On the possibility of a shared event display code base for Halls B & D

Outline

1. Event Display Philosophy
2. What do we have now?
3. Web Visualization via FLEX

1. Philosophy

- The primary role of an event display is *not* to visualize the detector.
- The primary role of an event display is to debug and diagnose the detector.
- In support of its primary role, unfaithful (to the geometry) displays are often more useful that faithful displays. Especially when there is a lot of “air.”

2. What do we have now?

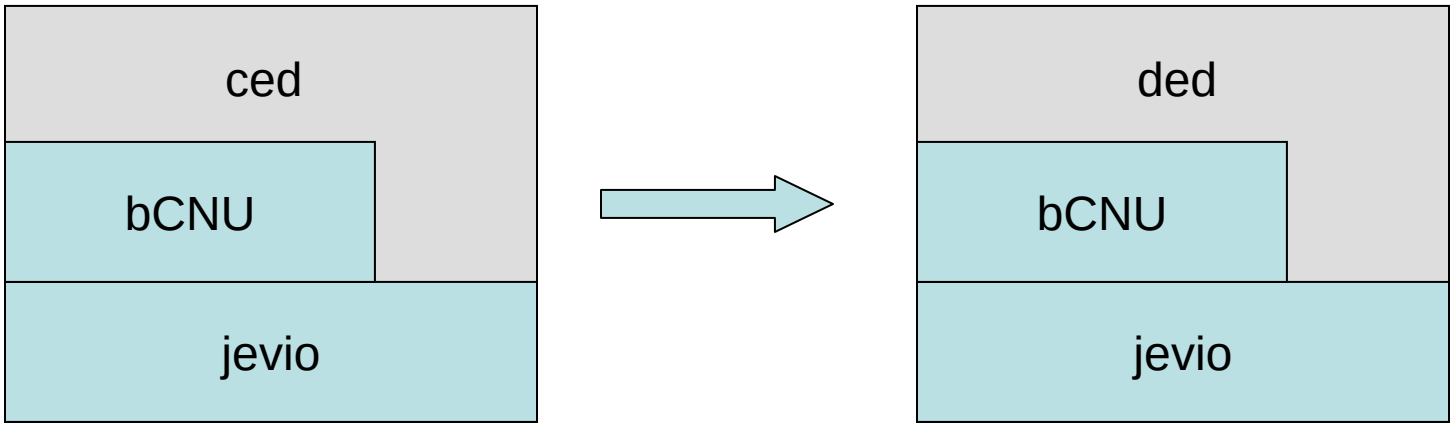
A solution based on three JAVA projects:

1. **jevio**[†]—reads evio (event) files and maps the events onto a tree data structure. Dependencies: *none*.
2. **bCNU**—Framework for creating a MDI (Multiple Document Interface) application and for distributing events to listeners. Knows nothing about any specific detector. Dependencies: *jevio*.
3. **ced**—the event display, magnetic field reader, and particle swimmer for CLAS 12.0 GeV. Dependencies: *jevio and bCNU*

– Implication: keep jevio, bCNU—replace ced with ded (“dead?”)

[†]**jevio** is no longer a Hall B project but a proper JLab DAQ project. Which means: don’t call us to report bugs.

Or:

