PORTING SCIENTIFIC SOFTWARE TO WINDOWS (OR NOT) ?

NOBUGS October 2004

Andy Hammersley

ESRF Grenoble France

(An Aside)

- Software engineering is about GOOD DESIGN (which meets the requirements)
- If the design is good new features can be added easily within the existing framework
- If the design is bad adding new features will “bend” and “break” the framework
- Every framework has its limits
- Unfortunately good design is not simple

OVERVIEW (Relevant to all User Controlled Programs)

- The Rise of Windows (and alternatives)
- What is a ported Windows application
- Porting experience
- Partial porting and alternative approaches
- Conclusion

WHY CONSIDER WINDOWS ?
(~10 Years ago we didn't)
1. PC HARDWARE

- Extremely Cheap (Remember workstation prices)
- Powerful (Only High end systems such as the Alpha and G5 significantly more)
- Versatile (Reasonably configurable)
- Convenient (Buy at Supermarket)
- Modular
- Portable (Even wearable !)
- Universal
- Becoming Scalable (64 bit, multi-core, Blade systems)

WHY CONSIDER WINDOWS ? 2. WINDOWS OPERATING SYSTEMS

- User Friendly (well much more than Unix)
- Enormous range of user software; both commercial and free
- Universally available
- Little System Administration
- Native on Latest Hardware e.g. Lap-tops
- Very high degree of compatibility between versions

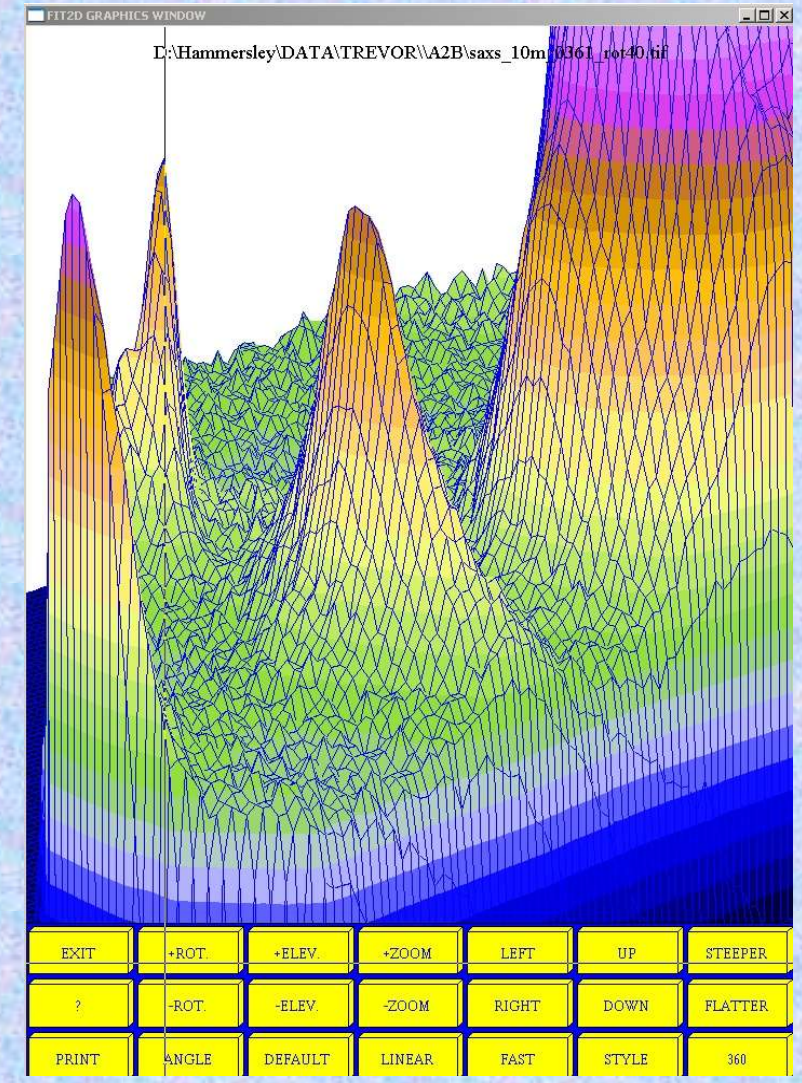
IS LINUX GOOD ENOUGH ON THE DESKTOP ?

