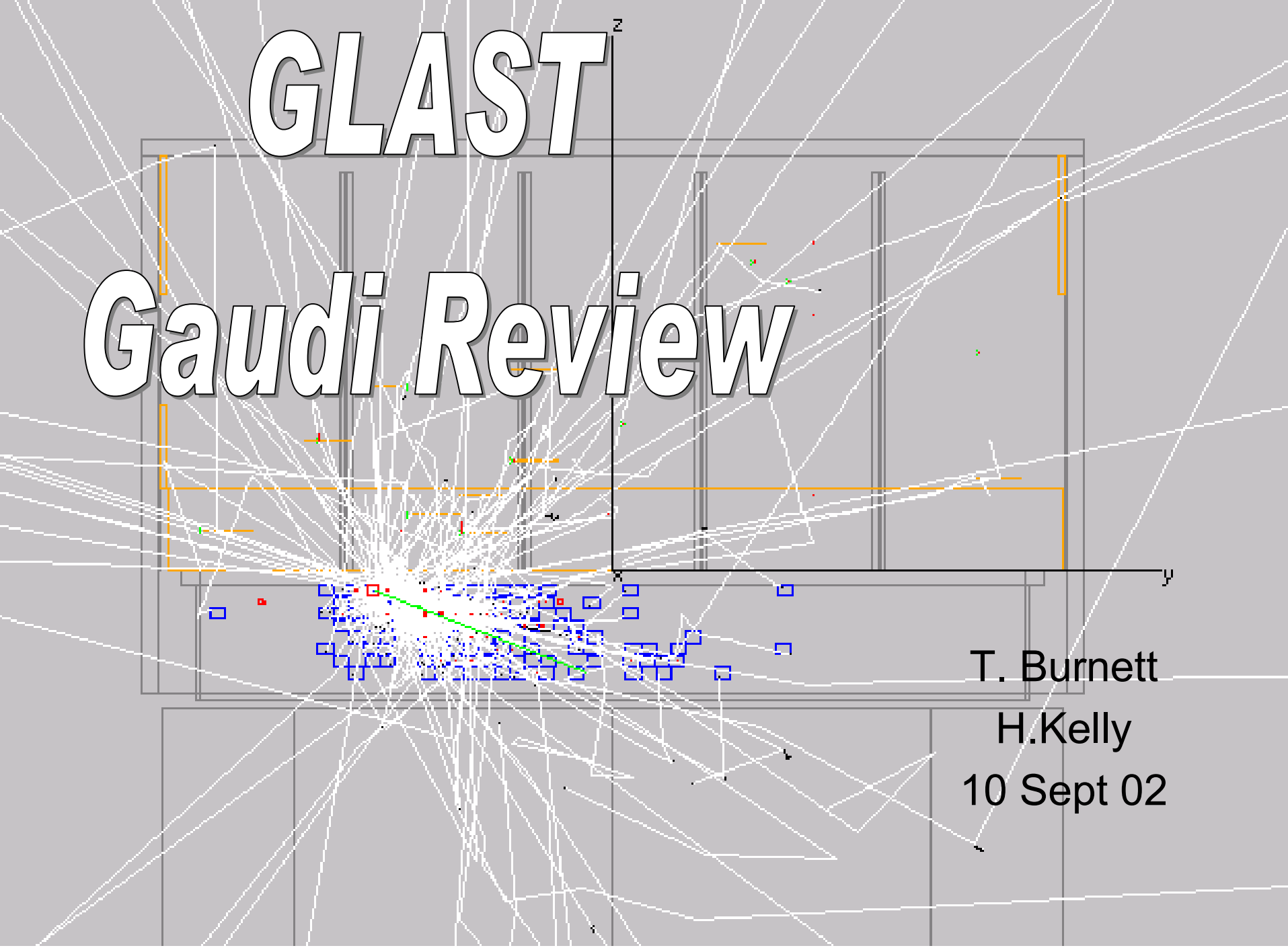
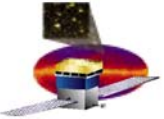
GLAST Gaudi Review



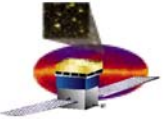
T. Burnett
H.Kelly
10 Sept 02



GLAST LAT

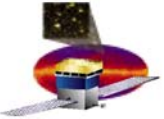
Purposes of the review:

- Trigger a comprehensive review of how we are using Gaudi
- Improve, for us and the reviewers, the documentation
- Generate a list of TODOs for improving usage and documentation
- Get feedback from our external reviewer!!!