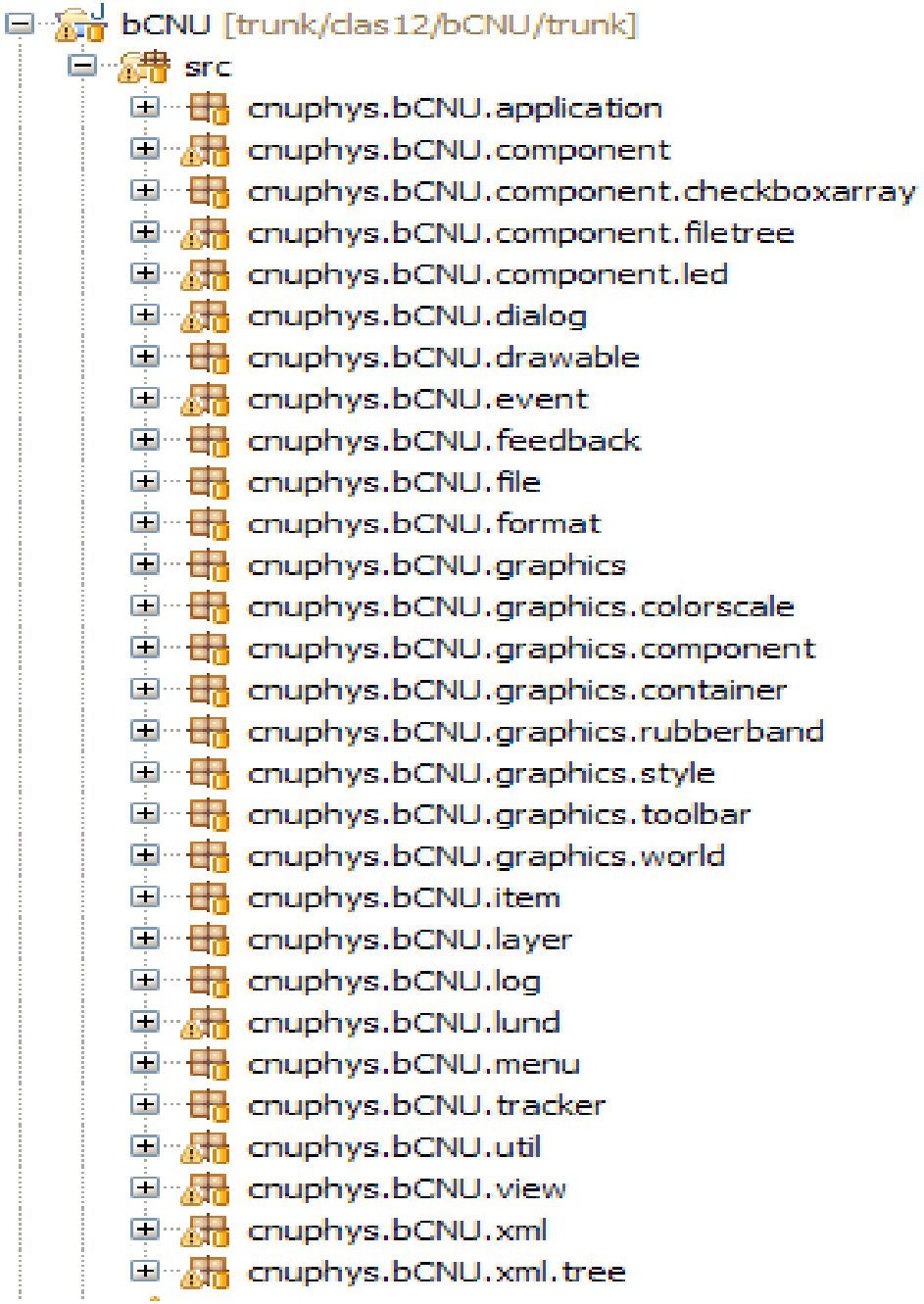


What does ***ced*** know that ***bCNU*** doesn't? To 1st order: ced knows about 1) CLAS detector geometry and 2) the CLAS event format.

Rough breakdown

- bCNU 57%
- ced 26%
- jevio 17%

- Note: roughly 20% of the ced project is devoted to magnetic fields viz., reading, interpolating, and swimming. **If we agree on a magnetic field (binary) format, all magnetic field related code could be moved to bCNU and shared.**



