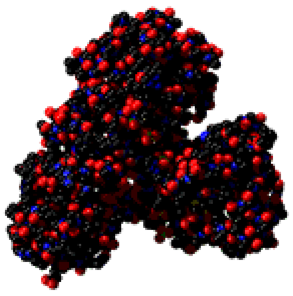
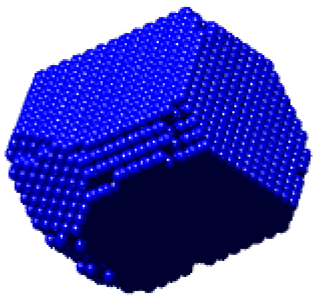
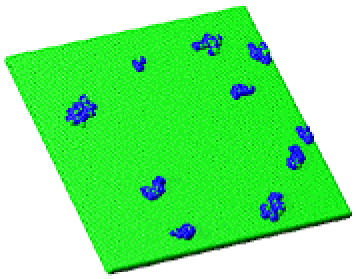
MESOSIM

Crystallographic and 3-Dimensional
Kinetic Monte Carlo Simulation of Mesoscopic Structures
For Basic Research and Education


Molecular Graphics


Voids and Precipitates


Heteroepitaxy

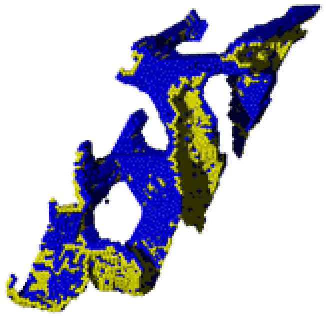

Alloy Corrosion

Table of Contents

Acknowledgement

The development of this software into user-friendly freeware was generously supported by the United States National Science Foundation under grant DMR-0092756

I. Introduction

MESOSIM was written to model morphological evolution in realistic systems comprising tens to hundreds of millions of atoms. These scales are too big for first principles calculations to be computationally feasible, and too small for continuum models to provide accurate results. MESOSIM uses the kinetic Monte Carlo algorithm (KMC) to approach a detailed description of atom motion on these so-called mesoscales.

KMC works for systems in which all types of events can be assigned a rate constant. For instance, consider vapor phase deposition. If we were to observe a crystal surface growing from the vapor on a very small scale we would see all sorts of atom motion – atoms would be diffusing on the surface, crystal steps would be fluctuating as kinks diffused along their edges, and atoms would be raining down upon the surface. For each of these motions, a rate (analogous to a chemical rate constant) can be assigned. There is a rate (sec^{-1}) at which an atom sitting on a terrace will hop from one spot to another, another rate at which an atom sitting on the step edge will diffuse along the step, and because the atoms are raining down on the surface at a constant deposition flux, there is a rate for this process as well.

MESOSIM takes a fully functional molecular graphics interface derived from the tradition of x-ray crystallography tools and merges this with a three-dimensional, computationally efficient, multi-component KMC simulation engine. A key advance in MESOSIM is to maintain in memory only those atoms that can actually do something, like diffuse. Atoms that are completely coordinated are removed from the calculation. This leaves the simulation to track only the "skin" on structures. As such, a system may have a surface area of 50000 atoms, but the actual volume simulated can be many orders of magnitude larger.

This demonstration version of MESOSIM allows the user to study morphological evolution in a number of model systems including

Heteroepitaxy
Electrochemical Etching
Nanoclusters and Nanovoids
Lattice gases

The graphical user interface of MESOSIM allows the user to observe these structures as they evolve, rotating them to find the most informative vantage point. A number of rendering styles are available to most informatively display the information presented.

This executable of this demonstration version of MESOSIM is constrained by a number of factors. Most notably, there are no explicit analysis tools included. This feature was not left out maliciously, and actually isn't gone at all. Instead, we invite the motivated user to delve into the code her/himself and customize the program to their own project of interest.

I hope that you find MESOSIM a useful and interesting tool, and there are some interested enough to continue to expand the usefulness of this program.

For further questions and information, I can be reached at Jonah.Erlebacher@jhu.edu

II. Installation Instructions

The package included in the MESOSIM.zip file includes

1. the source code Mesosim.exe

2. an auxiliary file entitled symopencoded.lib, which should be placed in the c:\ directory

3. code, header and resource files for use with Microsoft Visual C/C++ (actually Visual Studio .NET), which I used to develop this software. It's all there!

4. a few example molecules in a few different formats that can be opened to get a feel for the user interface:

smallmol.kmc
: a small molecule in native .kmc format

proteinboundtodna.pdb
: a protein molecule bound to a piece of dna in the protein databank format

proteinboundtodna.kmc
: same as above, but worked up a bit using MESOSIM

diamondlattice.kmc
: basis for the diamond structure; a good example for playing with the crystallography features.

III. Using the Molecular Graphics Interface

Basic Mouse Operations

Rotation

Rotation of the current molecule is accomplished by holding down the right mouse button. The farther you are from the center of the window, the faster the rotation will go.

Selection

Selection of atoms is a needed to delete them, bond them, or change their characteristics in any manner. Selection is accomplished by holding down the left mouse button to create a rectangle as you then drag the mouse. When the left mouse button is released, unselected atoms inside the rectangle are "selected" and drawn with a blue-gray box around them. Selection is a toggle operation – doubly selected atoms are de-selected.

Basic I/O

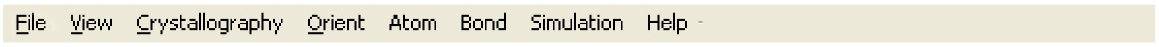
File Formats

There are four input file formats that are supported, but only one output file format (.kmc). These are more fully described below.

For graphics files, the primary output format is the command language used by the freeware program POV-RAY, a ray-tracing renderer that produced beautiful images. Ball and stick images may also be output in Encapsulated Postscript (.eps), and directly imported into pretty much any Adobe product, such as Illustrator. It seems like Word is also finally able to import .eps format without too many problems.

Menu

The menu bar of MESOSIM provides a graphical interface to both the visual data and the simulation parameters. This section describes the procedures available via the menu bar.



The menu bar is illustrated above. The general characteristics of each menu heading are listed below

File: Input and output of coordinate files, printing options, exiting the program.

View: Change the "look and feel" of the graphics interface – how the pictures are drawn and the coordinate system.

Crystallography: Control the automatic generation of crystallographic equivalents to an "asymmetric unit" – quickly generate large number of crystal atoms.

Orient: Change the orientation of the atoms on the screen.

Atom: Change attributes of particular atoms. Add/delete atoms

Bond: Change attributes of particular bonds. Add/delete bonds

Simulation: Enter simulation parameters. Run simulations.

Help: No help is available, yet, but if you're reading this manual then hopefully your question will be answered.

File : Open

MESOSIM recognizes four file formats for xyz representation of atom coordinate data: Protein Data Bank files (.pdb), ICRVIEW formatted files (.vue), simple xyz coordinates in a list, and the native .kmc format most compatible with this code.

Opening a file does not close the current file, it simply overlays it. See the "View : Lattice" command for implications of this.

The Brookhaven Protein Data Bank (PDB) format is one of the most common formats for crystallographic data. MESOSIM's functionality in reading PDB files is limited. It will read coordinates and cell dimensions only. Bonding, symmetry, and other formatting information will not be entered except possible incorrectly. Be sure to check that what you've entered makes sense!

