

HELSINKI UNIVERSITY OF TECHNOLOGY
Laboratory of Computational Engineering
August 2001

2D Boundary - a Molecular Dynamics Simulation Program
Version 1.1
Users Guide

Jari Rintala
jrintal@cc.hut.fi
48290N

Updated by Tapio Nieminen
tyniemini@lce.hut.fi

Contents

1	Preface	2
2	Quick Start	3
2.1	Overview	3
2.2	Compiling and Starting the Program	3
2.2.1	Command Line Options	3
3	Simulation Parameters	4
3.1	General Information	4
3.2	The Parameters Window	4
3.2.1	General Parameters	5
3.2.2	Lattice Parameters and Potential	5
3.2.3	Event File Selection	7
3.3	The Seaming Window	7
4	The Main Window “Boundary”	9
4.1	The Menubar	9
4.2	Information Screen	10
4.3	Display Controls	11
4.4	Main Simulation Area	11
4.5	Plotting Area	12
5	Auxiliary Windows	13
5.1	Animation	13
5.2	Pressure	13
5.3	Temperature	13
5.4	Modify Area	14
5.5	Zoom	14
5.6	Color Ranges	15
5.7	Track defects	15
6	Program Code	16
6.1	Program Files	16
6.2	A Little Bit of MOTIF	17
6.2.1	Basic Widgets	17
6.2.2	Manager Widgets	17
6.2.3	Customized Widgets	18
7	Appendix	19

1 Preface

BOUNDARY is an interactive simulation program for materials research in two dimensions. It has been developed in the laboratory of Computational Engineering in Helsinki University of Technology in Finland by J.Merimaa, L.F.Perondi, K.Kaski and further developed by A.Kuronen, J.Rintala, M.Robles and T.Nieminen. Even though all simulations are done in two dimensions, which is of course a serious limitation, BOUNDARY is also capable of doing materials research besides all the visualizing possibilities. An example of research could be a search for the critical layer thickness in lattice mismatched systems or research of interactions between dislocations in systems with initial cracks e.g.

This manual is about version 1.1 which is an improved version of the main version of the program (version 1.0). Improvements in this version are: better plot drawing, now also this version (previously only the side-version “-1.0”) enables periodic boundaries, bug fixes, “area modifier” -dialog, re-organized user-interface, lots of internal changes that make the code more readable etc.

It should be quite easy to learn to use the BOUNDARY program. Just compile it, run it and start playing with all the buttons and widgets and see what happens. In case the meaning of some button is unclear, hopefully this manual will help in understanding its purpose. I recommend especially reading the section 3.2.2 about the potential model and the widgets that change the potential model parameters. It should not take very long time to read (and understand) them but it helps you a lot to understand how BOUNDARY really works.

2 Quick Start

2.1 Overview

Starting the program opens two windows on the screen: The main window *Boundary* and the instrument panel window *Parameters*. In the middle of the main window is the simulation screen where all the action occurs.

The *Parameters* window contains several arrowbuttons, togglebuttons and pushbuttons. These are used for initializing all the parameter values before the simulation. When the system initialization has been done, the *Parameters* window can be closed by pressing the *OK* button and the simulation can be started by pressing the *Run* button. Simulation can be stopped/paused with the same button it was started and *Parameters* window can be opened again by selecting *New* from the *File* menu.

The main window contains also widgets on the left side of the simulation screen. All of these can be used during the simulation. In the left top corner of this window is an information screen, which shows data about the simulation. At first, the only information shown is how many atoms the system contains. Other information is activated when the *Parameters* window is closed by pushing the *OK* button.

2.2 Compiling and Starting the Program

To be able to run the program it must be compiled first by commands **xmkmf -a** and **make**. This creates the runnable file **BOUNDARY**, and now the program can be started by typing **boundary**. For details see *readme.txt*.

2.2.1 Command Line Options

The command line options that can be used (**boundary -option arg**):

- bg COLOR** set the background color to COLOR (e.g. red)
- fg COLOR** set the foreground color to COLOR (e.g. black)
- clean** remove all saved situation files (.xyz etc.)
- pe INTEGER** print total kinetic energy and potential to stdout at intervals of INTEGER timesteps
- readatoms X** read atom positions from a coords.X.xyz file where X is a 4-digit number, e.g. '0004' and parameters from boundary.X.param
- seed INTEGER** feed random number generator with INTEGER
- start** automatically start running the simulation

3 Simulation Parameters

3.1 General Information

The program has several ways to get simulation parameters at startup. The first method is internal hard-coded variable default values. These are set so that the result is something rational. The second method is reading the parameters from a file `BOUNDARY.PARAM`. This file is a normal text file which can be edited by any text editor and this is recommended method to permanently change some default parameter, because these parameters overrides the hard-coded values. The 3rd method, which overrides the previous ones, is command line options but this only affects very few parameters. The 4th and last method (overriding the previous ones) is interactively changing the values of the widgets especially in the *Parameters* window but also in all other windows.

Pressing the *OK* button in the *Parameters* window causes `BOUNDARY` to calculate other parameters according to these main parameters and to initialize the simulation. Command line option `-readatoms X` causes `BOUNDARY` to skip the whole parameters window and initialize the simulation according to atom coordinates given in the file `coords.X.xyz` and parameters in `boundary.X.param`.

In the next few subsections the main parameters in the *Parameters* and *Seaming* windows are explained separately. These parameters define the system configuration i.e. atom positions and types. The parameters in the other windows do not usually directly affect configuration (indirectly several of them do affect configuration). One exception is the *Modify area* (see section 5.4) which directly affects configuration and it is most often used after closing *Parameters* before pressing *Run*.

3.2 The Parameters Window

The *Parameters* window is the smaller window of the two windows that open automatically when the program is started. This window contains three columns with widgets used for initializing the system before simulation.