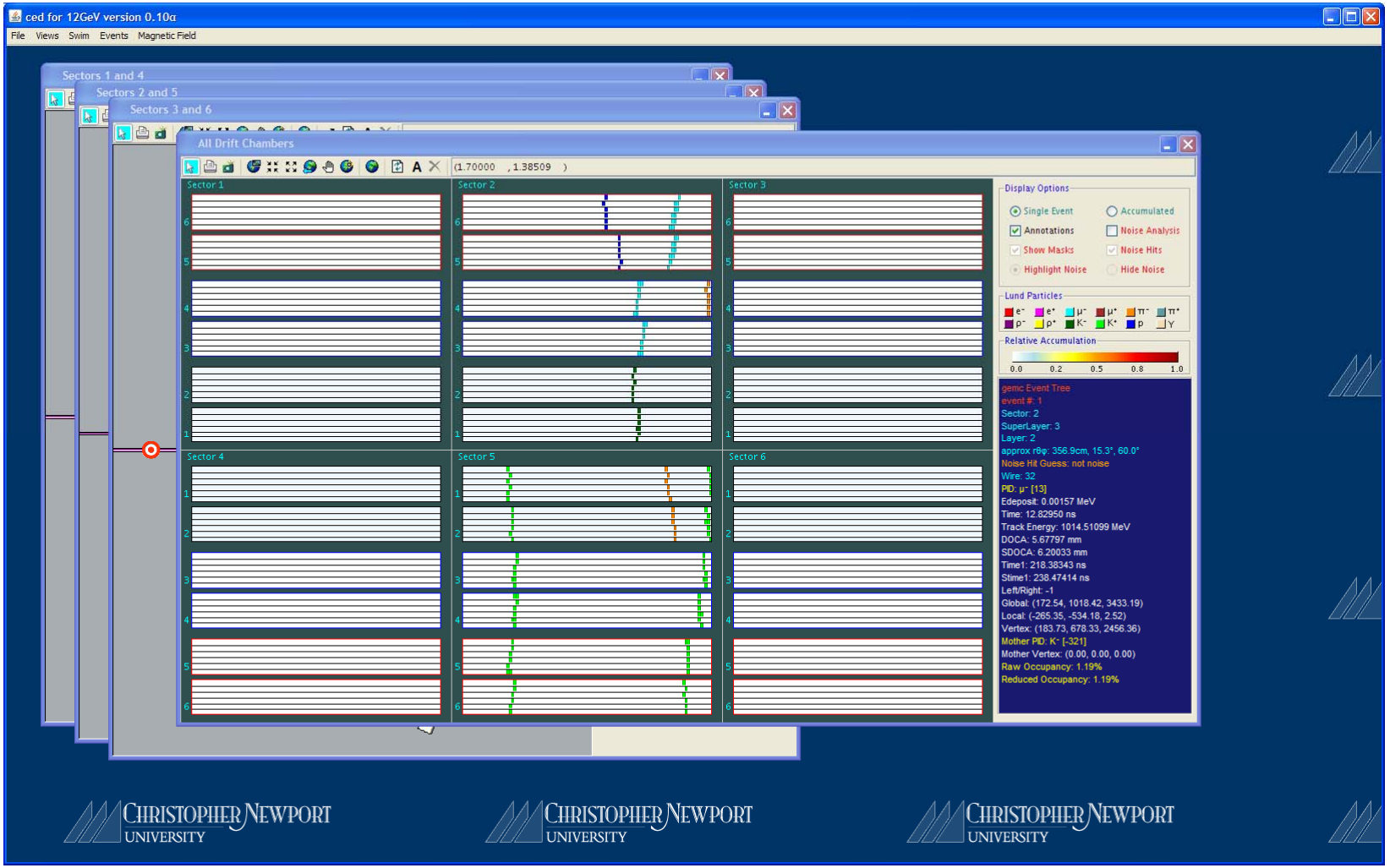
bCNU packages

While there are a couple of obscure packages (e.g., lund, which is a database of Lund particles) most are ~self-explanatory.

In a nutshell, bCNU provides:

1) A MDI Framework
2) Support for the windows on that framework (called views)
3) Support for items on the views (called items)
4) Support for mouseover feedback (including headsup display)
5) Layering
6) XML reading/writing/visualization
7) Evio events
8) Menus, toolbars, dialogs, etc
9) Some useful components (check box arrays, color scales)
10) Possibly magnetic field?

