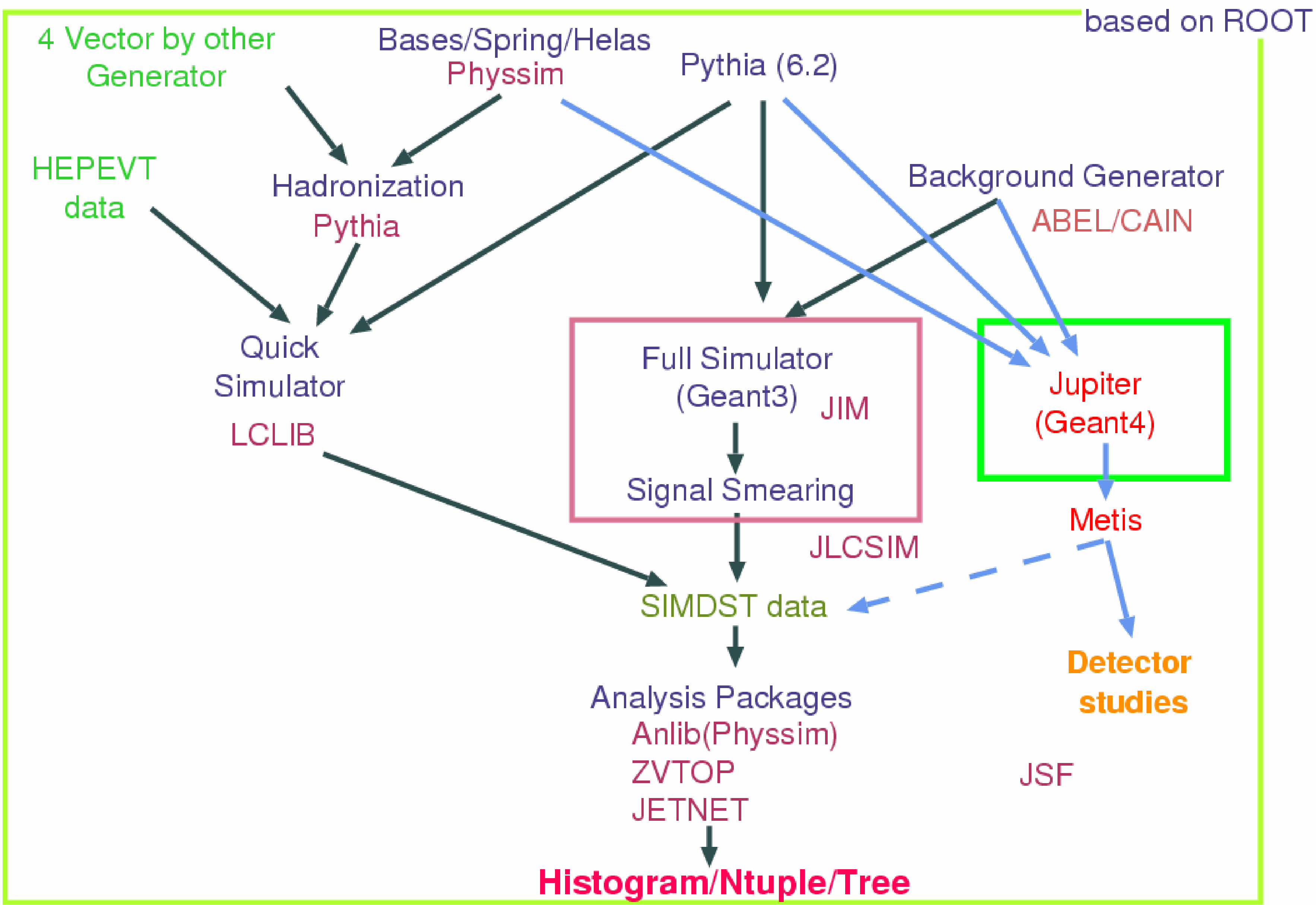
# Introduction to JSF

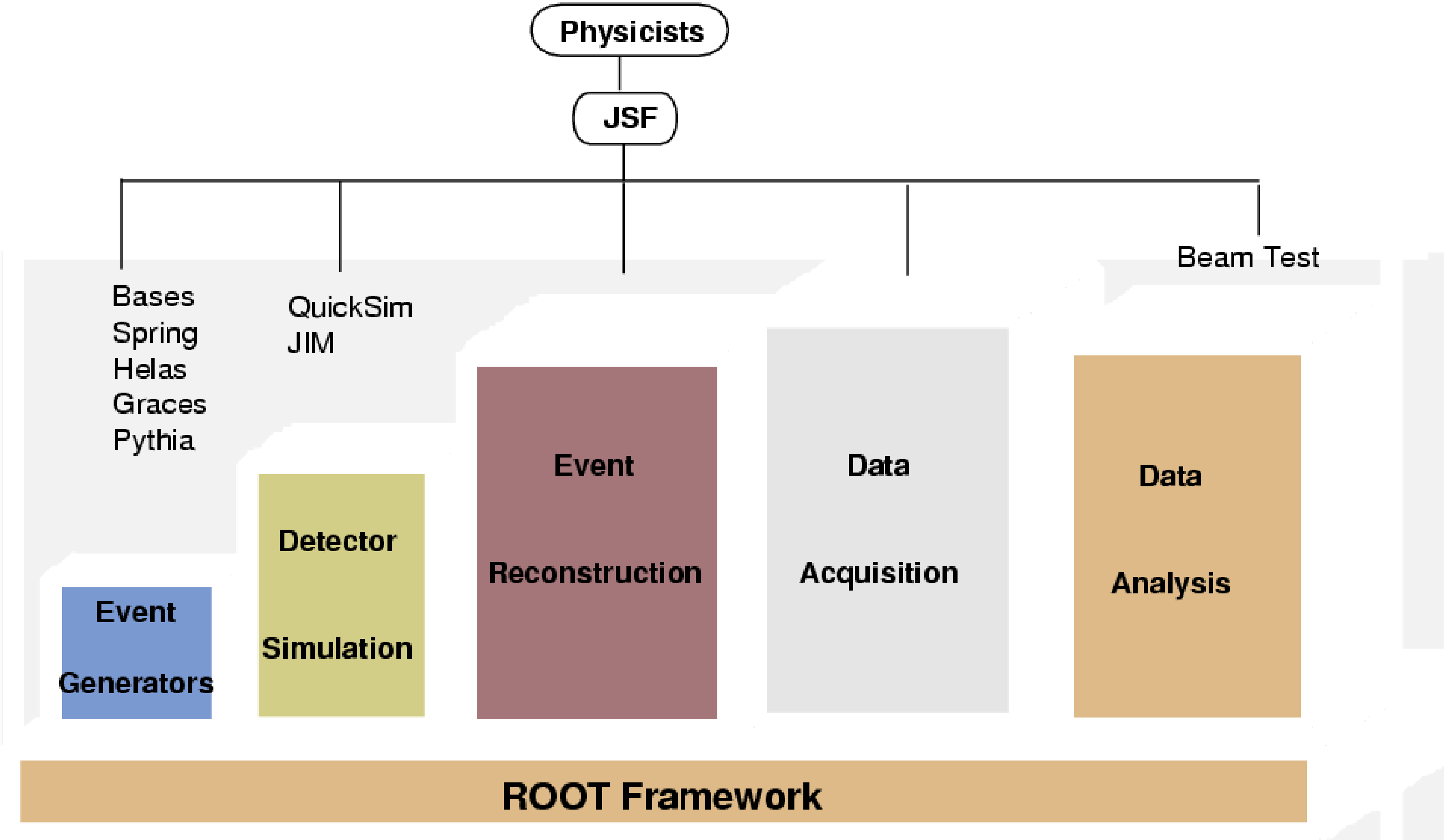
Akiya Miyamoto
KEK

# JSF and related packages



# Root-JSF-QuickSim

JSF is a root based application to provide a common interface to physicists




http://root.cern.ch/ for root
http://acfahep.kek.jp/subg/sim/softs.html for links to JSF, QuickSim, JIM, Physsim, ...