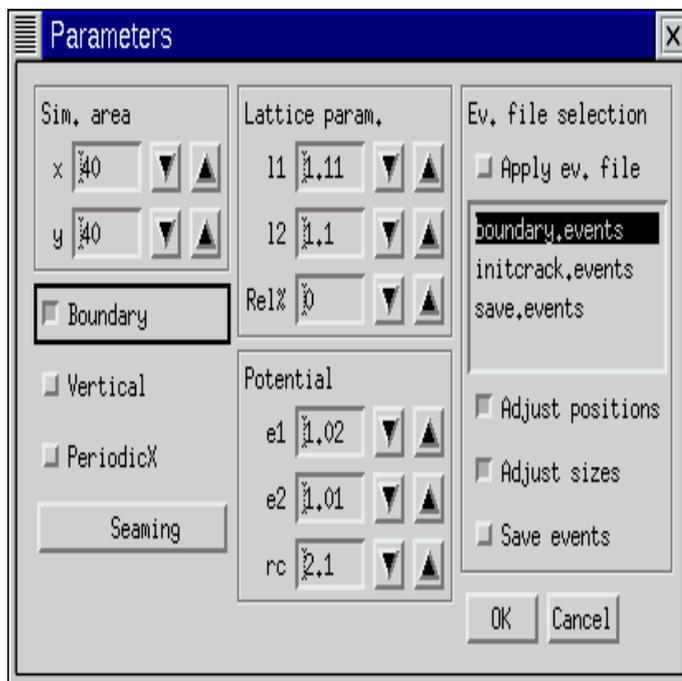


Figure 1: The Parameters Window

The widgets in the first column determine the size of the system (the amount of the atoms) and some general properties of the system. The parameters in the middle column determine how the

atoms interact with each other. The widgets in the right column select .event file options. The button *Seaming* in the first column opens a new dialog, whose widgets control the form of the interface (“the seam”) and the lattices.

3.2.1 General Parameters

Sim. area

- *X*: Increases the amount of atoms in *x* direction. Enlarges the simulation area in *x* direction.
- *Y*: Increases the amount of atoms in *y* direction. Enlarges the simulation area in *y* direction.

The change in the amount of atoms in the system can be seen in the information screen in the main window. Changing the size of the system changes also the zooming in the *simulation* screen. The zooming can be changed by using the text arrow buttons in the *View size* frame, which is found in the right top corner of the main window.

General Toggles

- *Boundary*: If this toggle button is on, the interface exists, if the button is off all the atoms are of type one.
- *Vertical*: If the toggle button is off, the interface is horizontal and if the button is on, the interface is vertical.
- *PeriodicX*: The system wraps horizontally, so that the atoms on the right side are neighbours of the atoms on the left side. **NOTE: PeriodicX doesn't work on all values of Sim. area x.**

Seaming Button opens the *Seaming* window. See section 3.3 on page 7.

3.2.2 Lattice Parameters and Potential

To be able to explain what these five text arrow widgets in Lattice param and potential frame do it is necessary to take a look at the potential model Boundary uses. Inter-atomic interactions in Boundary have been modelled by the Lennard-Jones pair potential:

$$U(r) = \epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right). \quad (1)$$

In (1) r is the distance between interacting atoms and σ and ϵ are Lennard-Jones potential model parameters. In order to speed up the computations the interaction range has been limited by setting the potential to zero beyond a maximum inter-atomic distance $r = r_c$. To avoid an anomalous behaviour of the force a linear term $(r - r_c) \frac{dU}{dr} |_{r=r_c}$ has been subtracted from U . So the interaction equations come to form:

Potential energy between two atoms in the distance r is calculated from equation

$$U(r) = V\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^6 \right) - V\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{cij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{cij}} \right)^6 \right) + 12V\epsilon_{ij} \frac{(r_{cij} - r)}{r_{cij}} \left(\frac{1}{2} \left(\frac{\sigma_{ij}}{r_{cij}} \right)^6 - \left(\frac{\sigma_{ij}}{r_{cij}} \right)^{12} \right) \quad (2)$$

, where r_{cij} is the cut off distance between two atoms of type i and j . Force acting on an atom is the negative gradient of the potential:

$$\vec{F} = -\frac{d(U(r))}{dr} \cdot \frac{\vec{r}}{|\vec{r}|} = 12V\frac{\epsilon_{ij}}{r} \left(\left(\frac{\sigma_{ij}}{r} \right)^{12} - \frac{1}{2} \left(\frac{\sigma_{ij}}{r} \right)^6 \right) - 12V\frac{\epsilon_{ij}}{r_{cij}} \left(\frac{1}{2} \left(\frac{\sigma_{ij}}{r_{cij}} \right)^6 - \left(\frac{\sigma_{ij}}{r_{cij}} \right)^{12} \right) \cdot \frac{\vec{r}}{|\vec{r}|} \quad (3)$$

In BOUNDARY it is possible to have two different kinds of atom types in the system. In equations (2) and (3) the σ_{ij} and ϵ_{ij} are Lennard-Jones potential model parameters, where i and j are the atom types of two interacting atoms. These σ s and ϵ s are calculated from equations

$$\sigma_{11} = \left(\frac{1}{2} \frac{\left(\frac{1}{r_c}\right)^7 - 1}{\left(\frac{1}{r_c}\right)^{13} - 1} \right)^{\frac{1}{6}} \frac{lparam1}{\sqrt{2}} \quad (4)$$

$$\sigma_{22} = \left(\frac{1}{2} \frac{\left(\frac{1}{r_c}\right)^7 - 1}{\left(\frac{1}{r_c}\right)^{13} - 1} \right)^{\frac{1}{6}} \frac{lparam2}{\sqrt{2}} \quad (5)$$

$$\sigma_{12} = \sigma_{21} = \left(\frac{\sigma_{11}^6 + \sigma_{22}^6}{2} \right)^{\frac{1}{6}} \quad (6)$$

$$\epsilon_{12} = \epsilon_{21} = 2\sqrt{\epsilon_{11}\epsilon_{22}} \frac{\sigma_{11}^3 \sigma_{22}^3}{\sigma_{11}^6 + \sigma_{22}^6} \quad (7)$$

And the interaction distance variables r_{cij} are calculated from equations

$$r_{c11} = \frac{cut * lparam1}{\sqrt{2}} \quad (8)$$

$$r_{c22} = \frac{cut * lparam2}{\sqrt{2}} \quad (9)$$

$$r_{c12} = r_{c21} = \left(\frac{(r_{c11})^6 + (r_{c22})^6}{2} \right)^{\frac{1}{6}}, \quad (10)$$

where *cut* is a parameter value that can be changed by using the *rc* text arrow widget in Potential frame.

Lattice param. And Potential Widgets

In frame Lattice param:

- *l1* value is the value of *lparam1* in the equations (4) and (8). It is the lattice parameter of the atom type one (blue atoms).
- *l2* is the *lparam2* value in the equations (5) and (9). It is the lattice parameter of the atom type two (red atoms).
- *Rel%* describes how relaxed the lattice two is. If this widget value is zero the atoms of type two are forced to be in the lattice of atom type one. If the lattice mismatch is large this means that a large stress is build initially into the system. If the widget value is 100 the atoms of type one are in their own lattice. This means less stress in the system but also existence of dislocations at the interface. Otherwise the lattice of atoms of type two is something between these two cases.