ICRVIEW is a shareware molecular graphics program developed at the Fox Chase Cancer Center (copyright Erlebacher and Carrell, 1992). MESOSIM'S graphics interface derives significantly from ICRVIEW. All view files, including formatting information, bonding, orientation information, etc. are available to MESOSIM.

The .xyz format is useful for a quick input of an atom list. The format is particularly simple, consisting only of a list in the following format

atomname1, x, y, z
atomname2, x, y, z
etc.

where *atomname1* is the name of the atom, and *x,y,z* are its orthogonal coordinates (versus fractions of lattice vectors as most commonly used in the other formats).

File : Close

Close the current file. All atom coordinate data and all bonding data will be lost. For this reason, you will be prompted to confirm closing upon executing this command.

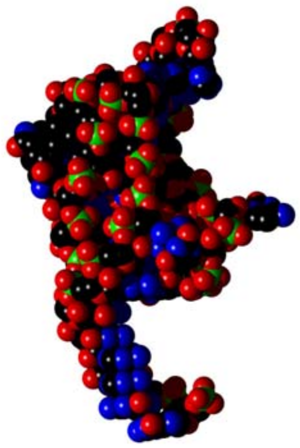
File : Export

This operation exports the current view of the molecule using one of two options, (1) POV-RAY format, or (2) EPS format.

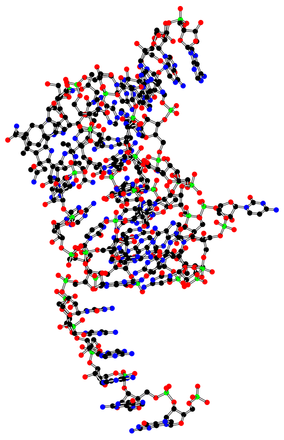
1. Export .POV ray: POV-RAY is a truly excellent freeware ray tracing program, which will accept files exported with this option. Atoms are drawn as spacefilling spheres; bonds are not drawn at all. The spheres are rendered shiny, with a light source off to the side. If you know how to edit .pov files using the POV-RAY editor, your options are limitless. POV-RAY is available for download at www.povray.com
2. Export the Adobe encapsulated postscript (.EPS) format: This option outputs current view with the classical crystallography "look" (sometime referred to as ORTEP-format) in encapsulated postscript format. This format is editable using software such as Adobe Illustrator.

Here are examples of what the different export options look like:

POV-RAY



Encapsulated Postscript



File : Export GIF

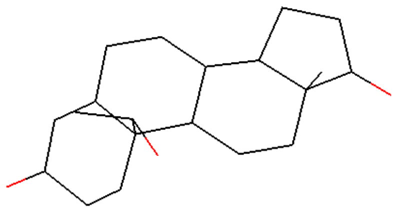
The export GIF function automatically rotates the molecule on the screen by 360 degrees, exporting a .pov image at each degree increment. I've used POVRAY to create an .gif images based on these output files that can be assembled into one giant "animated gif" that can be used in websites.

File : Exit

Close the current molecule and exit the program.

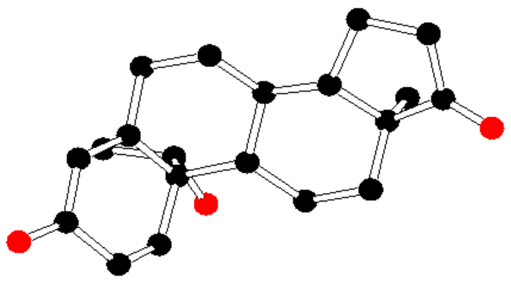
View : Draw As : Lines and Points

Draw the molecule with atoms as points and bonds as lines. An example (images on this page are screen-captured):



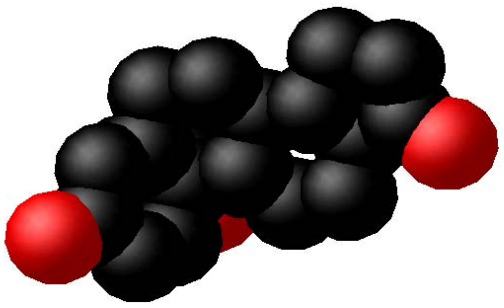
View : Draw As : Balls and Sticks

Draw the molecule with flat rendering and with circles for atoms and cylinders for bonds. The radius is set using "Atom : Radius : Balls and Sticks" or "Bond : Radius".



View : Draw As : Space Filling

Draw the molecule as spheres. The radius is set using "Atom : Radius : Space Filling".



View : Draw Unit Cell

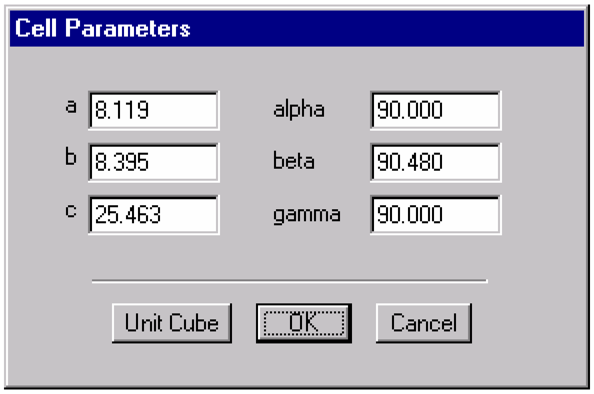
This command toggles drawing (on-screen only; not exported) the unit cell or cells currently occupied by the atoms being imaged. Some clipping of the unit cell often occurs, depending on rotation.

View : Console

This command opens up a DOS-style command line window that can be used to monitor a simulation, etc. There is a command line language for this console that is currently undocumented, but is essentially the same as the command line used in ICRVIEW. I'll write up documentation for this console at a later date – for now I consider it a useful diagnostic tool for more advanced (and adventurous!) users of the code.

Crystallography : Unit Cell

The lattice command is used to change unit cell axis lengths and angles. a,b,c are the axial lengths, in Angstroms; α, β, γ are the axial angles, in degrees. The Lattice dialog box is illustrated in Figure 3.2.6. The Unit Cube button returns a,b,c,α, β, γ to their default values of $a = 1$, $b = 1$, $c = 1$, $\alpha = 90.0$, $\beta = 90.0$, $\gamma = 90.0$.


Cell Parameters

a	8.119	alpha	90.000
b	8.395	beta	90.480
c	25.463	gamma	90.000

The primary purpose of this command is to transform input fractional coordinates into orthogonal coordinates that can be displayed on the screen. If the coordinates to be viewed have been already orthogonalized, and have units of Angstroms, the cell parameters should be $a = 1$, $b = 1$, $c = 1$, $\alpha = 90.0$, $\beta = 90.0$, $\gamma = 90.0$. This sets the cell orthogonalization matrix to the identity matrix.