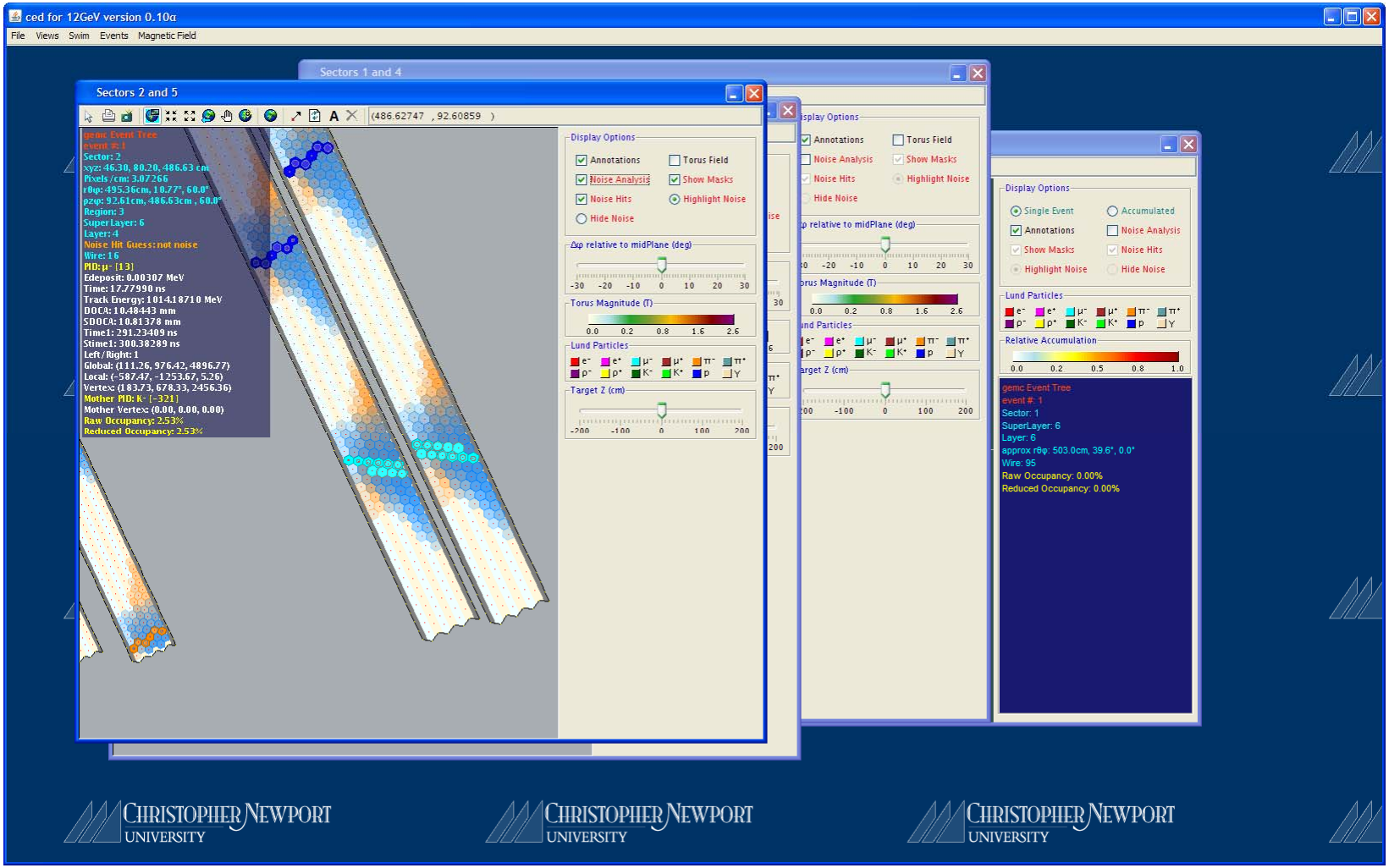
E.g., ced



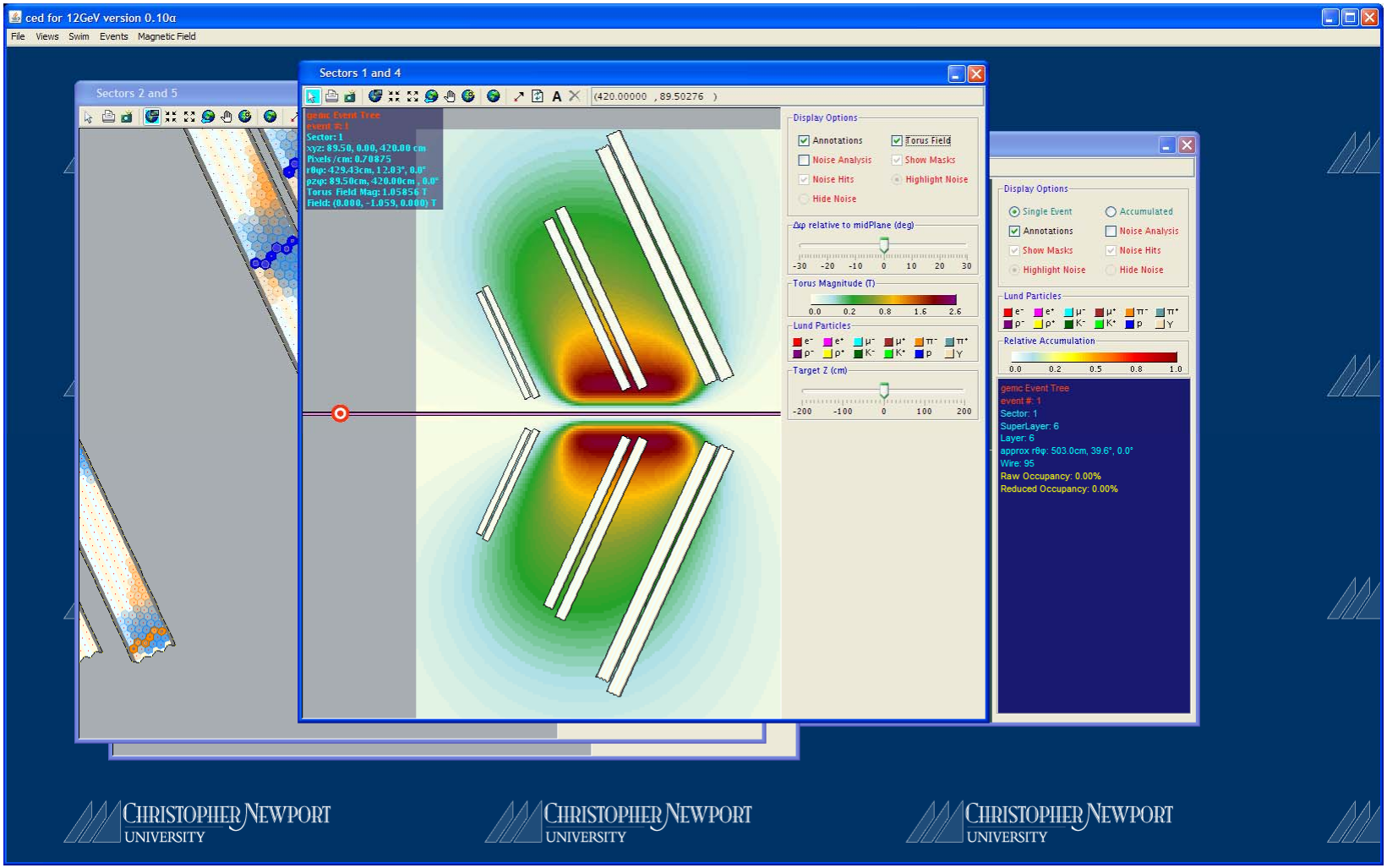
Ced: defines the drift chamber “item” based on bCNU base class. Everything else comes from bCNU.