- “Not for my grandfather” (or IBM)
- Are dual boot machines a sufficient answer ?
- When will there be a single agreed Linux window manager style ?
- What is happening with commercial Linux ?
- (Can Bill Gates and others use IP’s to kill free software ?)

SCIENTIFIC SOFTWARE HISTORY

- Some dates back to mainframe era
- Much developed on VAX/VMS
- Ported to many different Unix systems
- Graphics developed using X-11 Window System
- Often ported to Linux
- Many different types of graphical and command line interfaces

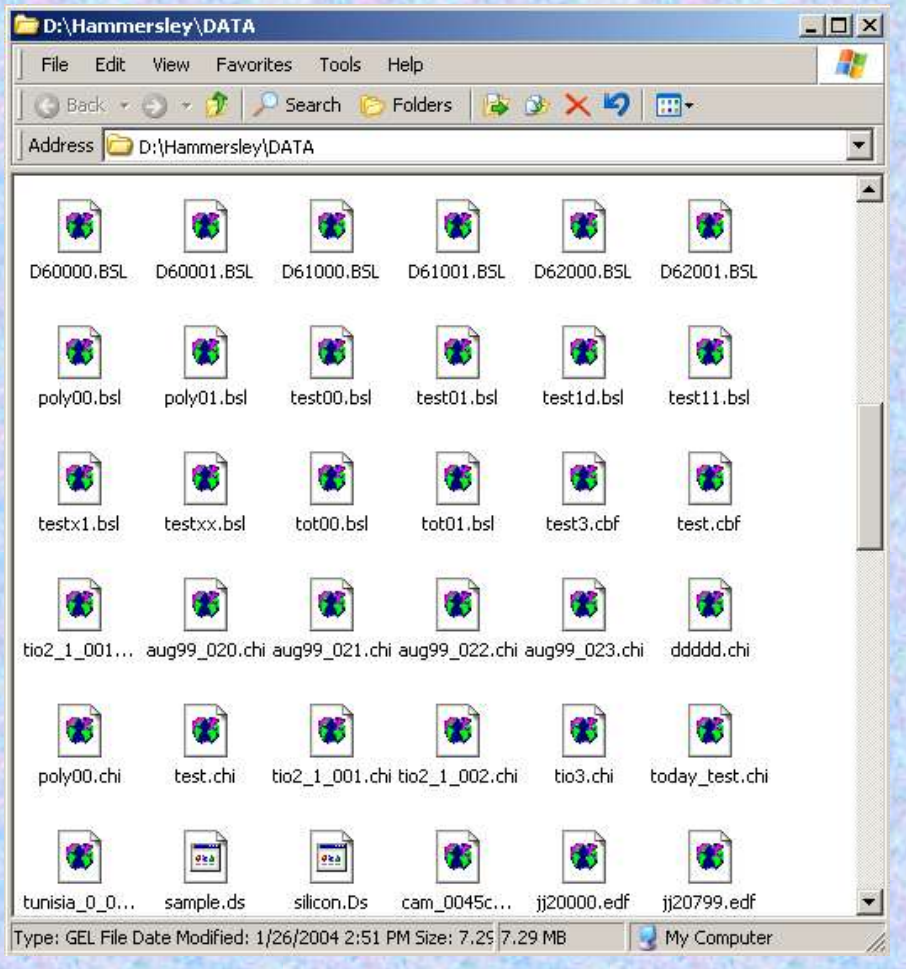
Keyboard and Graphical User Interfaces



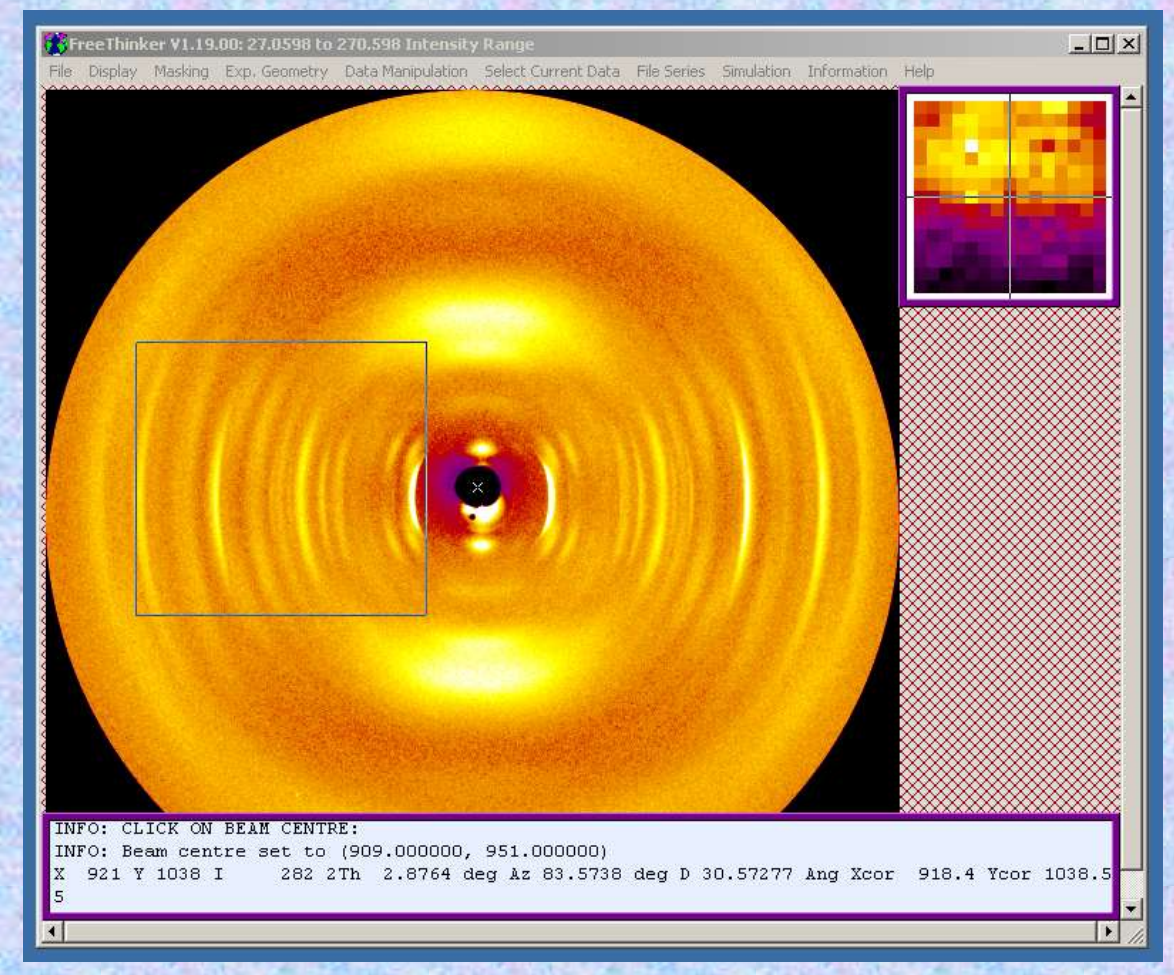
WHAT ARE THE CHARACTERISTICS OF A WINDOWS APPLICATION ?