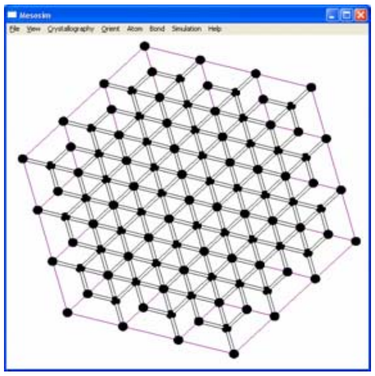
Be careful to check the cell dimensions when inputting new files! Let's say that you have been working on a set of coordinates in which you have specified cell parameters. Then you clear the atoms to make room for a new file. If the new file you enter has orthogonalized coordinates orthogonalized already, then it expects the cell dimensions to be $a = 1$, $b = 1$, $c = 1$, $\alpha = 90.0$, $\beta = 90.0$, $\gamma = 90.0$. However, if you do not reset these yourself, the cell parameters from the previous set of coordinates are retained and your picture will be skewed. Calculated bond distances, angles, and torsion angles will be wrong. This aspect of the program is useful, however, if you want to overlap sets of coordinates that have been orthogonalized in a similar manner.

Crystallography : Build Lattice

This command generates symmetric equivalents from a basis of atoms (the "asymmetric unit". It's use is perhaps best described with an example:

The diamond cubic lattice of carbon has the following cell dimensions: $a = b = c = 3.5670$ Angstrom, $\alpha = \beta = \gamma = 90^\circ$. The space group of the diamond cubic lattice is $Fd\overline{3}m$, which has 192 equivalent positions and is pretty complex. A simpler example that gives the gist is the $P\overline{1}$ space group which just contains and inversion center. For $P\overline{1}$, if there is an atom at (x, y, z) (in fractional cell coordinates), then there must also be an atom at $(1-x, 1-y, 1-z)$ in the unit cell. Therefore, we only need to specify one atomic coordinate and the space group. Getting back to diamond, for the particular case of carbon, most of these 192 equivalent positions are degenerate (i.e., they overlap), so the structure doesn't really contain 192 atoms/unit cell. In fact, the basis of atoms with coordinates $[(0,0,0),(1/4,1/4,1/4)]$ can be used as the asymmetric unit.

Within mesosim, use the **Crystallography : Unit Cell** menu to set the lattice parameters with the values above. Then, use the **Atom : Add** menu to add two atoms at $(0,0,0)$ and $(0.25, 0.25, 0.25)$. Select these atoms using the mouse or the **Atom : Select All** menu. Choose the **Crystallography : Build Lattice** command. All 230 space groups are represented. Choose #227 ($Fd\overline{3}m$, origin choice 1), and set it to build 3 x 3 x 3 unit cells, so 27 unit cells will be represented. (you may want to choose **View : Draw Unit Cell**). You'll see that all of the atoms in this volume are automatically generated. Since they are carbons, you can easily bond them up using **Bond : Create Smart**. Here's the image oriented looking onto the (111) plane (using **Orient : Crystallographic Plane**).



All 230 space groups are incorporated into symopencoded.lib, but I give no guarantee that I typed them in correctly. If you think there's a problem, please let me know.

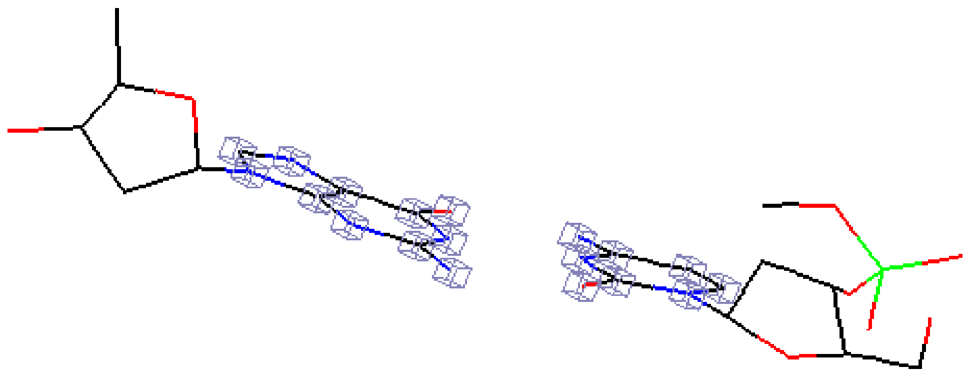
Orient : Default

Set the viewing angle to the default orientation

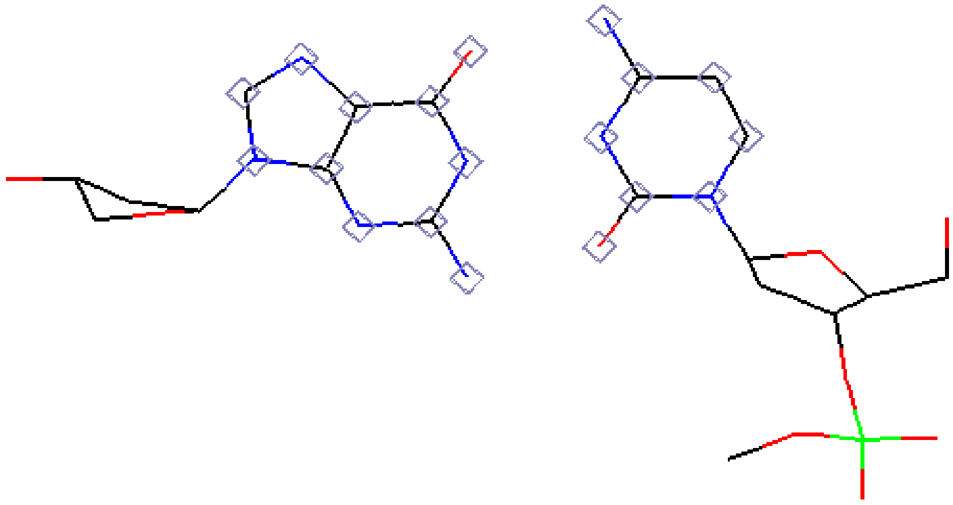
Orient : Best Plane

Set the viewing angle to be normal to the best plane through the currently selected atoms. An example:

before



after

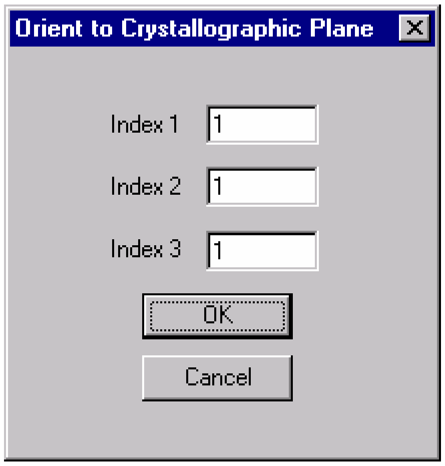


Orient : Along Bond

Aligns the current molecule such that the z-axis (the axis perpendicular to the screen) is along the line joining two selected atoms. This command will not do anything if any number of atoms other than two is selected.

Orient : Crystallographic Plane

Sets the viewing direction to be normal to the plane specified by the Miller indices in the dialog box.


Orient to Crystallographic Plane

Index 1: 1

Index 2: 1

Index 3: 1

Orient : Axis : a, b, c, a*, b*, c*

Orient the particular axis or reciprocal axis with the normal to the picture surface. The default orientation is often with c* perpendicular to the picture plane.