(cont.)

Headsup display (for mouseover feedback) conserves screen real estate.



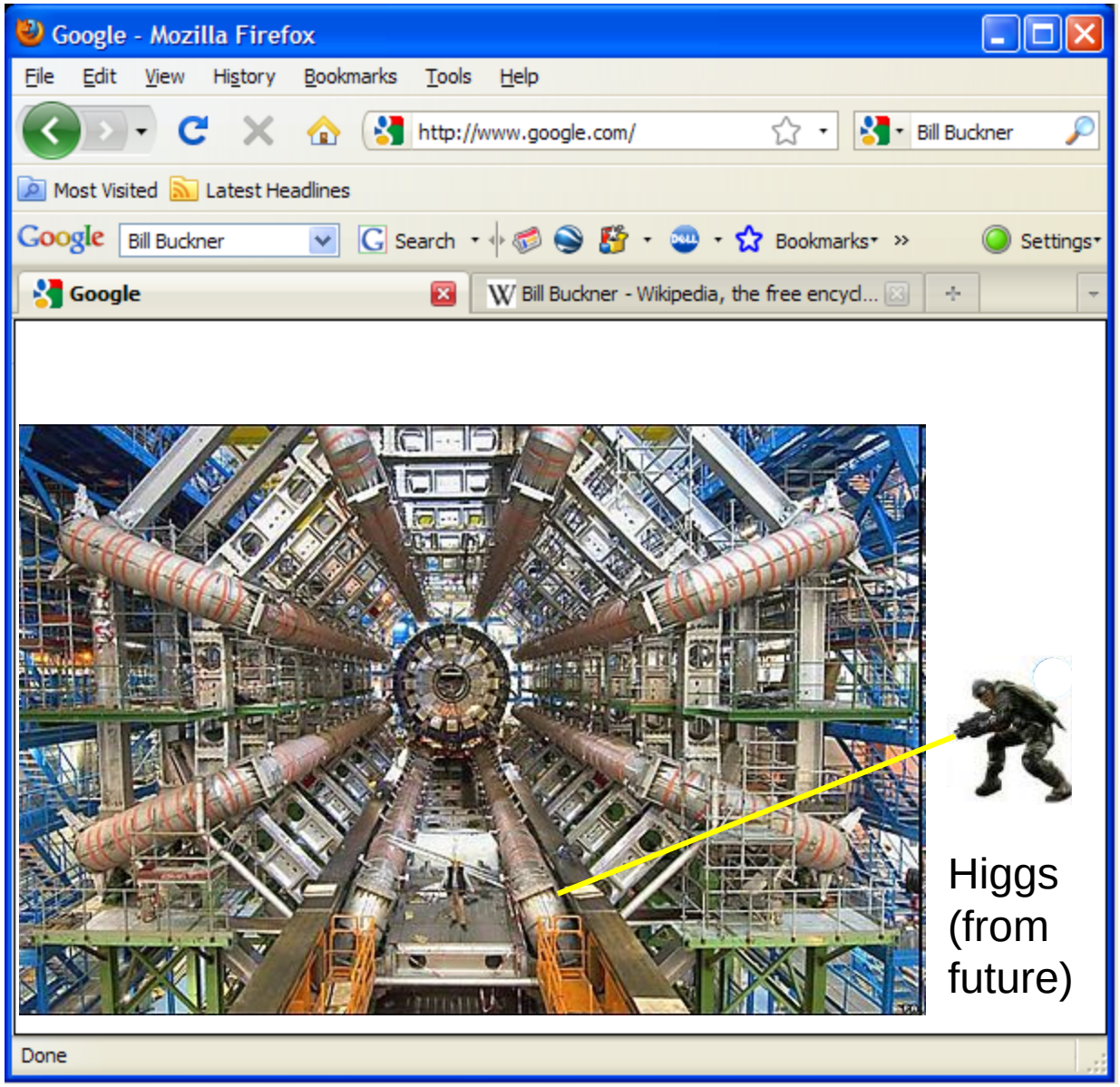
(cont.)


