JSF Overview
- GLD tools -

Akiya Miyamoto, KEK

16-April-2009

ILD Software Workshop

SimTools: package of GLD tools

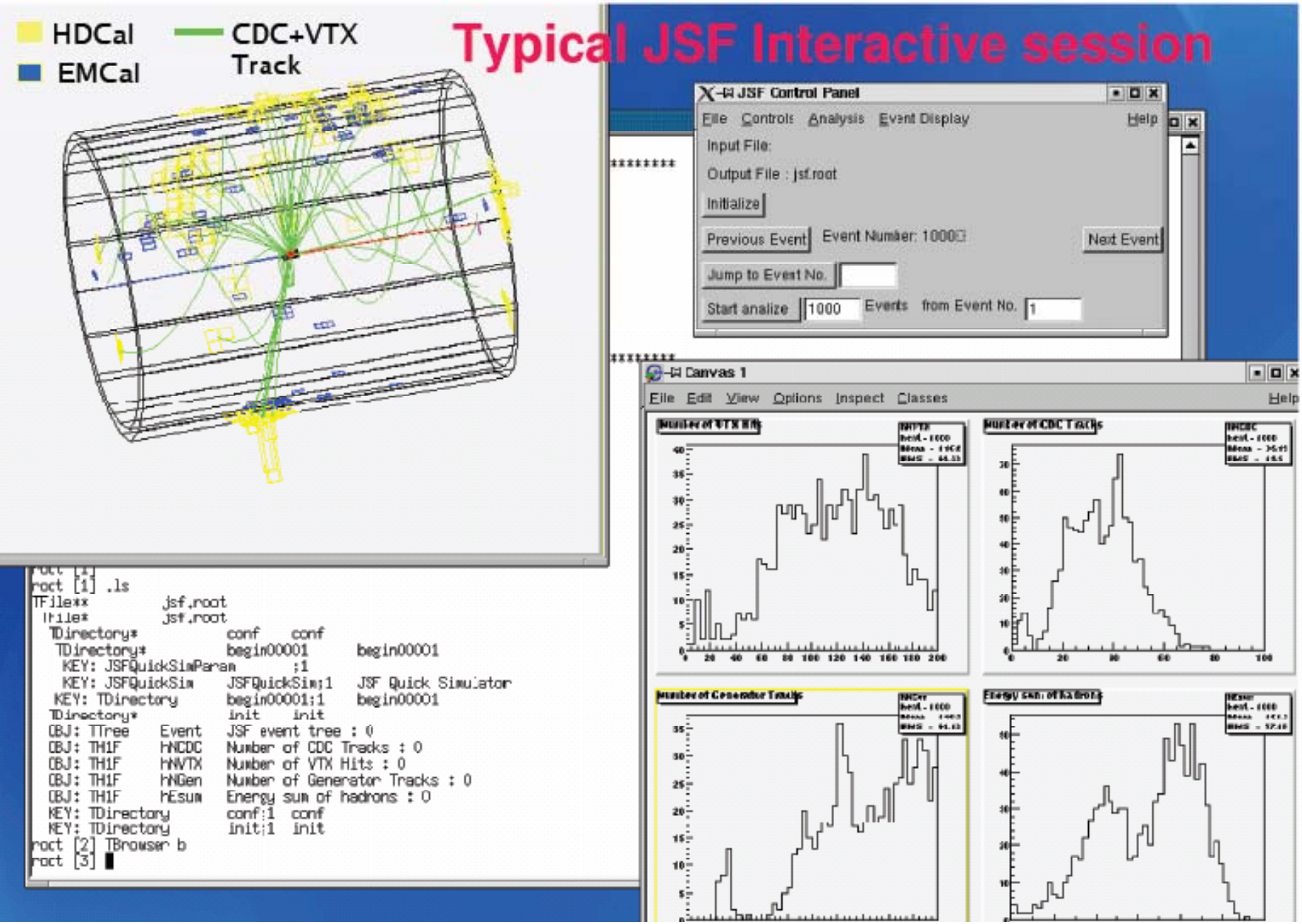
- lcbase : configuration files
- Leda : Analysis tools (Kalman fitter, 4vector and jet findinder utilities)
- jsf : Root-based framework
- lclib : QuickSim and old fortran based utilities
- physsim : Helas-based generator
- Jupiter : Full simulation based on Geant4
- Uranus : Data analysis packages
- Satellites : Data analysis packages for MC data

➢ All packages are kept in the CVS. Accessible from http://jlccvs.kek.jp/

JSF

- Framework: JSF = Root based application
  - All functions based on C++, compiled or through CINT
  - Provides common framework for event generations, detector simulations, analysis, and beam test data analysis
    - JSFModule: Base class of analysis modules
    - JSFEventBuf: Base class of event data
  - Unified framework for interactive and batch job: GUI, event display
    - Loading libraries and creation of objects at run time using ROOT macros
  - A configuration file, jsf.conf, and run time arguments are used to define analysis parameters.
  - Data are stored as root objects; root trees, ntuples, etc
  - Base class for LCIO is provided.
    - Actual implementation depends on data/objects