Atom : Select All, Select None

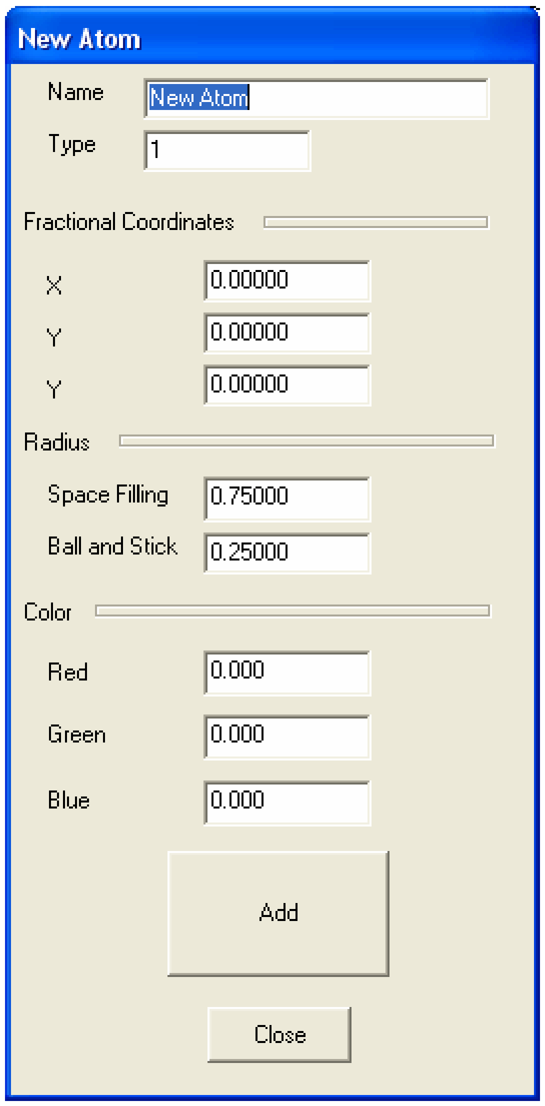
Mark either all atoms as selected, or none of them.

Atom : Delete

Delete selected atoms. There is no undo.

Atom : Add

This command allows the user to add one atom at a time. The fractional coordinates, color, etc., can be specified. Pushing the "Add" button will not close the window, so you can add one atom after the next with only minor changes between each.


New Atom

Name: New Atom
Type: 1

Fractional Coordinates
X: 0.00000
Y: 0.00000
Y: 0.00000

Radius
Space Filling: 0.75000
Ball and Stick: 0.25000

Color
Red: 0.000
Green: 0.000
Blue: 0.000

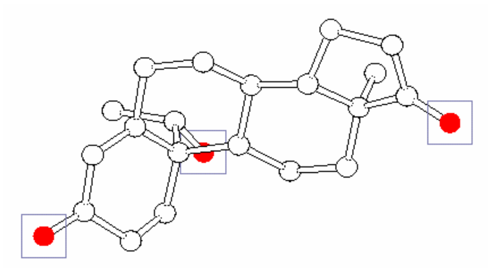
Atom : Color

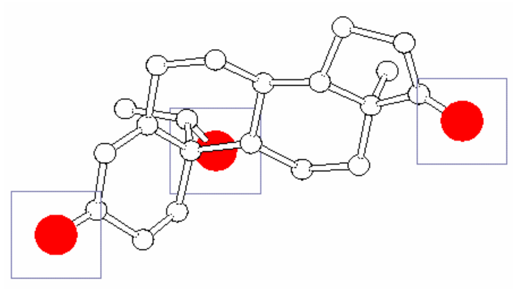
Brings up the standard Windows color dialog box, allowing one to change the color of selected atoms.

Atom : Radius : Balls and Sticks

The radius command brings up a dialog box allowing one to change the radius of selected atoms. Each atom contains two internal radius fields – one for balls and sticks mode, the other for space filling mode. Upon execution of this command a dialog box will appear allowing one to specify a new radius, in Angstroms.

Here's an illustration of using the radius command: (a) balls and sticks radius = 0.25 Å, (b) balls and sticks radius = 0.50 Å.

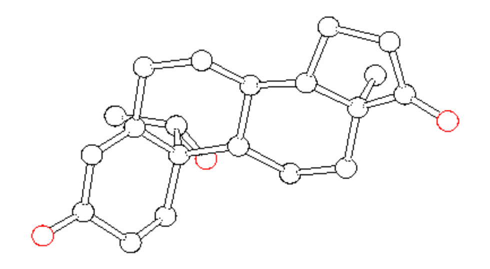


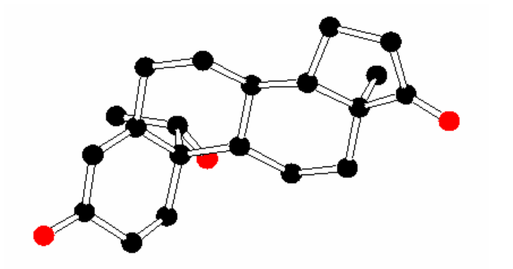


(a) (b)

Atom : Style : Outline, Filled

In balls and sticks mode, atoms can either be drawn as filled or open circles. Heres how each option looks: Atoms drawn as open circles (a), or as filled circles (b).



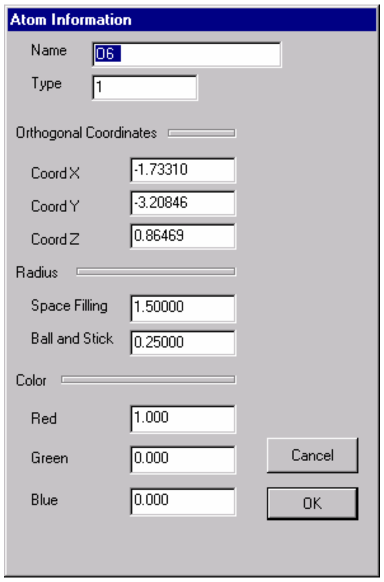


(a) (b)

Atom : Info

Pulls up a dialog box showing the name, type, color, radii, and *orthogonalized* coordinates of the selected atom.

This command will work only when one atom is selected.



Name: O6
Type: 1
Orthogonal Coordinates
Coord X: -1.73310
Coord Y: -3.20846
Coord Z: 0.86469
Radius
Space Filling: 1.50000
Ball and Stick: 0.25000
Color
Red: 1.000
Green: 0.000
Blue: 0.000

Atom : Chaff

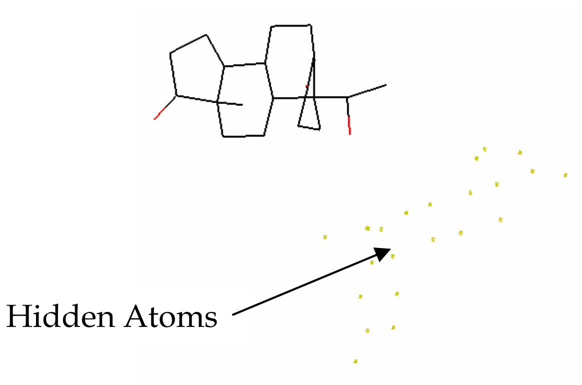
Chaff eliminates duplicate atoms. An atom is a duplicate of another if it has the same name and occupies the same position in space.