Outline

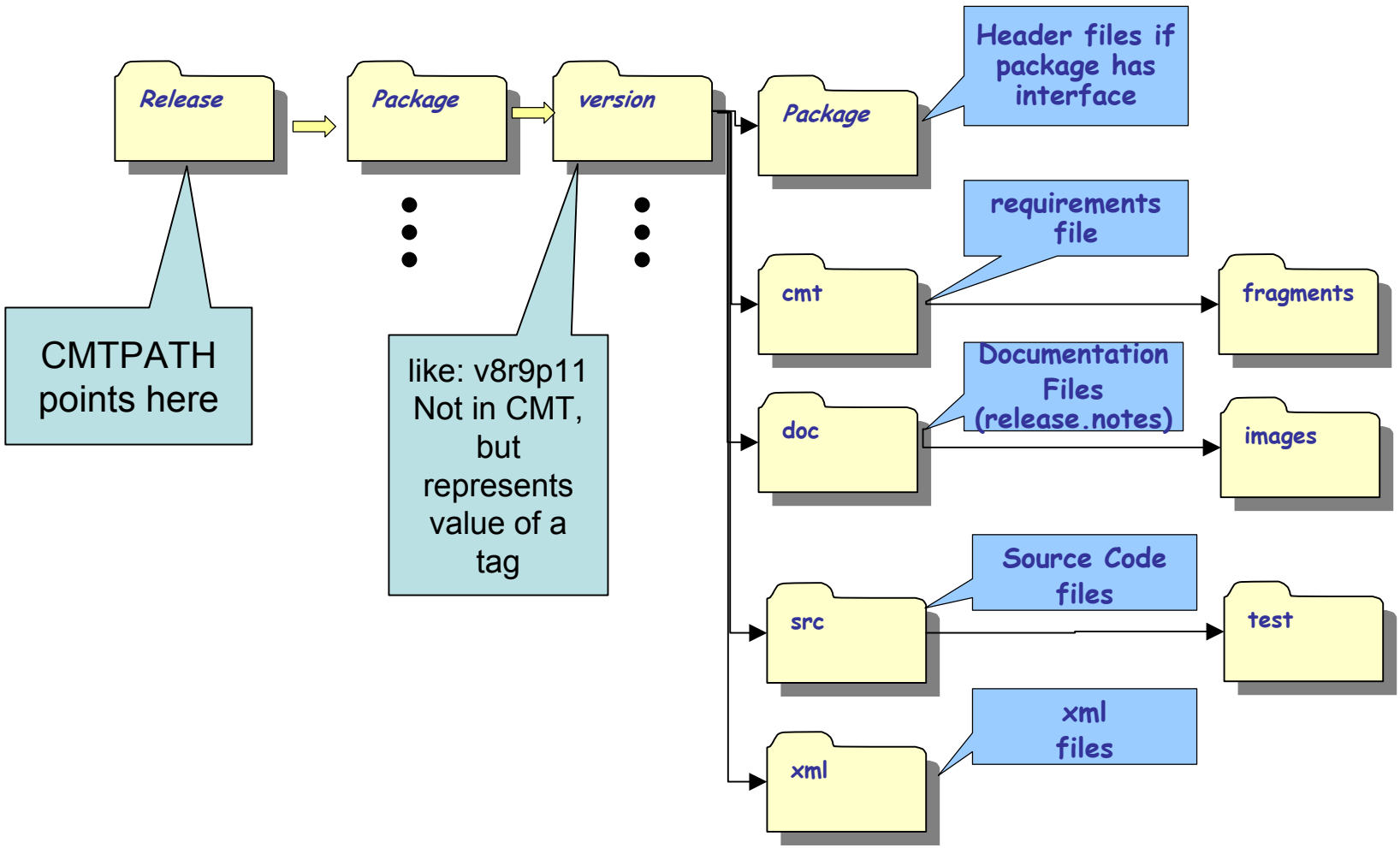
- Introduction: how we build Gaudi and GLAST software
- Gaudi design summary: what we use and how it works
- Glast executable building strategy
- Our primary package: Gleam
- Temporary and Persistent storage: Event and RootIo packages
- GlastSvc: converters and services for Glast