ced for 12GeV version 0.10α
File Views Swim Events Magnetic Field
Sectors 2 and 5
Sectors 1 and 4
(420.00000 , 89.50276)
gemc Event Tree
event #: 1
Sector: 1
xyz: 89.50, 0.00, 420.00 cm
Pixels/cm: 0.70875
rθφ: 429.43cm, 12.03°, 0.0°
ρzφ: 89.50cm, 420.00cm, 0.0°
Torus Field Mag: 1.05856 T
Field: (0.000, -1.059, 0.000) T
Display Options
Annotations
Torus Field
Noise Analysis
Show Masks
Noise Hits
Highlight Noise
Hide Noise
Δφ relative to midPlane (deg)
-30 -20 -10 0 10 20 30
Torus Magnitude (T)
0.0 0.2 0.8 1.6 2.6
Lund Particles
Target Z (cm)
-200 -100 0 100 200
Display Options
Single Event
Accumulated
Annotations
Noise Analysis
Show Masks
Noise Hits
Highlight Noise
Hide Noise
Lund Particles
Relative Accumulation
0.0 0.2 0.5 0.8 1.0
gemc Event Tree
event #: 1
Sector: 1
SuperLayer: 6
Layer: 6
approx rθφ: 503.0cm, 39.6°, 0.0°
Wire: 95
Raw Occupancy: 0.00%
Reduced Occupancy: 0.00%
CHRISTOPHER NEWPORT UNIVERSITY
CHRISTOPHER NEWPORT UNIVERSITY
CHRISTOPHER NEWPORT UNIVERSITY

3. Web Visualization

- We want a full-featured, fully interactive *ced* to operate in a browser. From anywhere in the world. To see live events in real-time. With minimal bandwidth penalty. To offer an unprecedented level of remote monitoring for our collaboration.
- The same technology could be used for other monitoring/analysis/simulation.
- The technology for doing this is, as they say, upon us.

Rich Internet Applications (aka Web 2.0)



1. Browser *delivers* virtual machine and *provides* real estate.
2. Compiled application runs in virtual machine.
3. Virtual machine, not browser, renders the display.

RIA Candidate Technologies

- Adobe FLEX (2004.) Uses FLASH player as virtual machine. **Approximately 98 percent penetration across all platforms**.†
- Microsoft Silverlight (2007.) So far, little penetration.
- SUN JavaFX (little chance to succeed.)
- In Web 1.0 you programmed to the browser (IE, Firefox, Chrome, etc.) Here you program to a single virtual platform. In Web1.0 the closest technology was JAVA applets. FAIL.

†This is the number one reason for adopting FLEX. Almost nobody will have to download anything.

FLEX vs. JavaScript/AJAX

OLD: JavaScript/AJAX	***NEW: FLEX***
• Interpreted • Browser dependent • Can be disabled! • Data transfer: text over HTTP • Request/Poll/Stateless (uses tricks like cookies to mimic statefulness.) • Page based • Limited, inextensible controls†	• Compiled ActionScript (fully OO) • Browser agnostic • Multiple transfer protocols including binary at 10× speed of XML or SOAP • Pub/Sub/Stateful • Data centric • Powerful, extensible controls including charts, tables, drag ‘n drop

†If you are Google you can work miracles like drag ‘n drop. Mere mortals cannot.