# JSF Features - 1

1. JSF is based on ROOT
   - ☛ User needs to lean just one language, C+

2. JSF provides a framework for modular analyses
   - ☛ Common framework for event generation, detector simulation, and analyses.
   - ☛ Same framework for beam test data analysis

3. Unified framework for interactive and batch jobs
   - ☛ GUI for control of an interactive run
     Histogram and event display packages included
   - ☛ A file similar to .rootrc is used to set parameters
     Default values an be overidden by command line argument at run time.

# JSF Features - 2

1. 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

2. Packages
   1) Included in the release
      - ☛ Pythia6.2, Bases/Spring++, ZVTOP, JETNET, BSGEN
   2) Provided as separated packages
      - ☛ Physsim (Event generators and analysis utilities)
      - ☛ LCLIB (QuickSim, Helas)
      - ☛ JIM (Geant3)
      - ☛ Jupiter (Geant4)

# Get Started

- Set environment variables: JSFROOT

bash
```
$ export JSFROOT=jsf_top_directory
$ export PATH=$PATH:$JSFROOT/bin
```

tcsh
```
$ setenv JSFROOT jsf_top_directory
$ set path=($path $JSFROOT/bin)
```

- Define macro path : ~/.rootrc

```
Unix.*.Root.DynamicPath:  .:$(ROOTSYS)/lib:$(JSFROOT)/lib:$(KFLIBROOT)/lib
Unix.*.Root.MacroPath:    .:$(ROOTSYS)/macros:$(JSFROOT)/macro
```

- Start JSF

```
$ jsf gui.C
```

# Using JSF Control Panel

- Controls menu
  - run mode
  - generator type
  - generator parameters
    - pythia
      - event type
        - zh
  - save parameters
- Next Event button

# UserAnalysis.C

- Example in $JSFROOT/macro/UserAnalysis.C
- Three functions:
  - UserInitialize() : Called at Job initialization
    define Histgrams, etc.
  - UserAnalysis() : Called at each event
    for event analysis
  - DrawHist() : Called to draw histogram

jsf/ex1/UserAnalysis_1.C

# Batch run

$ jsf -b -q --maxevt=100 gui.C

- root option:
  - -b : run without X
  - -q : quit at the end
- jsf option
  - --maxevt=N        : N is number of events

# Set parameters

- In a file, jsf.conf
- Command line arguments
  $ jsf -conf=conf_file  --optionN=valueN .... gui.C
  - conf_file : a parameter file name
  - optionN=valueN: parameter name and its value

Format of jsf.conf

```
Parameter.name : value
#!option_name
# comment-1
# comment-2
```

# Parameter file

All parameters are managed by JSFEnv class

In the user program, they are obtained by a class

JSFEnv::GetEnv(*"Parameter.Name"*, *default*)

At run time, parameter can be changed by three method

1. In a file, jsf.conf

   *Parameter.Name* : value
   #!*argname*
   # Comments
   ......

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

2. As a command line argument, like

   [%] jsf *--argname=value* gui.C

3. By popup menus of JSF Control panel

   PythiaGenerator: Type of process, CM energy, etc
   DebugGenerator: Particle ID, momentum, etc...

Each user can add their own menu by a function, *UserMenu()*

# Concept of JSF run control

General feature of HEP data analysis:

1. Event-by-event analysis
2. Event data consists of several sub-components, analizer of them needs initialization and termination when job or run begins

Standard flow of JSF job

Create *modules*
Job Initialize
Begin run
*Event Analysis*
End Run
Job Termination

- Execution flow are controled by a class JSFSteer.
- One Modules are created, calls of their function are controled by JSFSteer
- Thus, inclusion/exclusion of analysis module is easy.