Building Gaudi and Gleam

- We use cvs and *CMT* to manage source and *packages*. Use CMT interface packages for external code
- Mandate imbedded doxygen comments for documentation
- No distinction between GLAST and Gaudi packages: allows tuning requirement files
- Frozen versions:
  - CMT v1r10p20011126
  - Gaudi v9 $\rightarrow$ requires egcs 2.91.66 (VC++ 6.0 on windows)

The Glast Package file layout



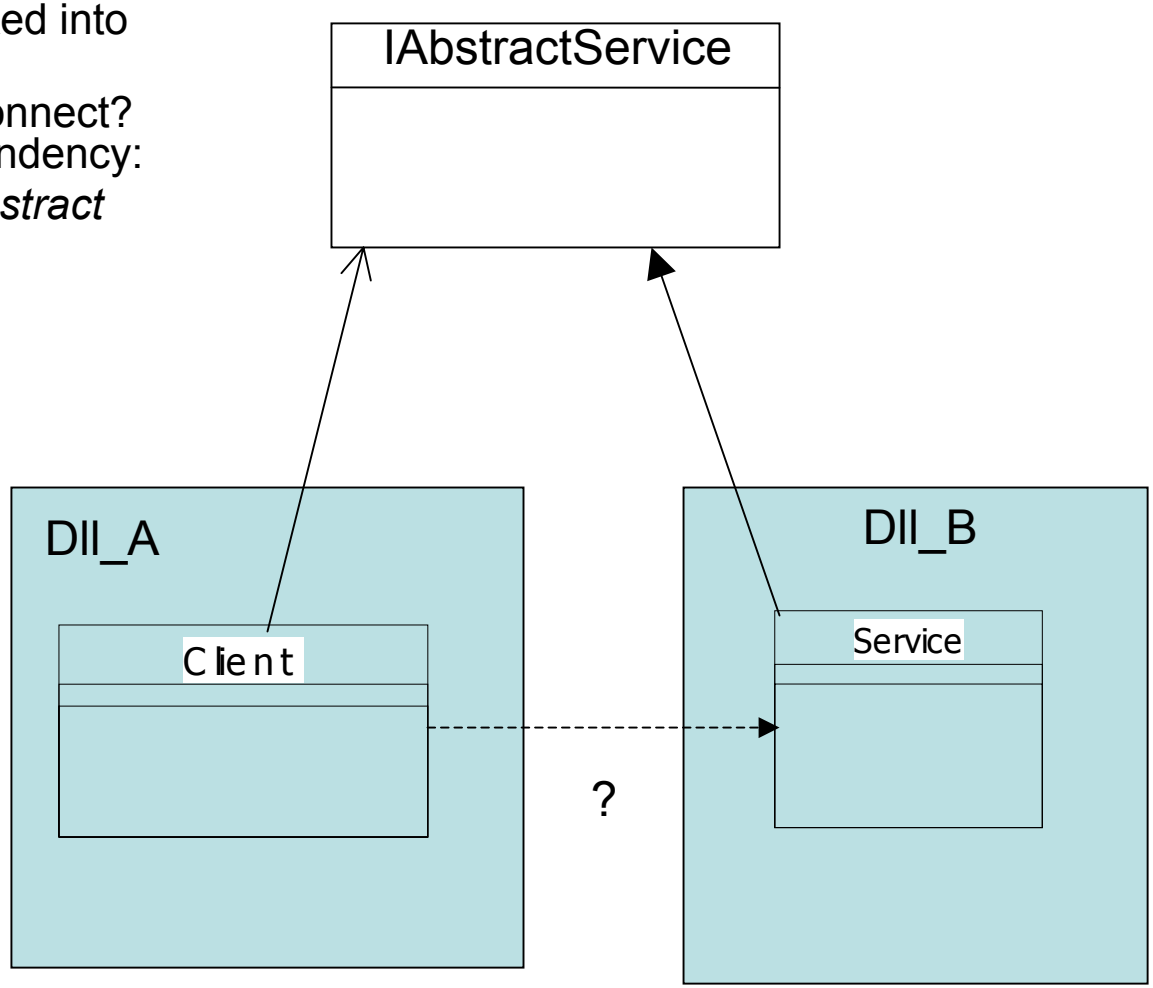
Gaudi design: DLLs, Abstract interfaces

Code compiled, prelinked into shareables (DLL's)

How does the Client connect? Compile time dependency:

must have common *abstract* header.

How about run time?

