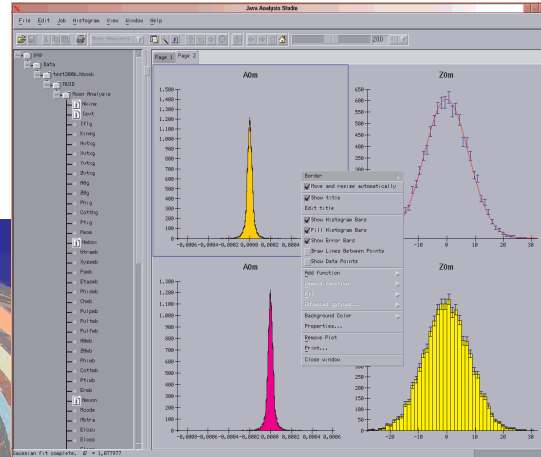


- *Pros*
- *Cons*
- *C++ & Java Solutions*
- *Plan*

Figure 1 consists of two panels. The left panel is a phase space plot showing a central point, concentric circles, and a dense network of trajectories in yellow, green, and blue. The right panel is a zoomed-in view of a single trajectory, highlighting its complex, fractal-like structure with a color gradient from blue to red.



Atlas, for the time being, has decided that the baseline implementation language is C++.

1

- we're thinking and discussing the evolution of Graphics for more than a year
- one of the alternatives has been from the beginning to rewrite everything in Java
- as time goes, it becoming clear that we'll migrate to Java "in future"
- at the same time, we see some excellent applications written in Java, which we want to use (they are shown on this slide)
- we've also looked at Java and found it much superior to C++
- so we should ask the question "why not to migrate now ?"
- I'll mention some Pros & Cons, I'll compare C++ & Java Solutions and I'll propose possible plan for further evolution of the Atlas Graphics

Graphics in Java

Pros

- ☺ *Many advantages mentioned by Steve*
- ☺ *Direct connection to Java applications, which are mostly better than their C++ equivalents (Wired, JAS,...)*
- ☺ *Trivial implementation of wide range of functionality (XML, Java3D,...)*
- ☺ *Easy creation of multiple object Representations, even during run-time*
- ☺ *Easy definition of default behaviour*
- ☺ *Easy run-time loading*
- ☺ *Easy personal storage*
- ☺ *Easy network access*
- ☺ *Easy multi-threading*
- ☺ *Other benefits mentioned later*

2

-
- if we don't migrate to Java, we'd have either to use obsolete C++ tools or interface Java Tools to C++
- XML, Java3D is much more natural to Java then to C++
- Java easily allows for flexibility in running code, the dynamic configuration is tricky in C++ but natural in Java
- dtto
- property of Java
- simple storage by serialisation
- network is natural to Java
- multithreading is required for Graphics and difficult in C++
-

Graphics in Java

Cons

☹ Current Atlas SW:

- ☹ F77
- ☹ F77 clever tricks
- ☹ Zebra
- ☹ C++
- ☹ C++ clever tricks
- ☹ Age
- ☹ F77 from Age
- ☹ C++ from Age
- ☹ F90
- ☹ Objy
- ☹ G4

*All that can be
migrated/interfaced/wrapped.*

*The overall complexity and coupling
makes in hard.*

*However,
any evolution of the Atlas SW
will have to face this problem.*

3

- current Atlas sw is incredibly complex

-
-
-

- the problems are specific to Java, any evolution will have to solve them

Graphics in Java

C++ & Java Solutions (1)

- Control:
 - *To satisfy requirements about independency of objects on Graphics and Graphics on concrete package*
 - *C++ requires tricky MultiMethods or ugly switches*
 - *Java can use Reflection*
- GUI:
 - *C++: Qt*
 - *Java: AWT, Swing*
- Scripting:
 - *C++: Cint, Swig*
 - *Java: BeanShell, DynamicJava,...*
- Simple Storage:
 - *C++: Objy, Rio, XML*
 - *Java: Serialisation, XML*

4

- basic Requirements on Graphics:
 - Graphics should not polute the rest, the rest shouldn't know about it
 - Graphics should be independent on any concrete graphical package
- Qt is quite good, it even looks similar to Java (messaging - callback,...), AWT & Swing are natural to Java
- Cint & Swig require additional work, BeansShell,... work from outside (the core code is untouched)
- Serialisation is natural to Java

Graphics in Java

C++ & Java Solutions (2)

- Event Display:
 - C++: Aravis, OpenInventor
 - Java: Wired, jAtlantis, Java3D, GraXML
 - Statistics:
 - C++: HBook, HTL, Iguana, Root
 - Java: JAS
 - XML:
 - C++: possible
 - Java: natural
 - Text (logging):
 - C++: easy
 - Java: automatic
- 

5

- we have all we need in Java, Aravis could be easily re-written
- see JAS talk
-
- each Java object can write itself

Graphics in Java

Plan Proposal (1)

- ✓ Up to Now:
 - ✓ *Proof of feasibility by (mature) products:*
 - ✓ *Wired, JAS, Colt, GraXML,...*
- Till May WS:
 - *Control designed and implemented*
 - *DOM/XML and Text/Log Scenes designed and implemented*
- Till August WS:
 - *Access to C++ objects demonstrated*
 - *Wired, JAS interfaced*
- Till November WS:
 - *Alpha version of complete Graphics available*
- Till EOY'00:
 - *Complete migration to Java*

the only difficult part

- feasibility is already proofed
- basic Control already exists
- interface to C++ is the only real problem (and it would disappear if also the rest in in Java)

Graphics in Java

Plan Proposal (2)

- *Support for the current (C++/F77) implementation till the EOY'00*
- *Close collaboration with other Java developpers and users (Wired, JAS, Colt,...) - Atlas-specific layer could be very thin, the rest could be shared*

7

- support for C++/F77 implementation till the migration is finished
- we can reuse & collaborate - Java is excelent for it

Graphics in Java

Info



Java for Atlas

<http://www.cern.ch/Atlas/GROUPS/GRAPHICS/Texts/Documentation/Graphics/Java/>

CERN Java Tutorials (free)

Tu & Fr @10:00 in IT

<http://consult.cern.ch/service/training/>

8

- links to pages of interest to Atlas
- Tutorial serie starts next Friday, good slides are on the Web