In the frame Potential:

- *e1* is the value of ϵ_{11} in equations (2),(3) and (7).
- *e2* is the value of ϵ_{22} in equations (2),(3) and (7).
- *rc* is the value of *cut* in equations (8) and (9).

The properties of an atom type in Boundary are dependent on two parameter values: The lattice parameter value and the ϵ value. This means that if *lparam1* equals to *lparam2* and ϵ_{11} equals to ϵ_{22} the two atom types are actually the same. When this happens the colour of the atom type two changes to the colour of the atom type one.

3.2.3 Event File Selection

A `.events` file is a file that contains instructions when to execute certain events. It can be used for automatically executing “temperature” or “modify area” -events (see section 5.4 on page 14). This means that `.events` provides a powerful way to automatically configure the system or apply stress etc. In previous versions of `BOUNDARY` there were separate controls for fixing borders, applying stress etc but now they are all in *Modify Area*, which can be used for initial/automatic configuration via `tscheme`.

- *Apply ev.file* uses the selected `tscheme` for this run.
- Selection box shows available `tscheme` files.
- *Adjust position* repositions `.events` modify area -events so that their positions fit the selected sim. area size. Otherwise they are in fixed positions, which may be desirable in some situations.
- *Adjust size* resizes `.events` modify area -events so that their positions fit the selected sim. area size. Otherwise they are in fixed positions, which may be desirable in some situations.
- *Save events* saves modify area -events in the simulation run (that begins after pressing *OK*) to `save.events`. This way the user can ease the task of producing new event files. Temperature events have to be added manually with some text editor and modify events can also be added this way. The syntax of an `.events` file is very simple. When a good `save.events` has been recorded, it should be copied or renamed to another `.events` file so that the program doesn't overwrite it.

OK and Cancel

OK closes the *parameters* window and allows the simulation to be started. If the *Parameters* window has been opened by selecting *New* from the *File* menu, *OK* initializes new simulation. Closing the window from a window frame control (like “X”) is equal to pressing *OK*.

Cancel can be used when the *Parameters* window is opened from the menu. *Cancel* closes the *Parameters* window and keeps the situation as it was before opening the window.

3.3 The Seaming Window

To avoid a total mess some of the not so often used widgets in the *Parameters* window are hidden behind the “Seaming” button. The controls in this window generally change the atom configuration by moving/removing/adding atoms.

Caution! If two or more atoms are moved too close to each other the program will crash when the simulation is started.

Fixing Suggestion

When this button is on (and the vertical button is off) the program returns a fixing suggestion number to the window, from which the program was originally started, every time the system is changed during the initialization. The fixing suggestion number tells how much farther or closer the red lattice and blue lattice should be to each other so that the distance in *y* -direction between the uppermost blue atom layer and the lowermost red atom layer would be the same as the distance in *y* -direction is between any two nearest blue atom layers. This can be needed when the displacement of the boundary must be changed when atoms of type two are in their own lattice. This is because if $lparam1 > lparam2$, then the blue layer needs more space than the red one. Now when one red layer is removed and one blue layer is added to the system, the lowermost red atom layer and the uppermost blue atom layer get closer each other in the interface area. This can be tried by setting the relaxation value to 100 % and using the `Bndr-displacement` -widget.

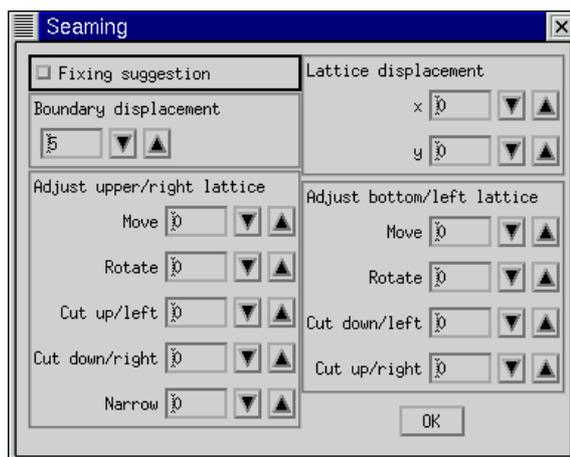


Figure 2: The Seaming Window

Bndr-displacement

If the boundary is horizontal the *Bndr-displacement* widget moves the interface in y -direction. In vertical case the interface is moved in x -direction.

Adjust Upper/Right Lattice

These widgets change the upper lattice or if the *Vertical* toggle is on, the right lattice. These widgets do nothing if the *Boundary* toggle is off.

- *Move* This widget moves the upper lattice in y -direction when the vertical toggle button is off, and the right lattice in x -direction when the vertical toggle button is on. The lower/left lattice doesn't move.
- *Rotate* Right/upper part of the system can be rotated around the z -axis by using this widget.
- *Cut up/right*: Removes the top (left if *Vertical*) atom layer from the upper (right if *Vertical* lattice.
- *Cut down/left*: Removes the bottom (right if *Vertical*) atom layer from the upper (left if *Vertical* lattice.
- *Narrow*: Removes atoms from the upper lattice from both sides. This “quantum dot” can now be moved in x -direction by using the x widget.

Lattice Displacement

- x : If the boundary is horizontal and narrow value is zero this widget moves the upper lattice in x -direction. If boundary is vertical and narrow value is zero both lattices move opposite directions along the x -axis. If narrow is not zero the upper lattice can be moved in x -direction by using this widget.
- y : If the boundary is horizontal using this widget makes both lattices to move opposite directions along the y -axis. If the boundary is vertical, widget y doesn't work properly.

Adjust bottom/left lattice

These widgets change the bottom lattice or if the *Vertical* toggle is on, the left lattice. If the *Boundary* toggle is off, these widgets change the whole system.

- *Move* This widget moves the bottom lattice in y -direction when the vertical toggle button is off, and the left lattice in x -direction when the *Vertical* toggle button is on. The upper/right lattice doesn't move.
- *Rotate* Left/bottom part of the system can be rotated around the z -axis by using this widget.
- *Cut up/right*: Removes the top (right if *Vertical*) atom layer from the lower (left if *Vertical* lattice.
- *Cut down/left*: Removes the bottom (left if *Vertical*) atom layer from the bottom (left if *Vertical* lattice.