Chaff is a good command to run after generating symmetry equivalents. Chaff will get rid of atoms that were duplicated over special positions. It will also get rid of the overlapping vertices of adjoining unit cell boxes. Granted, symmetry operations have not yet been incorporated into MESOSIM, but they will be soon.

Chaff does not analyze the number and kind of bonds connected to duplicate atoms. It arbitrarily chooses which duplicate to delete, and the bonds from that duplicate will also be deleted. For example, say duplicate one has two bonds, and duplicate two has three bonds. Chaff may, for instance, choose to delete atom one. In this case, only the bonds that were connected to atom two will remain, and the lost bonds will have to be regenerated.

Atom : Hide

Hide selected atoms. Hidden atoms will be drawn as a small yellow box around them in lines and points drawing mode, as shown below. They will not appear in other drawing modes, i.e., balls and sticks or space filling mode. They will also not appear in any exported print output. Hidden atoms can still be selected, deleted, their attributes can be changed, etc. Hiding atoms might cause memory leaks. Please use this option with caution.



Atom : Reveal

Selected hidden atoms will be re-established as visible.

Bond : Create

Creates a bond between two selected atoms. If more than two atoms are selected, this command will be ineffective.

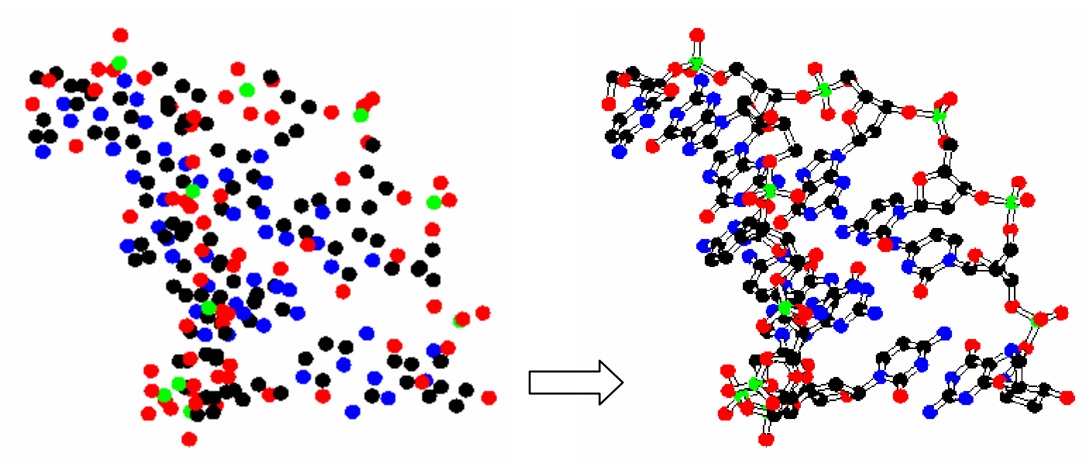
Bond : Create Smart

Create Smart is used to create multiple bonds between atoms in a pseudo-intelligent manner. The Create Smart algorithm goes through all potential bonds in the selected group. It then checks whether each bond is physically possible, based on bond distance considerations. For instance, if the routine were checking whether to create bonds between two atoms, C1 and C5, and the bonds distance between these atoms was 1.5 Å, a bond would be created. If the distance were 3.7 Å, no bond would be created. Create Smart will not create bonds between hydrogen atoms.

The bond distances for which Create Smart will create bonds are:

1.2 Å to 1.9 Å:	C-C, C-O, C-N, N-N, N-O, O-O
0.5 Å to 1.2 Å:	C-H, O-H, N-H, C-O

Create smart analyzes the atom type by checking the first letter of the particular atom name. If the atom is a carbon, but its name does not begin with C, depending on the first letter, the atom may or may not get bonded. It will certainly not get treated like a carbon.



Above is an example of Create Smart applied to atoms comprising a segment of a DNA structure.

Bond : Delete

Deletes bonds between selected atoms. There is no undo.

Bond : Color

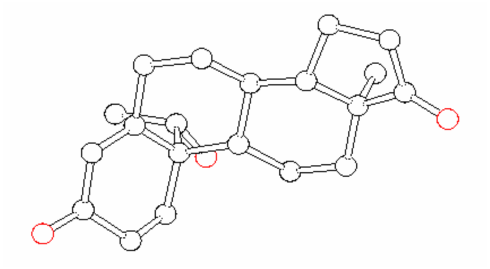
Brings up the standard Windows color dialog box, allowing one to change the color of bonds between selected atoms.

Bond : Radius

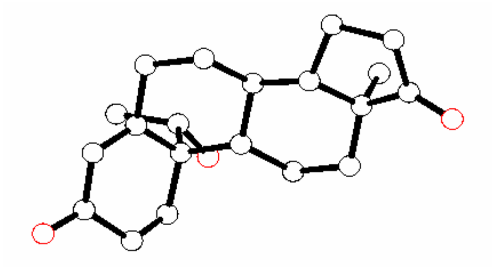
Allows the user to change the bond radius of those bonds between selected atoms.

Bond : Style : Outline, Filled

In balls and sticks mode, bonds can either be drawn as filled or open cylinders. The figure below illustrates the difference in look.



(a)



(b)

IV. Using the Simulation Interface

There are three aspects to running a simulation:

1. Setting the parameters for the simulation. Physical inputs to the simulation like temperature, bond energies are input through the **Simulation : Initialize Parameters** dialog box. The geometry of the initial conditions may be modified by adding new functions to the code in a straightforward manner.

2. Initializing the simulation. Here, the geometry of the simulation is made real, and the user can examine the initial conditions to make sure things are going to go smoothly.

3. Controlling simulation flow. This means starting it going, stopping, pausing, etc. This is modified using the **Simulation : Start, Stop, Pause,** and **Resume** menu commands. Physical parameters may be modified at any time during the simulation run using the **Simulation : Change Physics** command.

In this section, we will describe how to set up a simulation representing heteroepitaxial growth, in which atoms ("B" atoms) drop on the surface of a crystal comprised of "A" atoms.

Other example simulation modules are included:

- Equilibration of spherical atom clusters
- Watching vacancies diffuse in bulk crystals
- Solid state capillary action ("surface relaxation") by which rough surfaces smoothen
- A 2-D lattice gas

Understanding KMC simulations

To understand the other parameters in this dialog box, it is necessary to describe some detailed physics in the Kinetic Monte Carlo (KMC) algorithm.

KMC begins with the observation that all atom motion on crystals occur via motion that can be described as occurring at some rate. For instance, atom diffusion on surfaces often can be described using the Arrhenius form, i.e., atoms diffuse on the surface with a rate k_{diff} (sec^{-1}) equal to

$$k_{diff} = \nu \exp\left(-E/k_B T\right),$$

where ν is attempt frequency (set to 10^{13} sec^{-1}), k_B is Bolztmann's constant (eV K^{-1}), T is the temperature (K), and E is the activation barrier to diffusion (eV).

Most generally, if one has a system of atoms, each atom is able to diffuse about in different directions, dissolve away, etc. – lots of different options. You can make a list of all the rates for the transitions a particular atom may be perform. KMC starts with a list of all the transitions *all* of the atoms can perform. Clearly, with many atoms and a complex geometry, this list is very long and very complicated. Let's say at any given time there are a total of N transitions. At this time, we can then label each transition by its particular rate, k_i, where $i < N$.