*All analysis classes must be inherited* from JSFModule and JSFEventBuf

JSFModule : provide functions such as

Initialize(), BeginRun(), Process(), EndRun(), Terminate()

JSFEventBuf : A class to save event data in a ROOT file as a tree

# Access JSFModule and JSFEventBuf

- In scipt
  - JSFSteer *jsf  (defined in gui.C)
    jsf->GetEventNumber();
    JSFXXX *mod=(JSFXXX*)jsf->FindModule("JSFXXX");
    JSFXXXBuf *buf=(JSFXXXBuf*)mod->EventBuf();
- In compiled code,
  - JSFSteer *gJSF  (defined in JSFSteer.h)

# Build compiled library

- buildjsf command

jsf/ex2/buildjsf

# JSF Components

1. Libraries

Pre-compiled C++ classes to build JSF application
Such as libJSF.a, libJSFQuickSim.a, ...

2. Executables ( = jsf )

Root libraries + JSF libraries + Fortran libraries ⇨ jsf application

Fortran libaries = lclib, jlcsim, cernlib, ...

3. Macro

C++ program is used as Macro thanks to CINT, no need to compile and link

Macro can be used to set run parameters without compile/link

In the jsf distribution, gui.C, GUIMainMacro.C, and UserAnalysis.C are included as an analysis example

# JSFGenerator

- JSFGenerator
- PythiaGenerator
- JSFBases - JSFSpring - JSFHadronizer
- JSFMEGenerator - JSFSHGenerator
  JSFReadMEGenerator - JSFPythiaHadronizer

# PythiaGenerator

- Parameters
  - Process : ZH, ZZ, WW, enW, eeZ, gammaZ
  - BeamStrahlung
  - Decay: Z, W, H
- InitPythia.C

# JSFGeneratorParticle

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

# Quick Simulator

Detectr components:

VTX, IMT, CDC, CAL are included.
Detector parameters (resolution,geometry, etc) can be changed be a parameter file

Signal generation:

Particles are swimmed through VTX, IMT, CDC, and CAL.

Particles are smeared by multiple scattering by matterials such as VTX, IMT, etc.

VTX and CDC

Equally spaced N sampling with given $\sigma_{r\phi}$ and $\sigma_Z$ in solenoid field
5 dimensional error matric of the track parameter are smeared including the effect of the multiple scattering due to chamber gas.
VTX and CDC parameters are then averaged to get combined helix parameter

IMT Just create smeared hit points

CAL: Particle energy is spread laterally by $f(x) = a_1 \exp(-|x|/\lambda_1) + a_2 \exp(-|x|/\lambda_2)$

Generated energy is distributed to each countes after smearing according to the resolution.

$e$ and $\gamma$ : Deposite energy only in EM calorimter
hadrons : Deposite energy only in HD calorimter
$\mu$ : No energy deposite in calorimeters

# Crosssection of JLC detector

Detector size   8m(ϕ)x7.1m(z)

Magnet   3 tesla

Muon   Number of superlayers : 6

Calorimeter   Lead/Scint., compensated

EM Cal:   Thickness : $27.1X_0$

Segmentation : 4x4 (cm$^2$)

Radius(barrel) : 1.6 ~ 1.86m

$\sigma_E/E(\%) = 15\%/\sqrt{E} \oplus 1\%$

HD Cal:   Thickness : 6.5λ

Segmentation : 12x12 (cm$^2$)

Radius(barrel) : 1.86 ~ 3.4 m

$\sigma_E/E(\%) = 40\%/\sqrt{E} \oplus 2\%$

Central Drift Chamber(CDC)