OK closes the *Seaming* window.

4 The Main Window “Boundary”

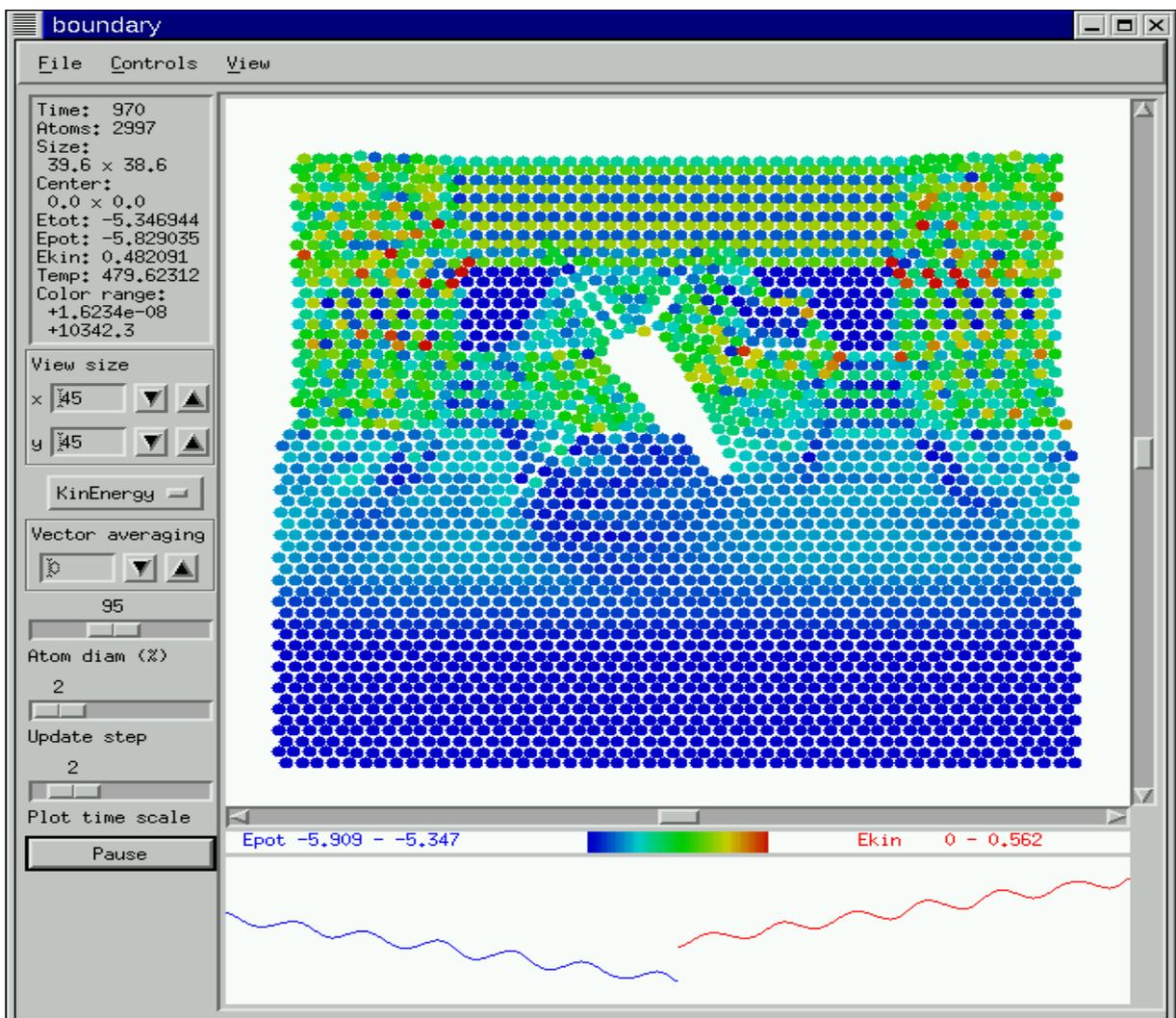


Figure 3: The Main Window

4.1 The Menubar

File

- *New* opens the parameters window.

- *Save snapshot* writes the current atom positions to a file. The file is .xyz or .pdb formatted (e.g. boundary.0001.xyz). This also writes the current parameters to a file (e.g. boundary.0001.param). For example RasMol can read the potential energies from a .pdb file. XYZ file contains only atom positions and types.
- *Save animation* takes multiple snapshots at regular intervals. The settings can be changed from a popup window which opens when the user selects *Save animation* from the menu. See section 5.1.
- *Quit* quits BOUNDARY.

Controls

- *Pressure* opens a window where user can apply pressure to the system. See section 5.2.
- *Temperature* opens the temperature controls window. See section 5.3.
- *Modify area* opens the *Modify area* window. It can be used for modifying arbitrary rectangle areas in the system before starting the simulation or during the simulation. See section 5.4 on page 14.

View

- *Rescale plot* rescales the plot vertically to minimum range that shows all the values in current the plot history.
- *Manual zoom* opens the *Manual zoom* window. The widgets in that window change the zoom settings. See section 5.5.
- *Color ranges* opens the *Color ranges* window. See section 5.6.
- *Track defects* opens a window which show defects in the system. See section 5.7.

4.2 Information Screen

- *Time*: Shows how long time the simulation has runned in units of time $t_0 = 10^{-15}s$.
- *Atoms*: Shows the amount of atoms in the system.
- *Size*: (Maximum x coordinate)-(Minimum x coordinate) * (Maximum y coordinate)-(Minimum y coordinate). Maximum x coordinate means the x coordinate of the atom nearest to the right edge of the simulation area. Minimum x coordinate correspondingly is the x coordinate of the atom nearest to the left edge of the simulation area. Simulation area is the box surrounded buy a blue line which can be seen when the view in the simulation screen is zoomed out enough. If atoms go over these lines they are discarded.
- *Center*: Coordinates of the center of the view. Changes when a scrollbar of the simulation screen is moved.
- *Etot*: Average total energy of an atom in the system in energy units.($1EnergyUnit = 0.0857eV$)
- *Epot*: Average potential energy of an atom in the system in energy units.
- *Ekin*: Average kinetic energy of an atom in the system in energy units.
- *Temp*: The system temperature in Kelvins.
- *Color range*: Shows the minimum and maximum values of energy (dependent on the current display menu selection, see) that can be properly visualized.