So, we have this list of rates, $\{k_i\}$. It is natural to ask how long will it take for something to happen? A bit of thought will yield the result:

$$\Delta t = \frac{1}{\sum_{i=1}^{N} k_i}.$$

Think about it this way: if there are two things that can happen, and each happens with a rate of 10 sec^{-1}, then the average rate of something happening is 20 sec^{-1}, and the average time for one thing to happen is 1/20 sec. (Technically, there is a logarithmic correction to the above formula, but the error is small if the system size is large).

The KMC algorithm works by first calculating Δt. Then, it asks what is the probability of each transition having occurred within this time? The total probability must equal 1, so the probability P_i of transition i having occurred must be

$$P_i = \frac{k_i}{\sum_{i=1}^{N} k_i}$$

By picking a random number between 0 and 1, the KMC algorithm can thus choose what happened in that time interval Δt. Accordingly, some atom will diffuse, dissolve, deposit, etc.

At this point, the algorithm is simple. A new Δt is calculated, and new transition is picked, and the simulation progresses. The complexity of the algorithm comes from the observation that during each Δt, the transition list changes because some transitions no longer exist, and new ones are created. In fact, Δt itself changes during each iteration. The bookkeeping required to keep this flowing in an efficient manner was the biggest challenge in developing this software!

Physical inputs to KMC

A simulation is only as good as the physics that are built into it. For the demonstration purposes of the code release, most of the physics correspond to realistic, idealized "toy models".

For instance, activation energies for diffusion are calculated according the coordination of the atom in question. For an "A" atom with *n* "A" neighbors and *m* "B" neighbors, the activation energy will be

$$E = nE_{AA} + mE_{AB},$$

where E_{AA} is the A-A bond energy and E_{AB} is the A-B bond energy, both set in the dialog box. For a diffusing "B" atom, the activation energy is $E = mE_{BB} + nE_{AB}$.

An evaporation rate is also included; this has been used to simulate electrochemical dissolution. Only "B" atoms will evaporate. They evaporate with rate

$$k_D = \nu_E \exp\left(-\left(E-\phi\right)/k_B T\right),$$

where $\nu_E = 10^4$ sec^{-1}, E is the binding energy as for diffusion, and ϕ is an "evaporation potential" set in the dialog box.

The particular microscopic physics controlling each simulation may be user-modified, and we encourage explorations in this area!

A Note About the System Size

If one would like to simulate the time evolution of a small cluster of atoms, the number of atoms can increase quite dramatically. This can slow up the simulation, and severely limit the scale to which one can simulate a structure.

Consider vacancy diffusion in the bulk of an fcc crystal. To simulate a fews tens of cubic microns of material, you'd need billions of atoms, right? No! If there are say, 100,000 vacancies, then all you'd need to track are 12x100,000 atoms, because it is only those atoms surrounding the vacancy that can do anything! 1.2 million is a lot less than a few billion.

MESOSIM implements this idea by carefully tracking all atoms in the simulation. If one atom gets "buried" by other atoms, then it gets taken out of memory. The burying atoms remember that there should be an atom there, and if this atom gets unburied ("resurrected" in the code language), it and its possible transitions get added back into the simulation.

Therefore, there is no limit to the system size. The only constraint is that the systems are (a) multiples of primitive unit cell dimensions, and (b) these multiples must be powers of 2.

Crystallography of the Simulation

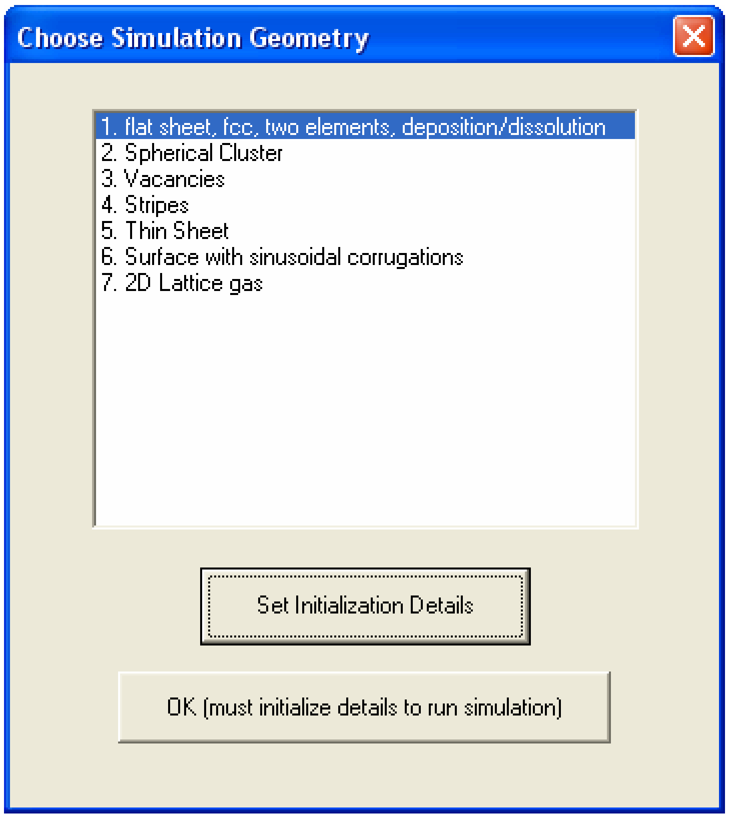
All of the example simulations involve systems constrained to the fcc lattice; however, simulation subroutines can be easily modified to use bcc or simple cubic lattices.

V. Examples

Example 1: Vapor phase deposition (heteroepitaxial growth)