Typical JSF Interactive session



JSF Kernel

- JSF is a framework for event-by-event data analysis
- Provides a modular framework suitable for analysis consists of several sub-detectors
- Job flow control
  - Job flow is controlled by a class, JSFSteer
  - Analysis modules are inherited from a class, JSFModule
    - Member functions of JSFModule
      Initialize(), BeginRun(..), Process(...), EndRun(), Terminate()

JSF job flow concept

- Create modules
- Jon Initialization
  - Begin Run
    - Event analysis
  - End Run
- Job Termination

A simple example without Macros is prepared in $JSFROOT/example/ZHStudy

JSF Kernel - FileIO

- Object I/O
  - Each modules can save/read their event data as branches of a root tree.
  - Job parameters, histograms, ntuples and private analysis tree can be saved in the same file

- A class, JSFEventBuf, is defined by JSFModule
  - It is used to define branch of a ROOT Tree
    (used to save/get event data)
  - JSFModule ←→JSFEventBuf : 1-to-1 correspondance
  - Information of JSFModule written in a root file is used to define branch for read-in data.

- In a user program,
  - To get pointer to JSFModule objects,
    mod= (JSFModule*) gJSF->FindModule("module_name")
  - To get pointer to JSFEventBuf objects,
    buf=(JSFEventBuf*)mod->EventBuf()

JSF Conponents

- Libraries ($JSFROOT/lib)
  - Pre-compiled C++ classes to build JSF application such as libJSFGenerator.so, libJSFQuickSim.so, …
- Executables (main program) ($JSFROOT/bin)
  - “jsf” command : built with ROOT+libJSF.so
- Macros ($JSFROOT/macro)
  - C++ program is used as Macro thanks to CINT (No need to compile and link)
  - In JSF, Macros are used to set run parameters and provide a simple analysis code, for example,
    - gui.C: Load GUIMainMacro.C and libraries for GUI
    - GUIMainMacro.C: Define a standard set of modules and their parameters
    - UserAnalysis.C : Allow simple user analysis (Hist. def, event analysis, ..)

Parameter file

- All parameters are managed by JSFEnv class
  - In the userprogram, they are obtained by a method,
    JSFEnv::GetValue("Parameter.name",default)

- At run time, paremetercan be changed by three method
  - In a file, jsf.conf
    Parameter.Name: value
    #!argname
    # comments
    ….
  - As a command line argument, like
    $ jsf –argname=value gui.C
  - Through the popup menues of JSF Control Panel
    Each user can add their own menu by a function, UserMenu()

argname is an alias of Parameter.Name
used to parse command line argument

Packages related to JSF

- Packages
  - Included in the release
    - Pythia6.4, Bases/Spring++, ZVTOP, JETNET, BSGEN
    - StdHep read/write interface.
    - Basic LCIO converter

- Provided as separated packages
  - Physsim (Event generators and analysis utilities)
  - LCLIB (QuickSim, Helas)
  - Jupiter (Geant4)
  - Uranus/Satellites
    - Jupiter output to LCIO

Summary

- JSF has been very useful for studies on Linear Collider Physics and Detector .

Backup slides

JSFGeneratorParticle

- Particle informationID,
  Mass, Charge, P, X, DL,Pointers to Mother, 1st_Daughter, NDaughter
- Example
  - jsf/generator
    - using JSFGeneratorParticle
    - EventShape

JSFQuickSim

- Quick Simulator module
  - Detector parameter file
    - $(LCLIBROOT)/simjlc/param/detect7.com-- "JLC-I" Green Book Detector (2 Tesla) , default
    - $(LCLIBROOT)/simjlc/param/jlc3T.com-- "ACFA Report" (3 Tesla)
    - $(LCLIBROOT)/simjlc/param/gld_v1.com-- "GLD_V1" (3 Tesla) (performance needs to be checked.)
  - JSFQuickSimParam : parameter class
  - JLCQuickSim.ParameterFile: env. param.
- Simulator Output data
  - JSFQuickSimBufVTX (+IT), CDC, EMC, HDC, LTKCLTrack

JSFLTKCLTrack

- Information based on "Combined Track Bank"
  - http://www-jlc.kek.jp/subg/offl/lib/docs/cmbtrk/main.html
- Data in class
  - P at closest approach to IP
  - Particle type:

    1=Pure gamma, 2=Gamma in mixed EMC, 3=Pure neutral Hadron,
    4=Hadron in mixed HDC, 5=Pure charged hadron, 6=Unmached Track
    11=Electron candidate, 13=muon candidate
  - Source of information : 100*IHDC + 10*IEMC + ICDC
  - Nsig
  - Pointer to CDC Tracks

Anlib

- ANL4DVector: TLorentz , Lockable
- ANLEventSahpe
  - Using TObjArray of ANL4DVector
  - Calculate Thrust, Oblateness, Major/Minor Axis
- ANLJetFinder
  - base class for Jade, JadeE, Durham jet finder
- ANLJet : ANL4DVector

See examples in $(LEDAROOT)/Anlib/examples