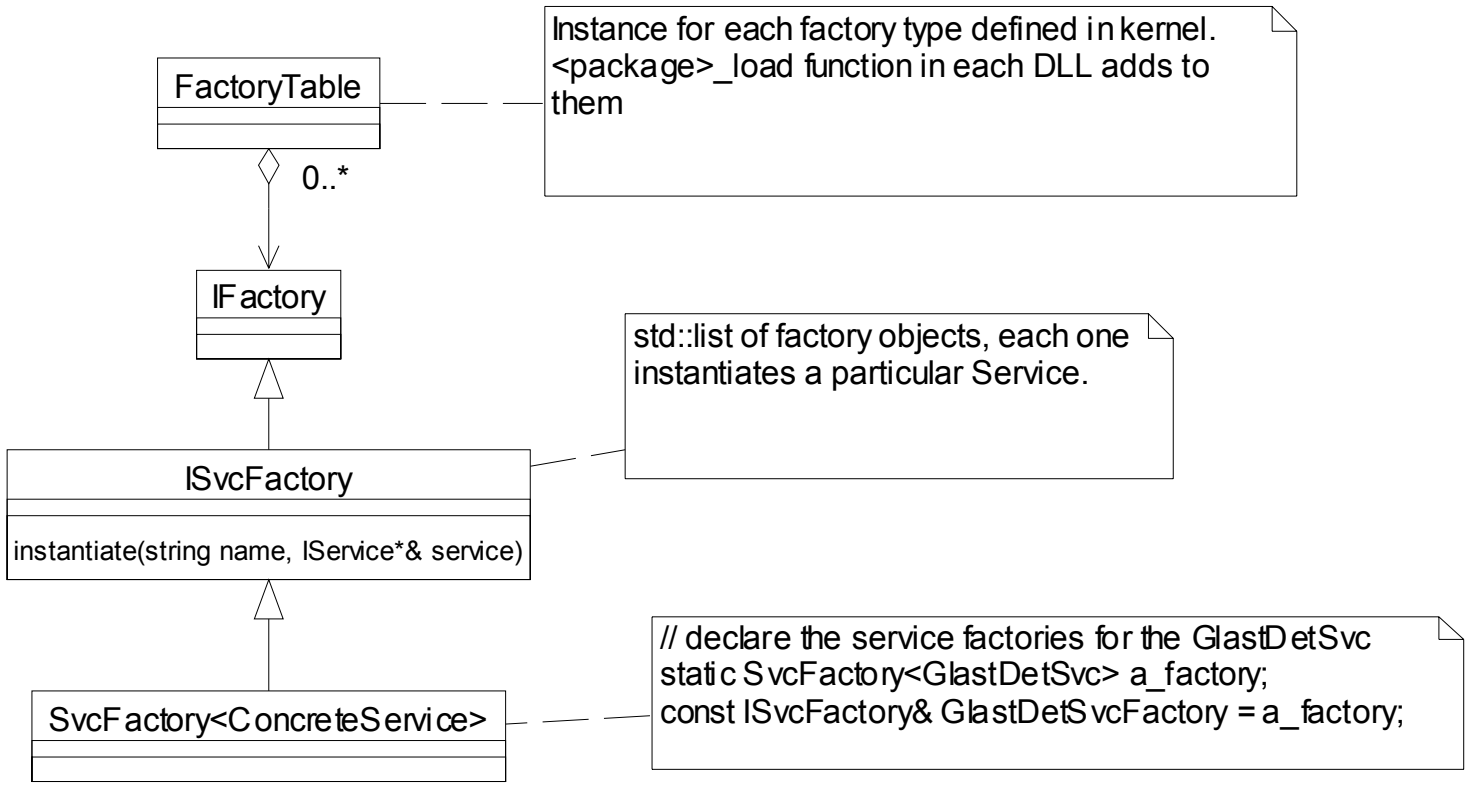


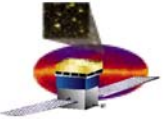
Gaudi design II: Factories

- Runtime: So how does the code in the Client method in DLL_A find out about the object in DLL_B? And how is it instantiated?
  - must happen at runtime.
  - (If one big object, could have used Singlet Pattern)
- Answer: A *FactoryTable*

Gaudi Factories

Factories in Gaudi





Gaudi design: the interface ID

- **ISvcFactory.h:**

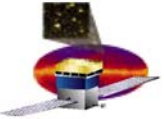
```
// Declaration of the interface ID ( interface id, major version, minor version)
static const InterfaceID IID_ISvcFactory(106, 1 , 0);
class ISvcFactory : virtual public IFactory {
public:
  /// Retrieve interface ID
  static const InterfaceID& interfaceID() { return IID_ISvcFactory; }
[...]
```

- Notes:
  - Two uses of C++ “static” keyword: file-scope defined symbol and class variable.
  - Compiled into the DLL.

- **Service.cpp**

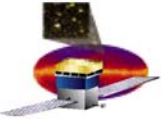
```
//--- IInterface::queryInterface
StatusCode Service::queryInterface(const IID& riid, void** ppvInterface)  {
  if ( IID_IInterface.versionMatch(riid) )  {
    *ppvInterface = (IInterface*)this;
  }
  else if ( IID_IService.versionMatch(riid) )  {
    *ppvInterface = (IService*)this;
  }
  else if ( IID_IProperty.versionMatch(riid) )  {