Step 1. Initialize the simulation parameters

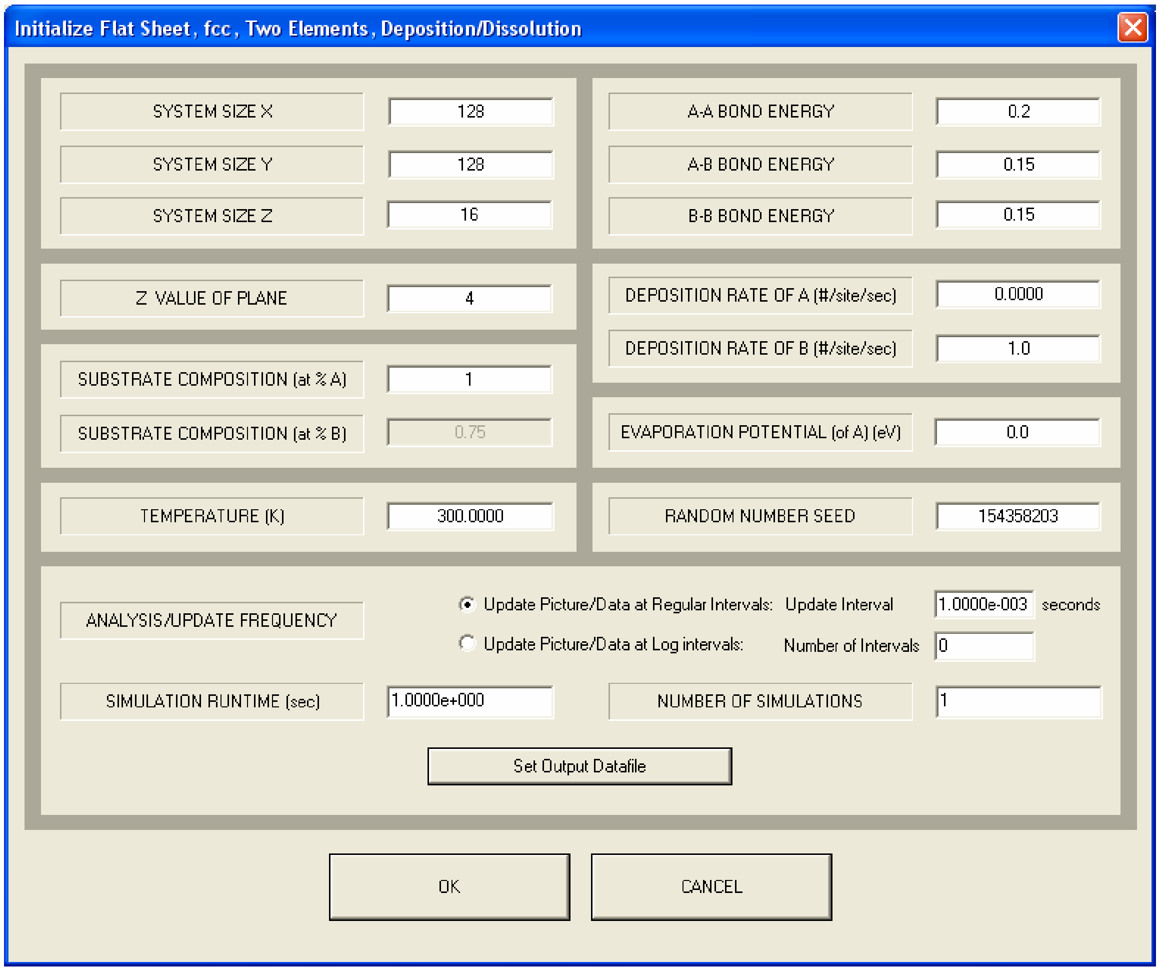
Choose **Simulation : Initialize Parameters**. You have to do this before doing anything else. A dialog box will appear:


Choose Simulation Geometry
Set Initialization Details
OK (must initialize details to run simulation)

1. flat sheet, fcc, two elements, deposition/dissolution
2. Spherical Cluster
3. Vacancies
4. Stripes
5. Thin Sheet
6. Surface with sinusoidal corrugations
7. 2D Lattice gas

The different simulation modules I've implemented appear (I invite new modules to be written and added to this list!). Choose #1, and put the "Set Initialization Details" button.

At this point, a pretty complex dialog box appear, as shown below. Let's walk through it piece by piece:


Initialize Flat Sheet, fcc, Two Elements, Deposition/Dissolution

SYSTEM SIZE X	128	A-A BOND ENERGY	0.2
SYSTEM SIZE Y	128	A-B BOND ENERGY	0.15
SYSTEM SIZE Z	16	B-B BOND ENERGY	0.15
Z VALUE OF PLANE	4	DEPOSITION RATE OF A (#/site/sec)	0.0000
SUBSTRATE COMPOSITION (at % A)	1	DEPOSITION RATE OF B (#/site/sec)	1.0
SUBSTRATE COMPOSITION (at % B)	0.75	EVAPORATION POTENTIAL (of A) (eV)	0.0
TEMPERATURE (K)	300.0000	RANDOM NUMBER SEED	154358203

ANALYSIS/UPDATE FREQUENCY

[●] Update Picture/Data at Regular Intervals: Update Interval 1.0000e-003 seconds

[ ] Update Picture/Data at Log intervals: Number of Intervals 0

SIMULATION RUNTIME (sec) 1.0000e+000 NUMBER OF SIMULATIONS 1

SYSTEM SIZE X, SYSTEM SIZE Y, and SYSTEM SIZE Z give the number of unit cells in multiples of the lattice basis vectors. This simulation is fcc, so this volume does not represent a cubic volume of space. (a note of caution: setting **View : Draw Unit Cell** is a bother with large volumes).

Set the X, Y sizes to 128 x 128, and the Z size to 16.

The Z VALUE OF PLANE is the initial plane of the substrate. Set this to 4. The simulation will make a plane comprised of A atoms. Furthermore, it will set the atoms to think that there are buried atoms below this plane that may be exposed if any A atoms diffuse away.

Set SUBSTRATE COMPOSITION to 1 (100%A).

Set Temperature to 300 K

This simulation will track B atoms depositing on the surface, so we need to input the physics of the various bond energies, as described above. Set the A-A bond energy to 0.2 eV, the B-B bond energy to 0.15 eV, and the A-B bond energy to 0.15 eV. (these are reasonably realistic for many metals)

If we set the deposition rate of B to 1/sec, then after 1 second of simulation time we should cover a single layer of B atoms. Let's do that!

Set the evaporation potential to 0.0 eV.

Choose a random number to seed the random number generator.

Set the SIMULATION RUNTIME to 1.0 sec, and the NUMBER OF SIMULATIONS to 1. One often runs multiple simulations to gather good statistics.

The ANALYSIS/UPDATE FREQUENCY tells the code how often to automatically re-draw the picture, perform some analysis routine, or perhaps to output a movie frame. You can either choose this to occur at equal time intervals or in logarithmic intervals, that spread out the intervals as time goes on.

Choose regular intervals at 0.005 seconds.

Pressing OK will take you back to the last dialog box, where you have to press OK again.

Step 2. Initialize the Simulation

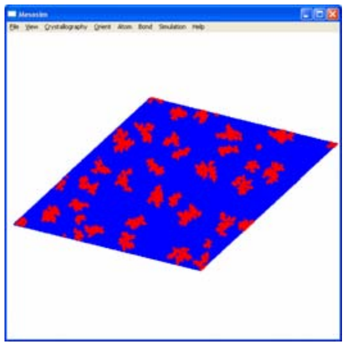
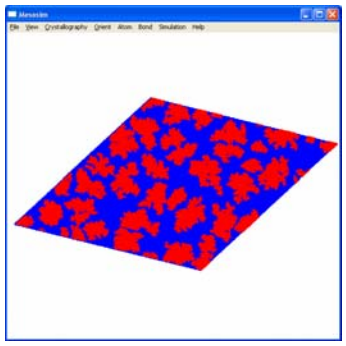
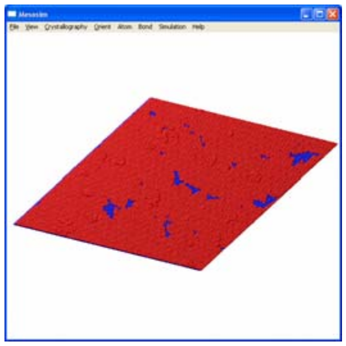
Choose **Simulation : Initialize Simulation**. You'll see a field of A atoms are generated. They are blue and you're looking down on the (111) plane.