4.3 Display Controls

Zoom changes the zoom setting. Zoom can also be changed by moving the mouse vertically in the drawing window with ctrl and right mouse button pressed. x - and y -directional zoom levels can also be changed separately and in a larger range from the *View* menu item *Manual zoom*.

Display menu changes atom color-coding:

- *KinEnergy*: Atoms are color-coded according to their kinetic energy
- *PotEnergy*: Atoms are color-coded according to their potential energy
- *StressXX*: Atoms are color-coded according to horizontal stress
- *StressXY*: Atoms are color-coded according to shear stress
- *StressYY*: Atoms are color-coded according to vertical stress
- *Atom type*: Atoms are color-coded according to their type
- *Force*: Atoms are color-coded according to their acceleration which equals to the sum of forces affecting the atom.
- *Vectors*: Doesn't show atoms but velocity vectors

Vector averaging works only when *Vectors* is selected in the *display menu*. The simulation screen is divided into several averaging cells. When *Vectors* is selected from the menu the program shows the average velocity vector of each cell. When *Averaging* value increases also the the cells get bigger. This means that one vector shows the average velocity of greater amount of atoms and the amount of vectors on the screen decreases.

Atom diam % changes the diameter of the circles that are representing atoms in the simulation screen.

Update step determines how often the program draws new picture of the atoms on the screen, during the simulation.

Plot time diam % determines how long potential/kinetic energy history graph is shown. Larger values mean longer history but lower visible accuracy. The internal accuracy and history lenght are always the same. Also the value of *Update step* affects accuracy, this time also the internal accuracy . Longer Update step causes lower accuracy.

Run/Pause starts or pauses the simulation.

4.4 Main Simulation Area

This area contains the picture of the system, either atoms or velocity vectors. You can click ctrl+shift+right mouse button on an atom to print info about it to stdout. The area can be scrolled using the scrollbars or by dragging the mouse in the area with right mouse button pressed.

4.5 Plotting Area

This area shows a history graph for system's total potential and kinetic energies. The left one is for potential energy and the right one for kinetic energy. The plotted ranges (vertical,energy) are shown above. The horizontal range (time) can be changed from the scale bar on the left. The plot automatically scales so that all values can be seen. However, if the system for example cools down so that both potential and kinetic energy are only a fraction of what they where before, the plot might become uninformative. To scale the plot back so that most of the plotting space vertically is used, select *Rescale plot* from the *View* menu.

5 Auxiliary Windows

5.1 Animation

Animation is a program that takes snapshots (see section 4.1) at regular intervals.

- *Set animation*: If Set animation toggle button is on the program takes a snapshot every time the *Wait* time has passed since the last snapshot.
- *Wait*: Interval between two snapshots.

If *Set animation* toggle button is on, the program goes on taking snapshots until the simulation ends. When the *Limits on* toggle button is on, the program will take only as many snapshots as showed in the Snapshots window.

- *Begin at*: Program starts taking snapshots after the simulation time reaches this value.
- *End at*: Program ends taking snapshot after the simulation time reaches this value.
- *End prog. aft end at*: If this toggle button is on the program ends after all the snapshots are taken.
- *Close*: Closes the Animation window.

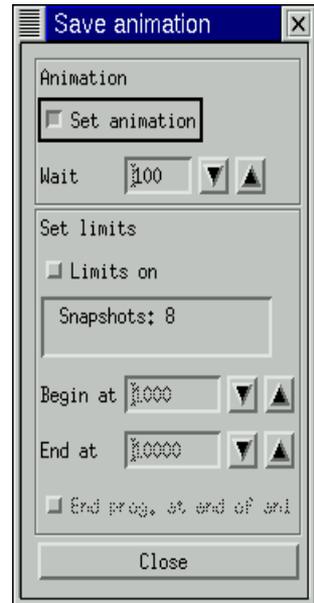


Figure 4. Save animation

5.2 Pressure

Pressure applies an y -directional compression into the system. The unit of pressure is $\frac{kg}{s^2}$.

5.3 Temperature

Temperature

- *Low temp* toggle button scales the temperature of the system to lowtemp value. See *Temp*. The arrow buttons on the right side of the *Low Temp* button cools down/warms up the system while an arrow is pushed. (You do not have to click these several times, just keep on pushing until the temperature is right).
- *Temp* changes the lowtemp value.

Cooling Parameters

Boundary's cooling program makes a simulated cooling in the system according to the parameters in the Cooling Parameters frame. The simulation starts when the *RUN* button is pushed. At first the temperature of the system is held at an initial temperature until a startCooling time is reached. After that the temperature is decreased linearly to zero and when zero temperature is reached the simulation stops.

- *Init Temp* changes the value of the initial temperature.
- *Start at* changes the value of the startCooling time.
- *slope* of the linear function $temp = Init.temp - slope(time - startCooling)$ that determines how fast the cooling to zero is done.
- *Start cooling* starts cooling the system according to previous parameters.
- *Close* closes the Cooling Parameters window.

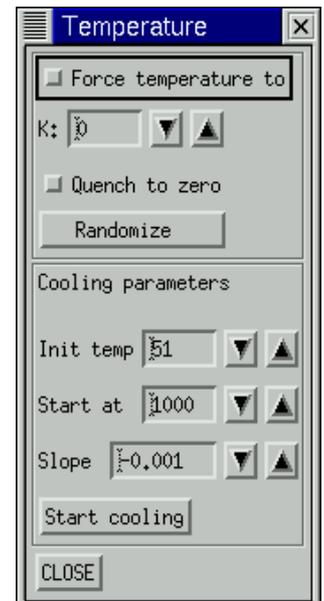


Figure 5. Temperature

5.4 Modify Area

The Area parameters in this window change the selection area. Selecting an area doesn't do anything to it. (Actually this whole window has only one control that actually does something to the atoms: *Modify* button.) The position, size and orientation of the selection can be seen in the main window as a black rectangle. If *Show selection* is selected, the atoms inside the selection are drawn in grey.

There is also an alternative, mouse-controlled way to change the selection. This is much faster but it might require some practicing. Clicking the left mouse button in the main window moves the center of the selection there. Dragging the mouse with left button pressed moves the selection. Pressing control and moving the mouse with left button pressed resizes the selection vertically and horizontally. Pressing shift and moving the mouse vertically with left button pressed rotates the selection. Finally, pressing ctrl, shift and left mouse button allows the user to move and resize the selection in sequence and sets the rotation to zero. This last method is the fastest method to select an area. After this it can be rotated with shift+left button.