    *ppvInterface = (IProperty*)this;
  }
  else  {
     return NO_INTERFACE;
  }
  addRef();
  return SUCCESS;
}
```

- Notes:
  - versionMatch requires interface and major major ids the same, minor must be <= local value.



Interfaces and Factories we use

- IAlgorithm (30 concrete)
  - all Algorithms (initialize, execute, finalize)
  - Instantiated by AppManager or Sequencer
- IService (12 concrete)
  - Instantiated by AppManager if in ExtSvc list.
  - Used for geometry, flux, Gui
- IAlgTool (6 interfaces, 20 concrete)
  - Created on demand
- IConverter (2 concrete)
  - Created when found

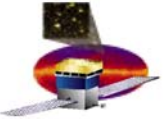


Gaudi execution model

- Executable has **only** Gaudi kernel code, simple main
- Expect CMT setup runs first to create env vars, especially *Package*Shr with paths to all shareables

```
GaudiSvcShr=/u/ek/burnett/myground/Gleam/GaudiSvc/v7r0p1/Linux-i686/libGaudiSvc
```

- Sequence:
  1. Use bootstrap code to load ApplicationMgr from GlastSvc dll
  2. Start basic services (job options, message, histograms, …)
  3. Find and read job options file
  4. Load specified dll's from list in ApplicationMgr.DLLs.
  5. Instantiate all Services in ApplicationMgr.ExtSvc.
  6. Instantiate all Algorithms in ApplicationMgr.TopAlg.
  7. Start event loop, calling list in TopAlg.



