495f,(gyt-offsetz)*0.008495f);
        glVertex3f(1.0f,0.0f,1.0f);
    glEnd();
}

Appendices – Appendix 5
void draw() {
    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glBegin(GL_TRIANGLES);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glEnd();
    glBegin(GL_TRIANGLES);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glEnd();
    glBegin(GL_TRIANGLES);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glVertex3f((lstx-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
    glEnd();
    glPopAttrib();
}

}
glVertex3f((grtx-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f);
glNormal3f( 0.0f, 0.0f, -1.0f);
glVertex3f((lsty-offsetx)*0.008495f, (grty+1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (grty+1-offsetz)*0.008495f);
glNormal3f(1.0f, 0.0f, 0.0f);
// Bottom Triangle
glVertex3f((grty-offsetz)*0.008495f, (lsty+1-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (grty-0.5f-offsetz)*0.008495f);
glNormal3f(-1.0f, 1.0f, 0.0f);
glVertex3f((grty-offsetz)*0.008495f, (grty-1-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lsty-0.5f-offsetx)*0.008495f, (grty-0.5f-offsetz)*0.008495f);
// The y-axis line for measuring
glBegin(GL_QUADS);
glNormal3f(0.0f, 0.0f, -1.0f);
glVertex3f((lsty-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (lsty-2-offsetz)*0.008495f);
glVertex3f((lsty-1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
glVertex3f((lsty-1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-offsetz)*0.008495f, (grty+1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (lsty-1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty+1-offsetz)*0.008495f, (grty+1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f, (grty+1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f, (grty-1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f, (grty-1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f, (grty-1-offsetz)*0.008495f);
glVertex3f((lsty+1-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (grty-1-offsetz)*0.008495f, (grty-1-offsetz)*0.008495f);
}

Appendices – Appendix 5
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (grty-offsetz)*0.008495f);
glNormal3f(-1.0f, 0.0f, 0.0f);
glEnd();
// left Triangle
// Back Triangle
glNormal3f(0.0f, 1.0f, 1.0f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (grty-offsetz)*0.008495f);
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (grty-offsetz)*0.008495f);
// The Z-axis line for measuring
} // End of § 4.0
glVertex3f((lstx-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (lstz-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glNormal3f(0.0f, 0.0f, 1.0f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);
glVertex3f((lstx-1-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);

// left Triangle

glBegin(GL_TRIANGLES);

// left Triangle

glNormal3f(1.0f, 0.0f, 1.0f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);
glVertex3f((lstx-2-offsetx)*0.008495f, (lsty+1-offsety)*0.008495f, (lstz-1-offsetz)*0.008495f);

// left Triangle

glNormal3f(1.0f, 0.0f, 0.0f);

// left Triangle

glVertex3f((lstx-2-offsetx)*0.008495f, (lsty-2-offsety)*0.008495f, (lstz-2-offsetz)*0.008495f);

// left Triangle

glEnd();
void DrawScene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
  glTranslatef(0.0f, 0.0f, 0.0f);
  glTranslatef(transx, transy, transz);
  glRotatef(RotationX, 1.0, 0.0, 0.0);
  glRotatef(RotationY, 0.0, 1.0, 0.0);
  glRotatef(RotationZ, 0.0, 0.0, 1.0);

  if (frames.lines == 1) glEnable(GL_POLYGON_OFFSET_FILL);
  if (frames.lines == 1) glPolygonOffset(1, 1);

  glNormal3f(0.0f, 0.0f, 1.0f);
  glVertex3f((lstx - 0.5f - offsetx) * 0.008495f,
             (lsty - 0.5f - offsety) * 0.008495f,
             (lstz - offsetz - 4.0f) * 0.008495f);

  // FRONT Triangle
  glVertex3f((lstx - offsetx) * 0.008495f,
             (lsty - offsety) * 0.008495f,
             (lstz - offsetz) * 0.008495f);
  glVertex3f((lstx + offsetx) * 0.008495f,
             (lsty + offsety) * 0.008495f,
             (lstz + offsetz + 4.0f) * 0.008495f);
  glVertex3f((lstx + offsetx) * 0.008495f,
             (lsty + offsety) * 0.008495f,
             (lstz + offsetz) * 0.008495f);