NOTE: These mouse selection methods can produce "strange" side-effects if the user doesn't release the left mouse button between different actions!

The upper part of the window controls modify action. From the menu the user can choose several actions:

- *Remove area* removes the selected atoms.
- *Stop atoms* sets the velocity of the selected atoms to zero.
- *Change to type 1* all selected atoms are changed to type 1 (blue).
- *Change to type 2* all selected atoms are changed to type 2 (red).
- *Add velocity* adds velocity to the selected atoms. The amount of velocity and its direction is defined in the below widgets.
- *Fix velocity* permanently sets the velocity of the selected atoms to the value of control widgets below. After this these atoms are not affected by forces of neighbour atoms, collisions or anything. They just continue on the same speed. This can be used for applying strain to the system, although this is not actually a force but it produces pretty good results. If the velocity is fixed to zero, these atoms can be used to form fixed borders etc.
- *Add const. force* adds a constant force to the selected atoms. The force equals to acceleration because the mass of an atom is 1.000. The atoms are affected by the sum of this constant force and the forces from nearby atoms. This is more realistic than the previous fixing velocity but also more difficult to use.
- *Remove fix./force* removes any fixed velocity or constant force that the selected atoms might have. After this, they are only normal atoms. However, if the atoms were accelerated by a force, this doesn't remove the speed gained by this acceleration.

X and *Y* are used for changing the velocity and force.

Clicking the *Modify* button performs the selected action on the selected area with the parameters *X* and *Y*. **Warning:** this change is permanent and the only way to get back atoms removed by this tool is to start a new simulation!

5.5 Zoom

- *x*: Changes the zooming in *x*-direction in the simulation screen.
- *y*: Changes the zooming in *y*-direction in the simulation screen.

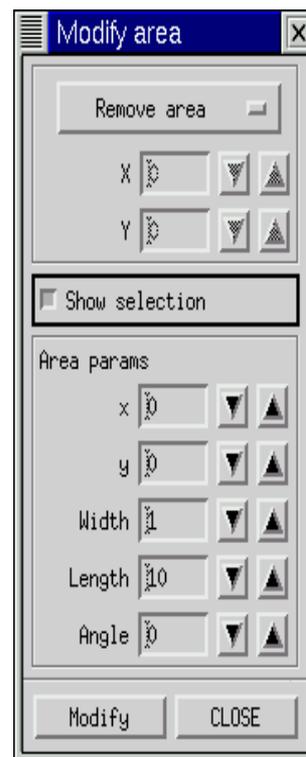


Figure 6. Modify area

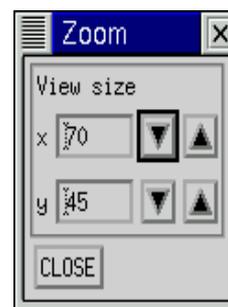


Figure 7. Zoom

5.6 Color Ranges

The *Color ranges* window sets the color ranges to atom color codings in the main view.

If *Auto-calc ranges* toggle is on, the color ranges are calculated automatically so that all values are in the range. If this toggle is off, ranges from the widgets below are used.

If *Hide atoms outside color ranges* toggle is on, atoms that are outside color range are not shown at all. If the toggle is off, atoms that are outside the current color range are shown in nearest color (blue or red).

Color range parameters:

- *EkMin*: minimum kinetic energy shown, drawcolor blue.
- *EkMax*: maximum kinetic energy shown, drawcolor red.
- *EpMin*: minimum potential energy shown, drawcolor blue.
- *EpMax*: maximum potential energy shown, drawcolor red.
- *SMin*: minimum stress shown, drawcolor blue.
- *SMax*: maximum stress shown, drawcolor red.
- *FMin*: minimum force/acceleration shown, drawcolor blue.
- *FMax*: maximum force/acc. shown, drawcolor red.

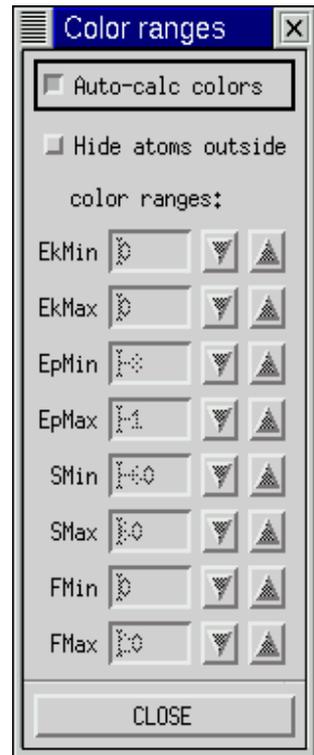


Figure 8. Color ranges

5.7 Track defects

This tracks clusters in the system. Clusters are areas, where the atoms are near each other.

Warning: the tracking consumes a lot of cpu time and can slow down the simulation considerably. If the simulation seems to get stuck with tracking, either quit or press “Track it!” toggle off (only one press) and wait one minute (more or less dependig on the cpu and the number of atoms).

- *Track it!*: Starts tracking
- *Range*: Defines how near b atom a has to be so that it belongs to the same cluster with atom b.
- *SFactor*: Defines a minimum kinetic energy that an atom has have so that it can belong to a cluster. $maxkinE - (maxkinE - minkinE) * SFactor = clusterminkinE$
- *BorderF*: Width of rectangle outer borders that is left out of cluster calculations.
- *Save positions*: saves defects to a file (e.g. TrkFile0001.dat)
- *Show positions*: shows the positions on screen

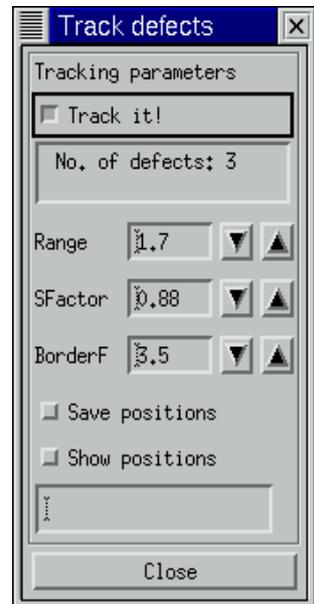


Figure 9. Tracking

6 Program Code

Boundary program has been written by using the C programming language. The graphical parts work in the X windows environment and rely on the MOTIF library.