- Associate program with file type(s)
- Ability to drop files onto program icon
- Standard menu bar options (plus others)
- Standard “File” menu options (plus others)
- Standard Short-cut keys (e.g. Print: CTRL-P)
- “Click and Drag” interaction

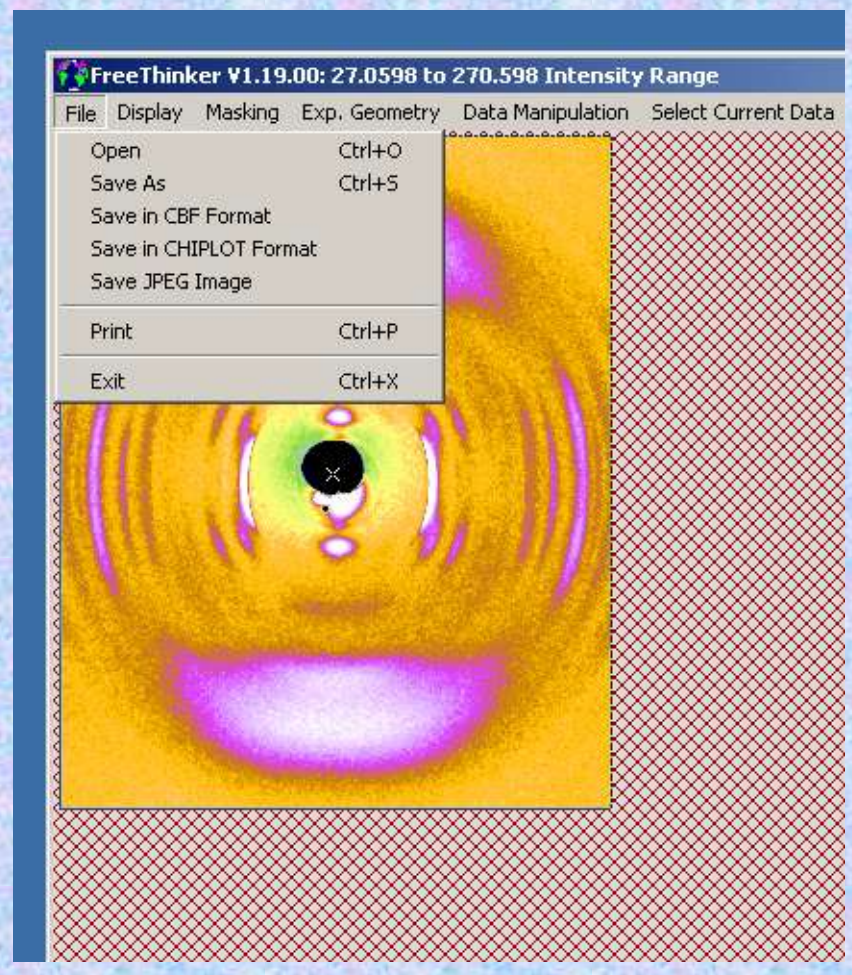
DATA FILES ASSOCIATED WITH FreeThinker



WINDOWS APPLICATION MENU BAR AND DISPLAY



WINDOWS “FILE” MENU



GOOD POINTS OF WINDOWS API

- Consistency
- Documentation (Generally)
- Menu-bar and menus
- Bitmap hardware display independence
- Rotated text
- Form / Dialogue editor and wizards
- Fortran 90 better on Windows than many sub-standard Unix products

BAD POINTS OF WINDOWS API

- Where's Posix (.1) Support ?
- Creating your own device independent bitmap and manipulating it is not trivial and only documented on the web
- Rotated text may not appear as requested, and without an error message

PARTIAL PORTING OF APPLICATIONS

- i.e. Get program running on Windows, but with the user interface as before e.g. FIT2D
- Much easier / faster approach
- Cygnus, etc. provide useful tools, but with drawbacks
- Command Window / Gnuplot approach

ALTERNATIVE APPROACHES ?

- Scripting languages: TOO SLOW, compatibility and support problems
- Java: Too slow, at best a hybrid solution would be necessary with its inherent problems
- High level platform independent component set (a pipe dream)

CONCLUSIONS

- (Porting) Applications to Windows will be more and more demanded
- Partial porting is possible and relatively easy, but results in a Unix application running on Windows
- Full porting may require complete re-write of applications and is a lot of work.
- Well written modular code can be re-used, but a lot of re-arrangement is necessary