Small cell jet chamber

Number of sampling : 50

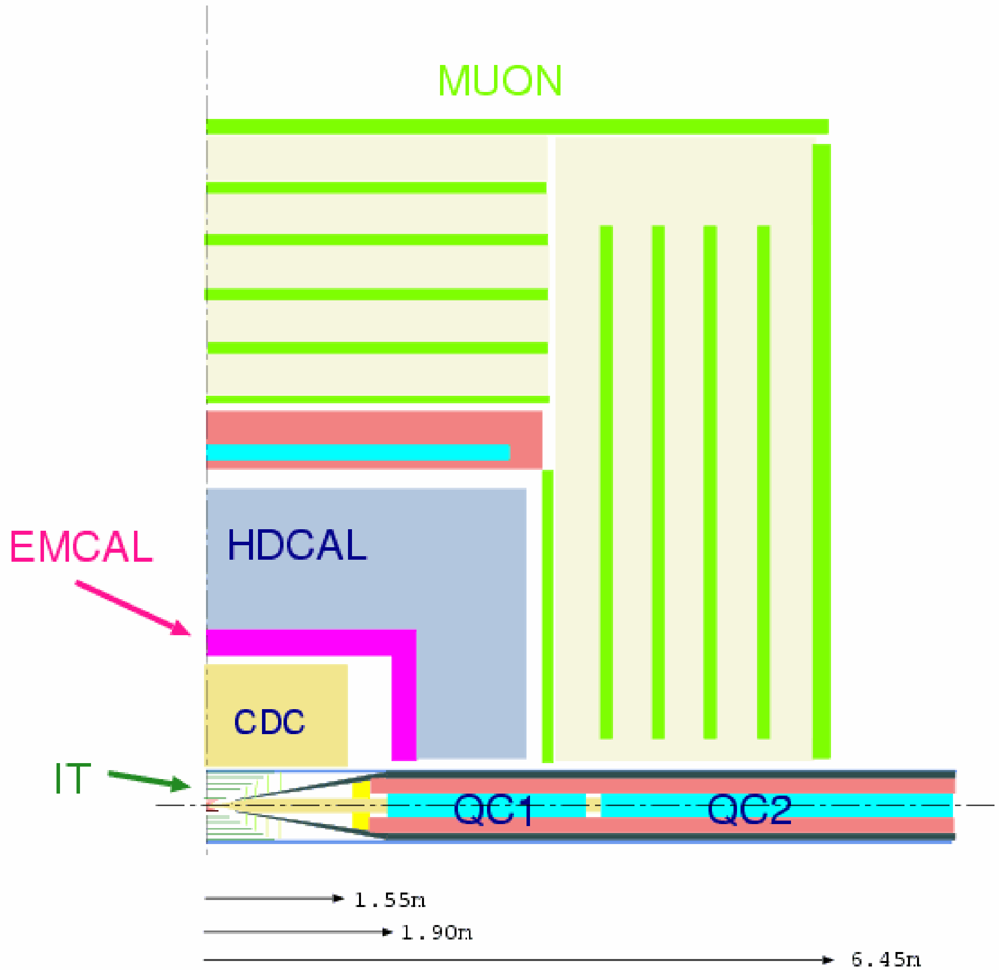
Position: r=0.45 to 1.55m, |Z|<1.55m

Position Resolution: $\sigma_{r\varphi} = 100\mu m(\mathrm{axial})$

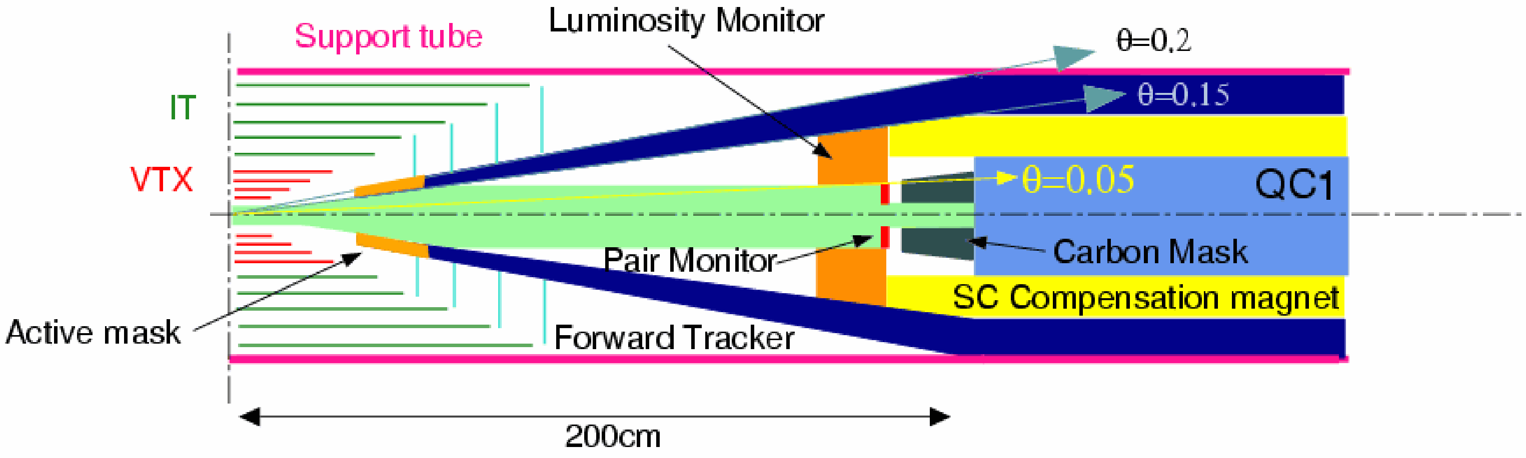
$\sigma_z = 1mm(\mathrm{stereo})$

Momentum Resolution:

$$\sigma_{p_t}/p_t = 3 \times 10^{-4} p_t \oplus 1 \times 10^{-3}$$
$$\sigma p_t/p_t = 0.9 \times 10^{-4} p_t \oplus 1 \times 10^{-3}(\mathrm{w.vtx})$$



# Detector system near IP




Intermediate Tracker(IT): Silicon strip/pixel

Geometry: 5 layers, r=9cm to 37cm, $|\cos\theta|<0.9$

Position resolution: $\sigma = 40\mu m$

Vertex Detector (VTX) : CCD

Position: 4 layers, r=2.4cm to 6cm, $|\cos\theta|<0.9$

Position resolution: $\sigma = 4\mu m$

Impact parameter resolution:

$$\delta = 3 \oplus 24/p^{3/2} sin^{3/2}\theta (\mu m)$$

Forward Tracker(FT) : Silicon pixel/strip

4 layers silicon

Coverage: $0.90 < |\cos\theta| < 0.98$

Pair monitor

Silicon 3D detector to monitor beam property

Luminosity monitor

W + Si pad, $42.9X_0$

Coverage: $0.05 < \theta < 0.15$(radian)

Segmentaion: radial 32, azimuthal 16

Active mask

8 layers of W + Si pad

Coverage: $0.15<\theta<0.20$(radian)