6.1 Program Files

The files:

- `calc.c` and `calc.h` initial atom position calculations
- `callback.c` and `callback.h` widget callbacks
- `control.c` and `control.h` simulation setup
- `fitsigma.c` and `fitsigma.h` fitsigma and energy functions
- `graphics.c`, `graphics.c` and `graphvar.h` widgets and drawing
- `simu.c` and `simu.h` force computations etc
- `track.c` and `track.h` tracking
- `gvar.h` global variables
- `boundary.param` parameters that can be modified
- `boundary.tscheme` example tscheme
- `readme.txt` installation instructions
- `manual.ps` this manual

Calc.c contains three functions for calculating the initial positions for atoms. `CalcRect()` calculates initial positions for a lattice containing only one kinds of atoms. `FormBoundary` basically calls `CalcRect` twice and makes the interface by uniting these two lattice parts. As you can see from chapter 2.4, there are several possibilities for cutting atom layers from the bottom or from the top of the lattices and for moving the lattices e.g. This all is done here. The third function `CreateInitialCrack` removes atoms from the crack area.

Callback.c defines what happens when a push- or toggle button is pushed or some other widget is used.

Control.c contains the `main()` function and all functions which handle command line options, read parameters from files or write files etc. `Control.h` contains important macro definitions that are used in several `.c` files.

Gvar.h declares global variables that are needed by several files. In general, the most important variables of the program are in `gvar.h`. `Gvar.h` is included in almost all `.c` files.

Fitsigma.c contains functions that calculate fitsigma calculations: they adjust simulation sigma parameters so that the relaxation is zero at least in lattice 1.

Graphics.c contains functions that draw the atoms, create widgets, open windows and window and in general every function that draws something, be it widget or point. `Graphvar.h` contains global variables that are needed only in the `graphics.c`, for example the contents of the menus.

Simu.c has all the functions for calculating forces and energies for atoms and finding new positions for them. `InitRun()` initializes the potential parameter values before the first simulation step. Then simulation goes around the function `MdCycle` until something happens. (The user pushes some button, animation ends, all the atoms have disappeared from the simulation screen...) Function `ComputeForces()` is the “holiest” part of the program. Over 95% of all cpu time used by `Boundary` is used inside the loops of `ComputeForces`. So changing the program here should be done with an extra care. Few lines of bad code inside these loops could slow down the program dramatically. New places and velocities for the atoms are calculated in `LeapfrogIntegrate()`. And new kinetic and potential energy values are calculated in `CalcEnergy()`.

Track.c contains the tracking algorithm functions. This code is not very well commented, optimized and tested. At least it is very cpu-intensive. It’s a bit experimental yet.

6.2 A Little Bit of MOTIF

In this chapter I’m going to give a rough picture how one can add some simple widgets like pushbuttons and dialog windows into the program by using copy paste technique. I will not give an exact explanation on how the Widgets in motif are build on top of other widgets or what does every detail in the Motif code means, these things can be read from any Motif manual [2]. Creating new widgets by copying existing code is in most cases perfectly enough. After reading this chapter you should be able to manage create these already existing widgets without reading hundreds of pages of MOTIF Manuals.

6.2.1 Basic Widgets

This chapter tells how to easily add widgets that are similar to the existing ones. **The general rule is: see how the original widgets are made and mercilessly copy and modify them.**

- First you have to find the place and type for the new widget. Let’s assume you want a pushbutton to the main window below the run button. If you want to know what widget something in the program is, you can just search the code for its label or something and eventually you will probably find it and its type.
- Secondly you need to create that widget. **Graphics.c** contains lots of functions that create simple widgets. For a push button, you must use `CreatePushB(..)`-function. Depending on the case you can make an entirely new global widget or local widget or “re-create” and old one without over-writing the original. To make a new one, you must add “**Widget w;**” to the start of the current function or to **gvar.h** if you want it to be a global variable which is seldom necessary. When the widget is declared, you just add a function call like “**w = CreatePushB(parent, label, callback)**”. To get the widget’s parent, see the widget above. The widget’s label is up to you.
- Callback is needed in buttons etc that do something. Look in **control.h** for existing callback numbers and add 1 to the last number of the proper widget type, give a macro-name to the number and add it to the list, e.g. **#define IDIOTBUTTON 101**.
- The last thing you have to make is to make the callback work. All you have to do is to add one case to the proper function (in this case `CbButton`) in **callback.c**.
- Compile and run.

6.2.2 Manager Widgets

This sub-chapter tells how to make new windows or how to get widgets in a proper order or what to do if the widgets don’t show up correctly.

- To make a new window, you can just look how the current windows and window are made. At least you need a new function to graphics.c (for example void MyWindow() ...). To get the window work, you can copy one of the existing windows and then modify it. One window requires a pretty long code fragment but you can copy-paste it and then edit it. When this is done, add a function call to your new window in a proper place.
- How to arrange widgets? Currently the widgets are not very heavily arranged and the current program structure requires a lot of work if you want to make something special. If you just want to make something simple, copy-paste. The basic ideas for copy-paste are: rowcols arrange widgets, forms and panes contain rowcols, windows contain forms and panes. The other types are pretty easy but the rowcol is a complex one. There are currently several rowcol-building functions for different needs. One of these should work just fine...
- If a widget doesn't appear on screen or appears only partially, the most common fault is a wrong or no manager widget (usually rowcol). Text- arrow widgets require a horizontal rowcol (CreateHRowCol(...)) and most other widgets need a vertical rowcol in this program, because the basic layout is vertical. So make sure that the widget's parent is a vertical rowcol if the widget is not a textarrow. Also, there are special rowcols for a window main rowcol and a rowcol to arrange widgets in columns as in the "Parameters" window.

6.2.3 Customized Widgets

To create a widget of a type not already in the program or a special version of an already existing widget requires placing motif commands directly to the code. Currently at least the scrollbars and drawareas are implemented this way. If you for example want to create a special pushbutton, you can copy-paste the pushbutton motif code from function CreatePushB and modify it to fulfill your needs. The other steps you have to do are similar to the ones in chapter 12.1.

7 Appendix

This part contains three figures showing views from the main window during a simulation. In the first figure the velocity vectors of each atom are showed in a system where a small block containing atoms of type two is placed on top of a substrate containing atoms of type one. This kind of block can easily be built into the system by using the widgets in the *Fit the Boundary* window (See chap 4). In the second figure the potential energy of each atom is showed in a system containing an initial crack. To be able to create systems like this see 3.12-3.15. In the third figure the system in the 2nd picture is compressed in the y direction. This is done by using the pressure widgets in the *Strain etc* window (see 8.2).