FLEX Virtual Machine

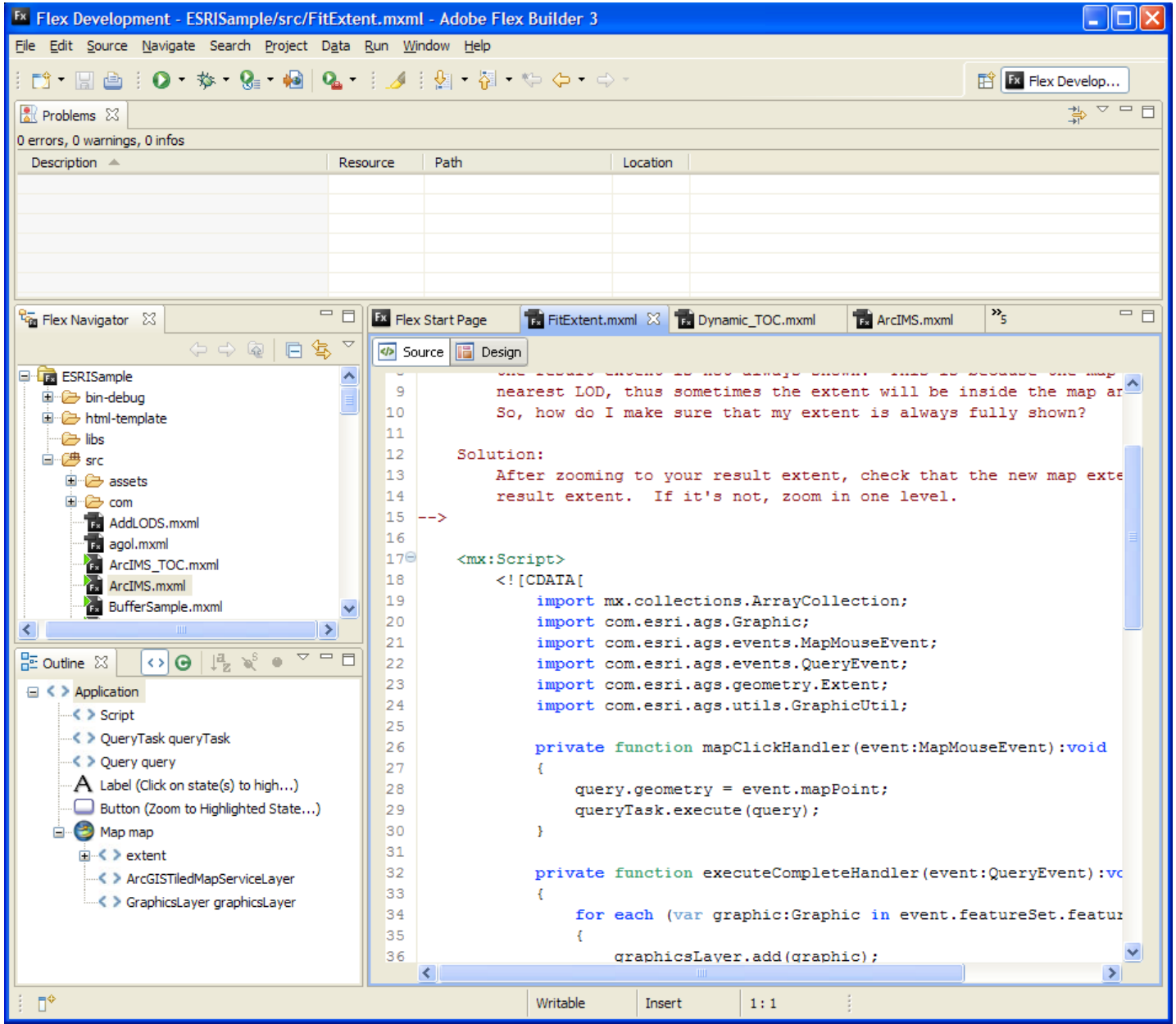
- The ubiquitous FLASH player
  - Powerful, small, fast, multithreaded
  - Windows, MAC OS, Unix, Linux, Phones, PDAs’
- Why FLEX instead of FLASH?
  - FLASH has an animation paradigm (timelines, layers, etc. Caters to creative types.)
  - FLEX has a programming paradigm (normal OO constructs. Welcome nerdy developers.)
  - FLEX has profiling, refactoring, wizards, graphical design, charting, data grid, etc.

FLEX Development

- Free SDK
- No runtime license cost (i.e., everyone has flash)
- eclipse based IDE (Flex Builder) free for students, faculty, (JLAB staff??)†
- Essentially two languages:
  - MXML for the display (V in MVC—Model View Controller paradigm)
  - ActionScript for the “brains” (C in MVC)

† Probably free. Someone should try. $699 for everyone else. Bummer.