  // right Triangle
  glVertex3f((-1.0f, 0.0f, 1.0f) + offsety * 0.008495f,
             (lstz - offsetz) * 0.008495f);
  glVertex3f((-1.0f, 0.0f, 1.0f) - offsety * 0.008495f,
             (lstz - offsetz) * 0.008495f);
  glVertex3f((-1.0f, 0.0f, 1.0f) - offsety * 0.008495f,
             (lstz + offsetz + 4.0f) * 0.008495f);

  // BACK Triangle
  glVertex3f(0.0f, 1.0f, 0.0f) - offsety * 0.008495f,
             (lstz - offsetz) * 0.008495f);
  glVertex3f(0.0f, 1.0f, 0.0f) + offsety * 0.008495f,
             (lstz - offsetz) * 0.008495f);
  glVertex3f(0.0f, 1.0f, 0.0f) + offsety * 0.008495f,
             (lstz + offsetz + 4.0f) * 0.008495f);

  glEnd();

  // The text for the measurements
  sizex = int((grtx - lstx) * 0.6);
  sizey = int((grty - lsty) * 0.6);
  sizez = int((grtz - lstz) * 0.6);

  glColor3f(1.0f, 1.0f, 1.0f);
  glRasterPos3f(0.0f, (lsty - offsety - 30) * 0.008495f,
                (grtz - offsetz) * 0.008495f);
  glPrint("%in", sizex);

  glColor3f(1.0f, 1.0f, 1.0f);
  glRasterPos3f((lstx - offsetx - 50) * 0.008495f,
                0.0f, (grtz - offsetz) * 0.008495f);
  glPrint("%in", sizey);

  glColor3f(1.0f, 1.0f, 1.0f);
  glRasterPos3f((lstx - offsetx - 50) * 0.008495f,
                0.0f, (lsty - offsety - 50) * 0.008495f);
  glPrint("%in", sizez);
}
if (frames.lines == 0) glEnable(GL_POLYGON_OFFSET_FILL);

// glPushMatrix();

int firstframe, lastframe, offsetx, offsety, offsetz, grtx, grty, lstx, lsty, lstz;

if (frames.dataloaded==true)
{
  firstframe = frames.frame[frames.currentframe].StartFrame;
  lastframe = frames.frame[frames.currentframe].EndFrame;
  offsetx = frames.frame[frames.currentframe].OffsetX;
  offsety = frames.frame[frames.currentframe].OffsetY;
  offsetz = frames.frame[frames.currentframe].OffsetZ;
  grtx = frames.frame[frames.currentframe].GreatestX;
  grty = frames.frame[frames.currentframe].GreatestY;
  grtz = frames.frame[frames.currentframe].GreatestZ;
  lstx = frames.frame[frames.currentframe].LeastX;
  lsty = frames.frame[frames.currentframe].LeastY;
  lstz = frames.frame[frames.currentframe].LeastZ;

  //Axes
  // HUD(lstx, lsty, lstz, grtx, grty, grtz, offsetx, offsety, offsetz, offsetx, offsety, offsetz);

  glEnable(GL_LIGHTING);
  glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
  LTL3D(firstframe, lastframe, offsetx, offsety, offsetz);
  // LTA3D(firstframe, lastframe, offsetx, offsety, offsetz);
  // 3D LTA
  // SOD3D(firstframe, lastframe, offsetx, offsety, offsetz);
  // 3D Sodalite

  if the draw polygon lines option is flagged then disable
  lights and draw them
  if (frames.lines == 1) glEnable(GL_LIGHTING);
  if (frames.lines == 1) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
  if (frames.lines == 1) LTA3D(firstframe, lastframe, offsetx, offsety, offsetz);
  // 3D LTA
  if (frames.lines == 1) SOD3D(firstframe, lastframe, offsetx, offsety, offsetz);
  // 3D Sodalite
}

  // glPopMatrix();
  glFlush();
}

Resource.h

// Menu
#define IDM_ABOUT 100
#define IDM_EXIT 101
#define IDM_OPEN 102
#define IDM_SAVE 103
#define IDM_SAVEALL 104
#define IDM_PROCESSALL 105
#define IDM_LAYERS_M1 106
#define IDM_LAYERS_M2 107
#define IDM_CAGES - 108
Appendices – Appendix 5