Gleam – package managing the GLAST sim/recon program

- Actually two executables, build from code in GlastPolicy and GuiSvc
- By contrast, largest DLLs are G4Generator (12.3 MB) and TkrRecon(10.9 MB).
- Contains joboptions to define several configurations.
- ROOT scripts for detailed tests
- Heavy use of Sequence algorithm
- Special GUI/event display mode, using IRunnable interface

Type	External Symbols	Size (MB)
Gui	1980	2.3
non-Gui	178	0.2

Job options files (in src)

- basicOptions.txt
- guiOptions.txt
- jobOptions.txt
- readdigi.txt
- gamma_1_gev_normal.txt

GLAST LAT

Program Flow: basicOptions.txt

Sequencer/Top	Sequencer/Generator	FluxAlg		
		G4Generator		
	Sequencer/Digitization	TkrSimpleDigiAlg		
		CalDigiAlg		
		AcdDigiAlg		
	Sequencer/EventDisplay			
	Sequencer/Triggered	TriggerAlg		
		Sequencer/Reconstruction	Sequencer/Cal1	CalXtalRecAlg
				CalClustersAlg/first
			Sequencer/Tkr	TkrReconAlg
			Sequencer/Cal2	CalClustersAlg/second
			Sequencer/Acd	AcdReconAlg
		Sequencer/RecoDisplay		
		Sequencer/Output	mcRootWriterAlg	
			digiRootWriterAlg	
			reconRootWriterAlg	

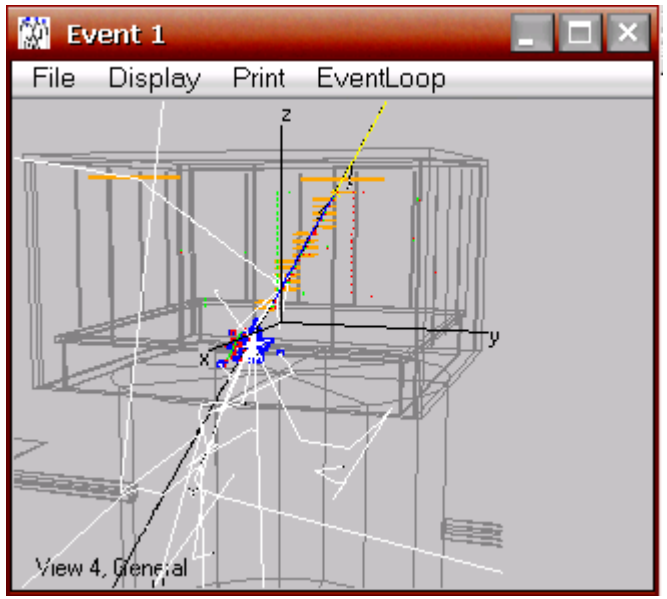
GLAST LAT

GUI/event display configuration: GuiSvc

- Implemented by special joboptions, guiOptions.txt, should be included after basicOptions.

```
ApplicationMgr.ExtSvc += {"GuiSvc"};
ApplicationMgr.Runable = "GuiSvc";
```

- Service with factory in executable (requires special link options).
- Implements *IRunnable* interface to control event loop.
- During initialization, locates all tool factory instances for IGuiTool tools: creates each and have them set up GUI objects. (Currently two: FluxMenu and DetectorDisplay.)



GLAST LAT

The Event and detector model; I/O

- Event model: DataObject classes in Event.
- Detector model: GlastDetSvc Service.
- Converters: EventHeader and MCEvent.
- I/O: ROOT conversion algorithms

Now, to Heather for the details