Segmentaion: radial 8-10, azimuthal 32

# 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)
  - JSFQuickSimParam : parameter class
  - JLCQuickSim.ParameterFile: env. param.
- Simulator Output data
  - JSFQuickSimBuf
    VTX (+IT), CDC, EMC, HDC, LTKCLTrack

# JSFSIMDST(Buf)

- The format agreed among ACFA group.
- JSFQuickSIM + JSFGenerator
- Same information can be written to a file accesible by FORTRAN program.

# Classes for QuickSim Output
# JSFSIMDSTBuf

Important Member functions:

```
Int_t GetNLTKCLTracks();
Int_t GetNCDCTracks();
Int_t GetNVTXHits();
Int_t GetNEMCHits();
Int_t GetNHDCHits();
Int_t GetNSMHits();
Int_t GetNGeneratorParticles();

TObjArray *GetLTKCLTracks();   // Pointers to LTKCLTracks objects array
TClonesArray *GetCDCTracks();   // Pointers to CDCTracks object array
TClonesArray *GetVTXHits();       // Pointers to VTXhits object array
TClonesArray *GetEMCHits();      // Pointers to EMhits object array
TClonesArray *GetHDCHits();      // Pointers to HDhits object array
TClonesArray *GetSMHits();        // Pointers to SMhits object array
TClonesArray *GetGeneratorParticles();  // Pointers to GeneratorParticle objects array
```