#define IDM_LINES 109
#define IDC_TEST2 400

// About Dialog
#define IDR_MAINFRAME 200
#define IDD_TEST2_DIALOG 201
#define IDD_ABOUTBOX 202
#ifndef IDC_STATIC
#define IDC_STATIC -1
#endif

// Icons
#define IDI_LARGE 300
#define IDI_SMALL 301

Visualiser.rc

#include "resource.h"
#include "windows.h"

// Menu

IDC_TEST2 MENU
BEGIN
POPUP "&File"
BEGIN
    MENUITEM "&Open", IDM_OPEN
    MENUITEM "&Save", IDM_SAVE
    MENUITEM "&Save All Frames", IDM_SAVEALL
    MENUITEM "&Process a Directory", IDM_PROCESSALL
    MENUITEM "E&xit", IDM_EXIT
END

POPUP "&Colouring"
BEGIN
    MENUITEM "&Layers M1", IDM_LAYERS_M1
    MENUITEM "&Layers M2", IDM_LAYERS_M2
    MENUITEM "&Cages", IDM_CAGES
END

POPUP "&View"
BEGIN
    MENUITEM "&Polygon Lines", IDM_LINES
END

POPUP "&Help"
BEGIN
    MENUITEM "&About", IDM_ABOUT
END

END

// Icons

IDI_LARGE ICON "large.ico"
IDI_SMALL ICON "small.ico"

// About Dialog

IDD_ABOUTBOX DIALOGEX 350, 150, 170, 98
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About Crystal Growth Visualizer"
Appendices – Appendix 5

FONT 8, "MS Shell Dlg"
BEGIN
  ICON IDR_MAINFRAME,IDC_STATIC,14,14,21,20
  LTEXT "Crystal Growth Visualizer",IDC_STATIC,42,14,114,8,SS_NOPREFIX
  LTEXT "Version 1.0",IDC_STATIC,42,26,114,8
  LTEXT "Center of Nanoporous Materials",IDC_STATIC,42,38,114,8
  LTEXT "The University of Manchester",IDC_STATIC,42,50,114,8
  DEFPUSHBUTTON "OK",IDOK,113,69,50,14,WS_GROUP
END
Appendices – Appendix 6

Appendix 6

Here are some of the configuration details for compiling the programs presented within this thesis.

The fortran sources have been tested against all compilers supported by The University of Manchester IT Services department. The fortran source was mainly compiled and run on the Epsilon Beowulf cluster (deprecated) and the Research Computing Services Condor Pool. In both cases the Intel compiler was used to compile the code with the following format

    ifort nameofsource.f90 –o nameofexecutable

Since Jan 2013 it was necessary to also link to the compiler by running the following line prior to compiling

    Source /opt/intel/composer_xe_2013.1.117/bin/compilervars.sh intel64

The programs used to connect to the condor pool were PuTTY (console) and FileZilla (FTP client).

The following file comprises a submit file for a condor job submission file.

    universe = vanilla
    requirements = (opsys == "LINUX") && (arch == "X86_64") && (HAS_INTEL111 =?= True)
    Rank = Memory
    request_memory = 8200
    should_transfer_files = yes
    notification = never
    executable = LTL
    input = input/input$(Process).txt
    output = output/output$(Cluster).$(Process).txt
    log = output/log.txt
    error = output/err$(Cluster).$(Process).txt
transfer_input_files = input/input$(Process).txt

when_to_transfer_output = on_exit

queue 200

This file will submit 200 (queue 200) instances of simulations using the LTL executable which will be stored with that same name and case in the same directory. The file will also take input files from a folder named input, in that folder there should be 200 input files with the simulation input information for each one. These should be stored with the following naming as specified input.x.txt where x runs from 0-199. Another folder called outputs should also be present in the same location as the executable and submit file to store error logs and outputs from the executed simulations.

The visualiser program was developed and extensively tested with Microsoft Visual C++ Express 2010. A new solution and project should be created and then the .cpp, .h and any other files imported into the project structure. In some cases large numbers of error upon compiling from a fresh project creation can occur, this can be avoided by making sure the following things are done. By right clicking the project and selecting the properties, make sure the following things are done for all configurations (debug and release).

In configuration properties > General > Character Set Multibyte

In configuration properties > Linker > Input

Make sure that the glu32.lib and opengl32.lib are listed in the dependencies list.

In addition make sure the project type is win32 and not console.