ASH 0.35
very short introduction

Teemu Hynninen

1 Introduction

What is this?

This is a minimal introduction to Atomic Structure Handler, i.e., ASH, version 0.35. ASH is a program written in Java designed for simultaneous construction and visualization of atomic structures, aimed at people doing atomistic simulations. ASH is not meant to provide a full featured simulation environment, such as for example the ASE package [1], but to be a quick and easy tool for building structures.

ASH has a built in manual which explains how various commands and options work and what kind of syntax they require. Therefore, the purpose of this document is to help you, the user, get the program running, teach how the logic of the program works and show where the built in help functions are.

The very very short version of this guide is:

- launch by `java -jar ash.jar`
- type `list command` and `man` followed by a command name to access the manual
- try it out, there is an `undo` if it doesn't work like you thought

The code is provided as-it-is with no guarantees or support. If you have comments or wish to report bugs, please send them to `teemu.hynninen@tut.fi`.

2 Running

How to get started?

ASH is provided as a jar-package. There are both an independent version and a version that uses the jline package to emulate a shell. Use of the latter is highly recommended, but it requires that you get the jline extension from "http://jline.sourceforge.net". The program is written using features from Java 1.5, so you need to have the Java Runtime Environment updated to at least that version.

The program is run from the command line. To execute the program, go to the directory where you have put the jar file and give the command `java -jar ash.jar`. You should see a short splash text (see Figure 1 (a)) and a window should open see (Fig. 1 (b)). To do things, type commands in the ASH command prompt. To see thing, look at the graphics window which should update automatically as you proceed. The 3D view can be rotated using the mouse and arrow keys.

(a)

```
nayru-2 ~/Documents/Java/Ash/demo >java -jar ash.jar
This is Atomic Structure Handler 0.35

- view the available commands by typing 'list command'
- view the manual by typing 'man' followed by the name of a command
- edit launch options in the file .ash_launch
 > open script skripti3.txt
frame 2
new supercell:
cell vector 1   (     3.00000,      0.00000,      0.00000)
cell vector 2   (     0.00000,      3.00000,      0.00000)
cell vector 3   (     0.00000,      0.00000,      3.00000)
 > value nparts
nparts =    57.000000
 > quit
unknown command quit
 > exit
nayru-2 ~/Documents/Java/Ash/demo >
```

(b)

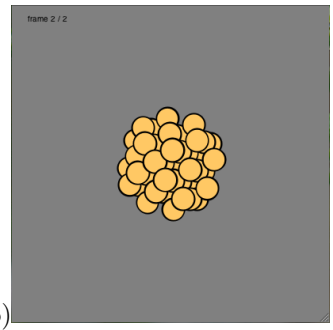

Figure 1: (a) ASH is run from the command line. (b) A visualization window will draw what you build as you progress.

3 The manual

What to do?

To do things, you need to command the program. To do that, you need to know what commands it understands. To see a list of commands, type `list command`. To see how the commands actually work, give the command `man` followed by the name of a command, e.g., `man create` (see Figure 2). You will see a message detailing the syntax (optional arguments and options are wrapped in square brackets `[]` and non-keyword arguments such as numbers are wrapped in parenthesis `()`), functionality, and possible options and arguments of the command. If you try to use an invalid syntax for a valid command, the syntax part of the manual is automatically shown, which may or may not remind you of the proper syntax.

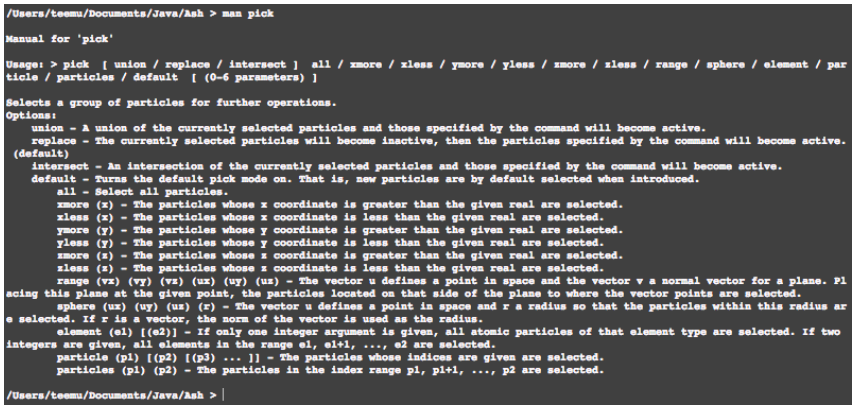

Figure 2: Access the built in manual with the command `man`.

4 The calculator

What goes where?