Step 3. Start the Simulation

Choose **Simulation : Start**. Nothing will happen, seemingly! But that's because it has deposited an atom which is diffusing very quickly; it won't re-draw until 0.005 seconds elapse. You can choose, if you are bored, to override the automatic re-draw using **Simulation : Update Picture**, and you can track the progress of the simulation using **Simulation : Information Panel**, which shows the elapsed time, etc. (you can use this to update the picture, too).

You'll note that as the simulation progresses, the rate at which things happens will seem to accelerate. This is because fast transitions will become rarer and rarer; this, even though the number of atoms is increasing!

At this temperature (300 K), growth is "layer-by-layer", so that the monolayer of B will be flat. During the filling of the layer, you'll see nuclei form, grow, and coalesce. The simulation takes a little while (10 min on my PC), but passes through ~2 million iterations. Here's how it looks as the simulation progresses (the last was drawn in space-filling mode).

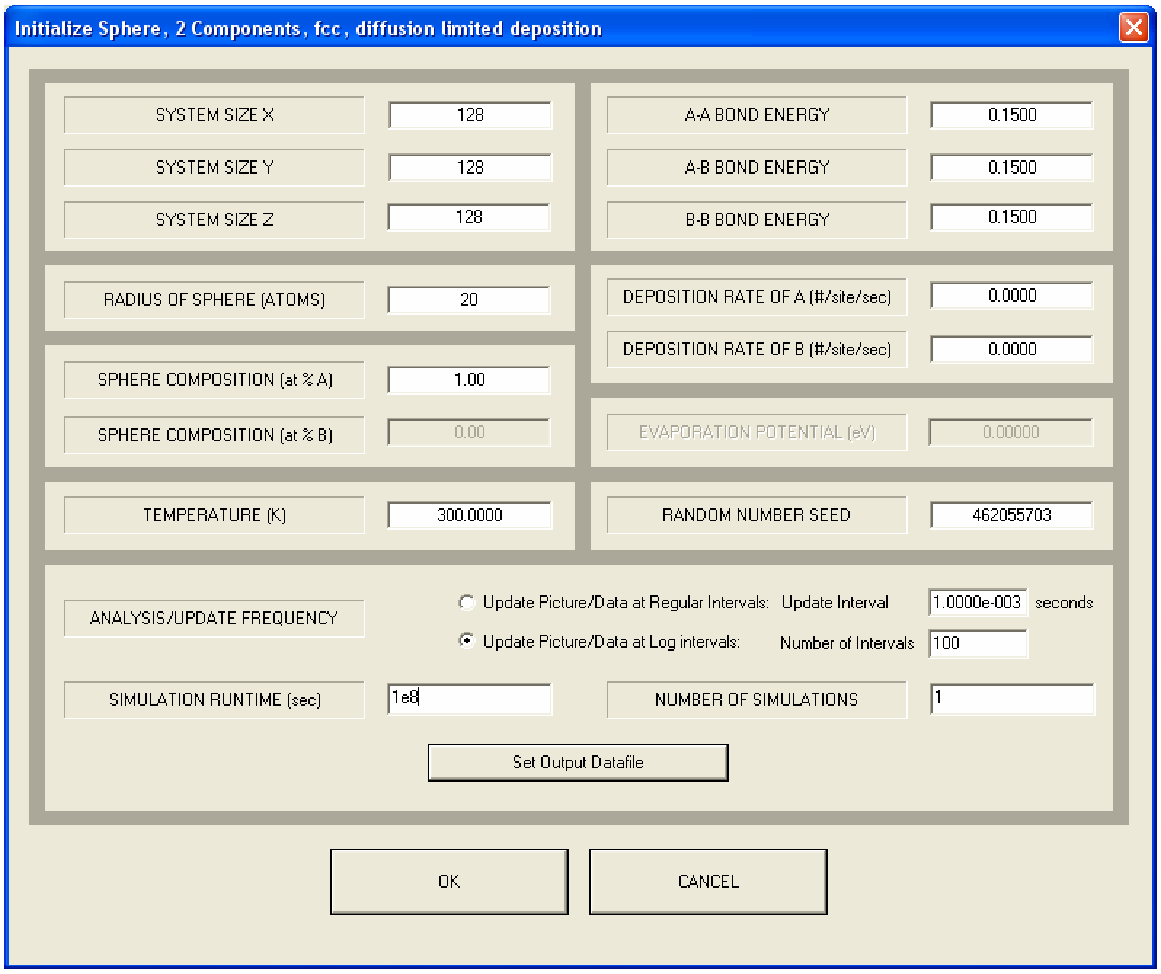




0.16 seconds 0.5 seconds 1.0 seconds

Note that at 1.0 seconds, it is not perfect filling; new islands have nucleated on top of the original layer.

Example 2. Equilibration of an fcc nanocrystal

Crystals are faceted, so if we start with a sphere of atoms, we should be able to watch faces form and sharpen. The equilibrium shape of an fcc crystal is a geometric object called a *truncated octahedron*, which is an octahedron with its pointy bits lopped off. Using the Spherical Cluster simulation, we can watch this entity form. Here are the initial conditions (note the use of the log time interval):



Initialize Sphere, 2 Components, fcc, diffusion limited deposition

SYSTEM SIZE X	128	A-A BOND ENERGY	0.1500
SYSTEM SIZE Y	128	A-B BOND ENERGY	0.1500
SYSTEM SIZE Z	128	B-B BOND ENERGY	0.1500
RADIUS OF SPHERE (ATOMS)	20	DEPOSITION RATE OF A (#/site/sec)	0.0000
SPHERE COMPOSITION (at % A)	1.00	DEPOSITION RATE OF B (#/site/sec)	0.0000
SPHERE COMPOSITION (at % B)	0.00	EVAPORATION POTENTIAL (eV)	0.00000
TEMPERATURE (K)	300.0000	RANDOM NUMBER SEED	462055703

ANALYSIS/UPDATE FREQUENCY

Update Picture/Data at Regular Intervals: Update Interval 1.0000e-003 seconds

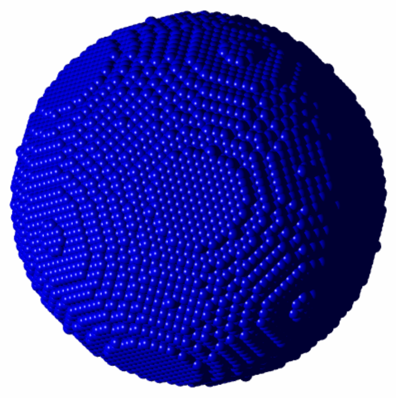
[✓ Update Picture/Data at Log intervals:] Number of Intervals 100

SIMULATION RUNTIME (sec) 1e8

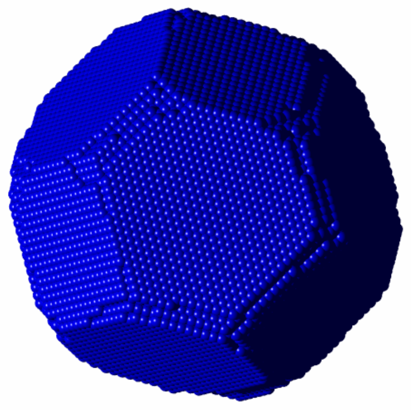
NUMBER OF SIMULATIONS 1

Don't panic – the initialization takes a while.

During the course of the simulation, the system evolves like this:



Initial condition



10^5 seconds