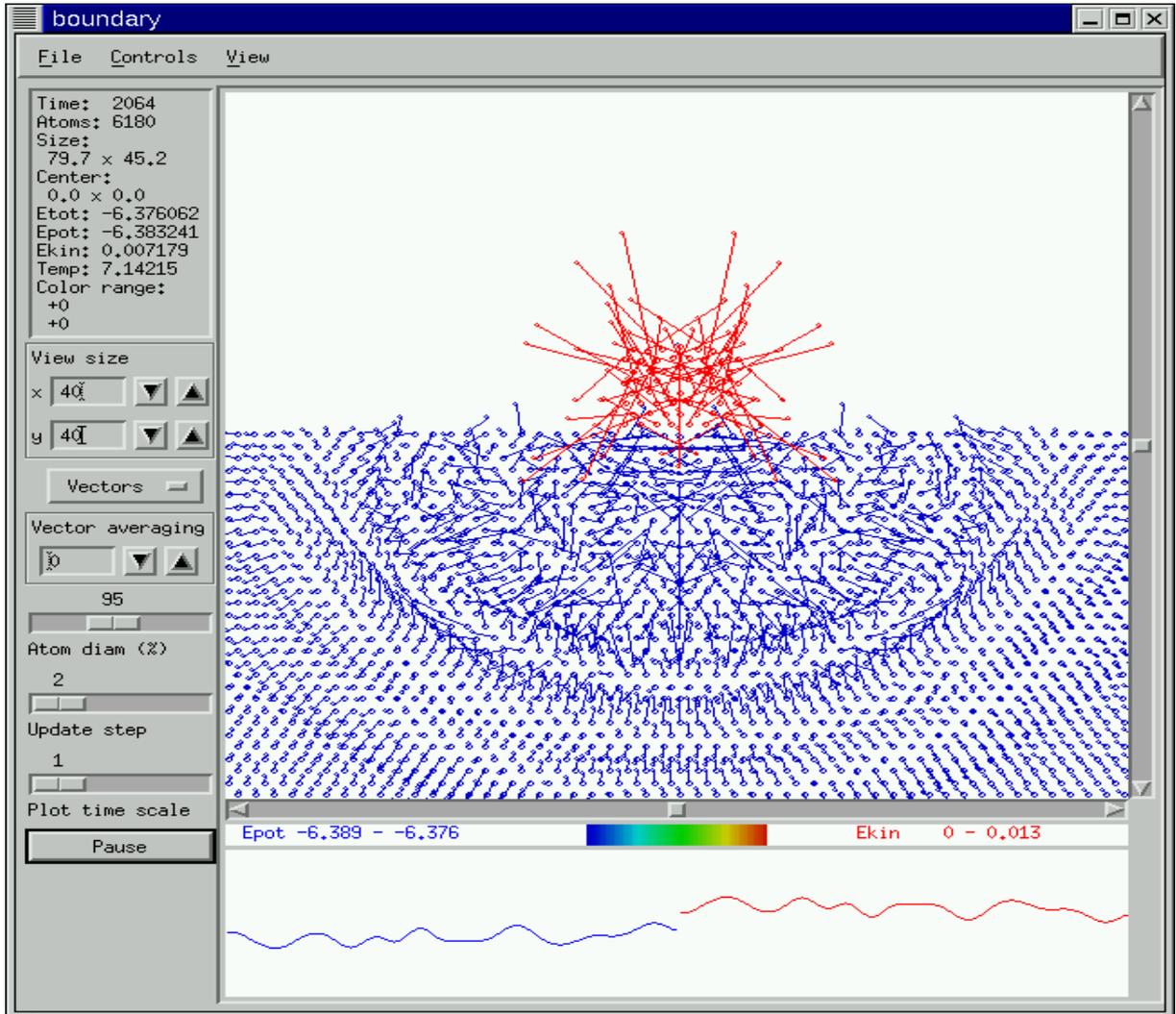


Figure 10: The velocity vectors of each atom showed during a simulation in a system that contains a “quantum dot”.

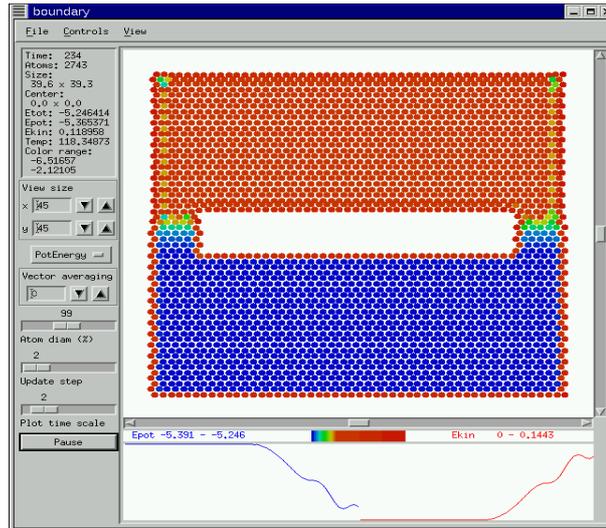


Figure 11: The potential energy of each atom showed during a simulation in a system that contains a large crack and two different materials.

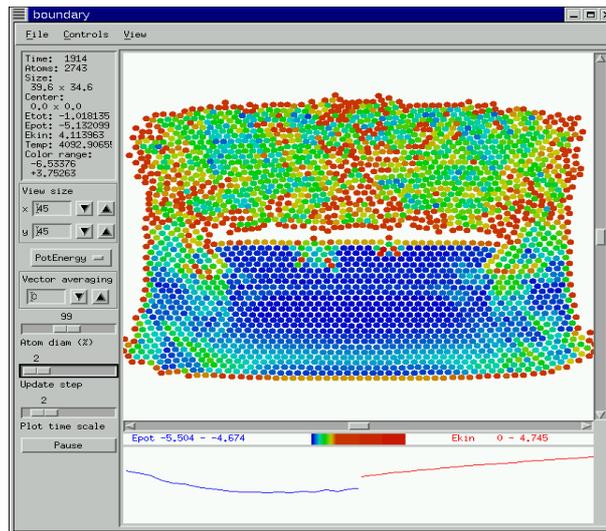


Figure 12: Heavy pressure compressing the same system.

References

- [1] J.Merimaa, L.F.Perondi and K.Kaski,
An interactive simulation program for visualizing complex phenomena in solids,
Comp. Phys. Comm. **124** (2000) 60.
- [2] http://www.cm.cf.ac.uk/Dave/X_lecture/X_book_caller/index.html