The program has a simple command line calculator to allow the calculation of, for instance, coordinates on the fly (see Figure 3). The calculator has access to structural data in the current geometry through variables and functions such as `cell` and `coord`, and it can handle vector and matrix operations. Custom variables and functions can be defined using the command `define`, and evaluation of expressions using `value`. The calculator is automatically invoked when expressions are given where numerical values are expected. Evaluation can also be forced within a command before the command itself is parsed by wrapping an expression in dollar signs (`$`). A list of variables can be seen with the command `list value` and similarly functions are listed when `list function` is typed. A separate manual exists for the calculator functions, invoked by the command `calcman`.

5 The logic

What's the big idea?

Manipulation of structures in ASH is done almost exclusively by typing in commands in the prompt. The available operations include creating and deleting atoms and clusters, shifting and rotating them, deforming structures, etc. The general logic of these operations is such that a manipulation command defines the operation to be performed and that operation

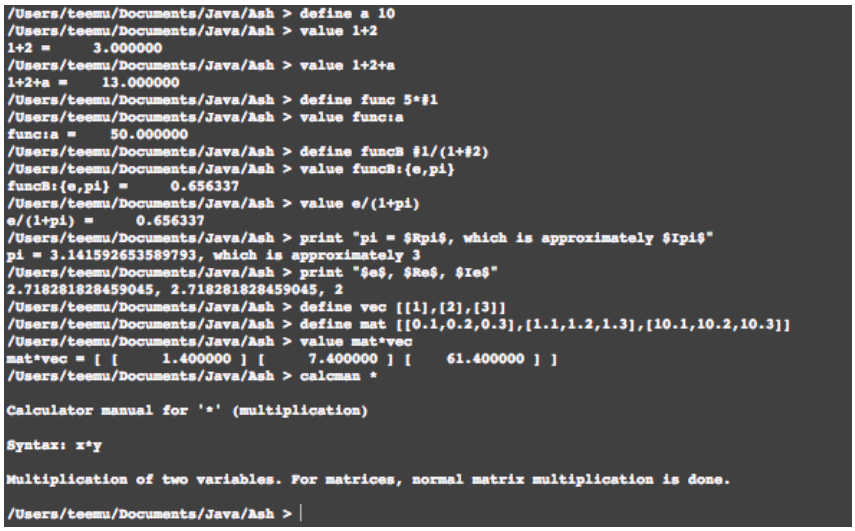

Figure 3: The calculator features can be accessed in several ways.

then acts on all currently *active* particles. The activeness of particles is defined using the `pick` and `unpick` commands, and it is displayed in the graphics window through particle highlighting. In addition, the commands `cluster` and `uncluster` can be used for joining atoms in clusters which are then manipulated as units in most operations.

Commands are usually full words with recognizable meaning, such as `shift`, `rotate`, `create`, `delete`, etc. Instead of giving a separate command for every task, many operations are grouped under one main command, a header if you like, and specified through additional options. On the other hand, for some commands options are required just to specify details on how the operation should work. For instance, `open` needs an option specifying the format of the file it tries to open (and the filename), e.g., `open xyz geo.xyz`, but `show directory` and `show cell` do different things, although both are related to what information is displayed. As the command names tend to be long, you may want to rename them at some point. This can be done using `alias`.

Options and arguments always follow the main command and are separated by spaces. If a numeric argument is expected, the program automatically tries to parse a number or evaluate a mathematical expression. If a vector is expected (e.g., the displacement vector when shifting particles) it can usually be given either as a vector or as a list of scalars.

The key characters defining how input is parsed are spaces (' '), quotes ('"' and ''''), semicolons (';'), dollar signs ('$'), and triple hashes ('###'). Spaces act as delimiters for arguments. Semicolons are delimiters for full commands. Quotes group strings into single arguments overriding both spaces and semicolons they wrap. Quotes can be nested by alternating between single and double quotes. Dollar signs force the evaluation of an expression before parsing the actual command. Hashes are used to define comments. In the current version they are absolute: you cannot wrap a triple hash in quotes or try to escape it, as it will always be treated as a comment.

The graphics window is usually just a display of the current geometry, and it updates automatically. Particles are displayed according to their type (size, color) and activeness (outline), and the view can be rotated using the mouse and arrow keys. The display is configurable using commands `recolor`, `resize`, `view`, etc. It is also possible to select particles or inquire their properties with the mouse by first giving the command `mouse` to set mouse interaction options.

ASH can also be run through scripts. Scripts can be executed with `open script` and the whole program can be set to run a script upon launching by including the command as an launch argument, e.g., `java -jar ash.jar open script script.txt`. (You can give other commands as launch arguments as well.) If a file `.ash_launch` exists, it will always be treated as a launch script file. Scripts can contain all the normal commands accepted during interactive use, and in addition also `if`–`else`–`endif` and `while`–`endwhile` logic structures.

References

[1] For ASE, see "https://wiki.fysik.dtu.dk/ase/overview.html".