FLEX Builder (eclipse)



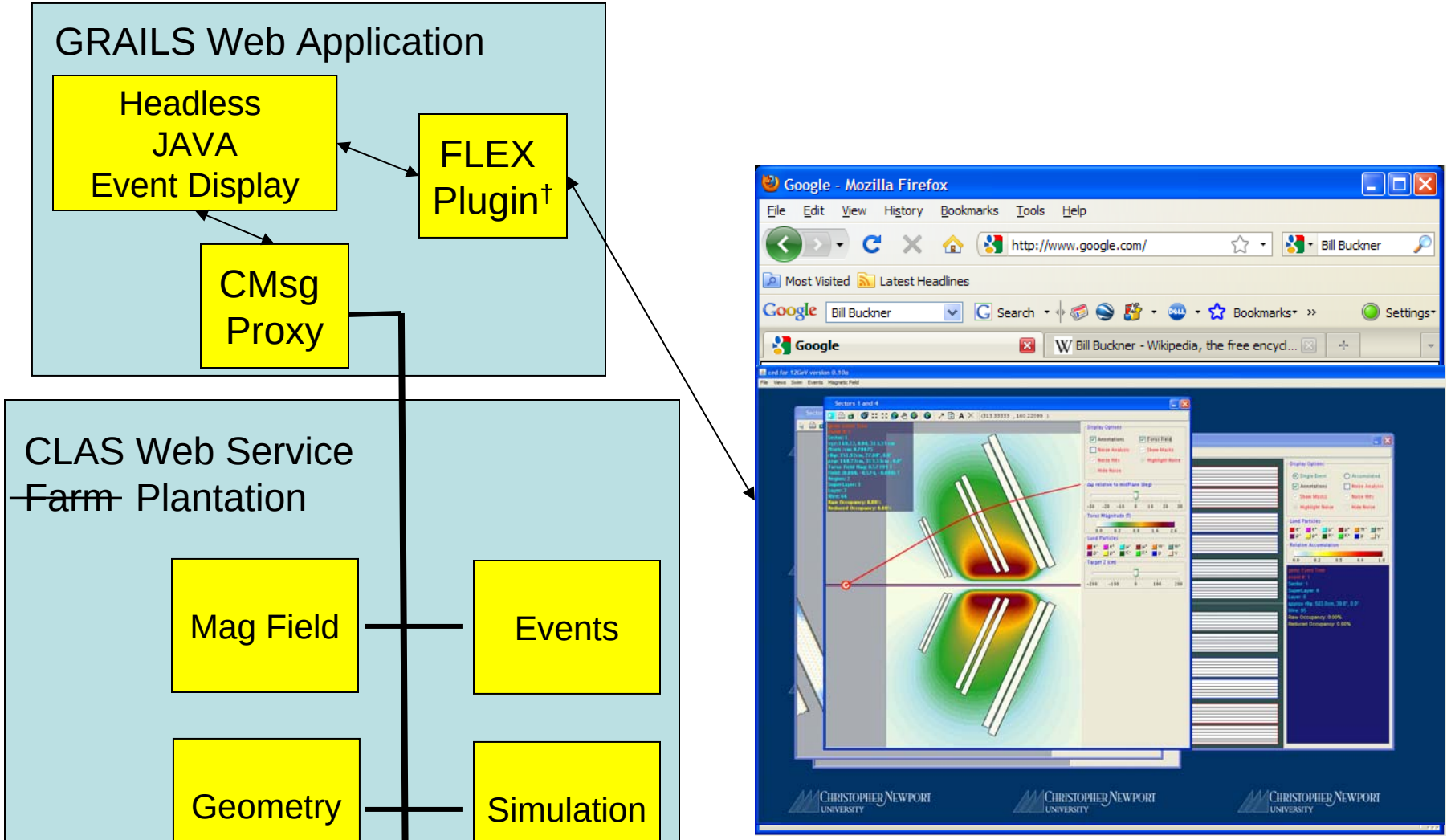
Data Exchange

- Most natural way is through standard Web Services
- FLEX is Web Service Aware out of the box
- Also possible: higher level server side frameworks (e.g., GRAILS, CLARA). What do they provide? To first order they handle all the annoying plumbing involved in using web services, including (GRAILS) mapping local objects onto remote objects.

Proposed Architecture

- The basic idea is to use the stand-alone code in the backend as a headless server
- A map server (like Google maps) takes a request and provides an image on top of which other things may be drawn
- The event display server would do the same, but with a detector view rather than a map

Architecture Cartoon



†FLEX plugin for GRAILS already exists

Conclusion (regarding Web Visualization)

- We can preserve existing bCNU/ced code base
- The stand-alone event display has no technical risk
- The FLEX technology is a *minor* technical risk, not on the critical path, and independent of the stand-alone application.
- The GRAILS backend is a somewhat greater technical risk, and alternatives should be explored
- The benefits are legion: no deployment, extremely powerful remote monitoring, no meaningful bandwidth hit.