# 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

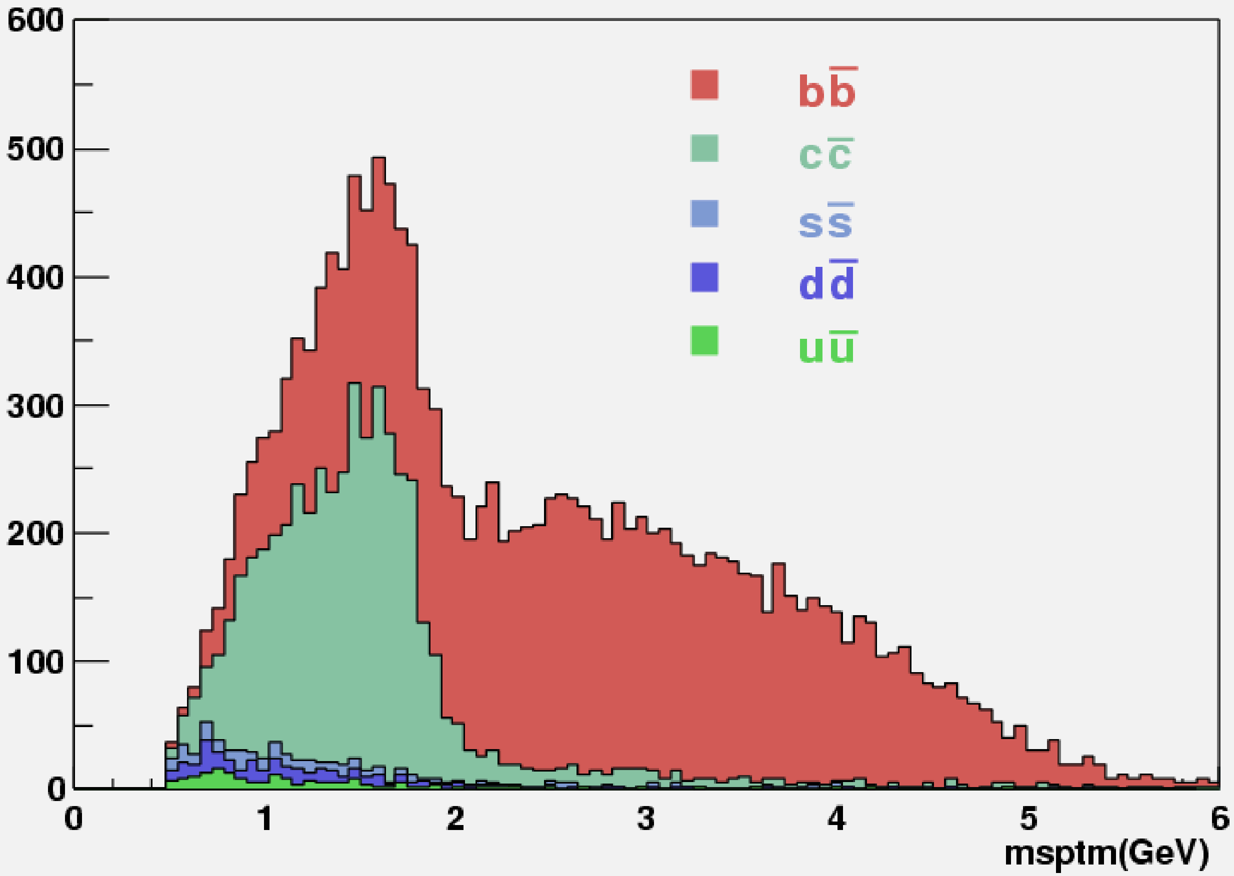
See examples in $(KFLIBROOT)/Anlib/examples

# Sample Analysis

- jsf/quicksim
  - e+e- → ZZ event
  - Use JSFLTKCLTrack class
  - Use ANLJetFinder
- jsf/jetanal
  - e+e- → ZH event ( Z→qq only )
  - Compiled program to create TTree, JetEvent
  - JetEvent → JSFJet → JSFVertex
  - Example to access by Tree->Draw()
  - Example to access by your own event loop

# Pt corrected Vertex Mass (MSPTM) distribution

for $|\cos\theta_{jet}| < 0.8$ and decay length > 300μm

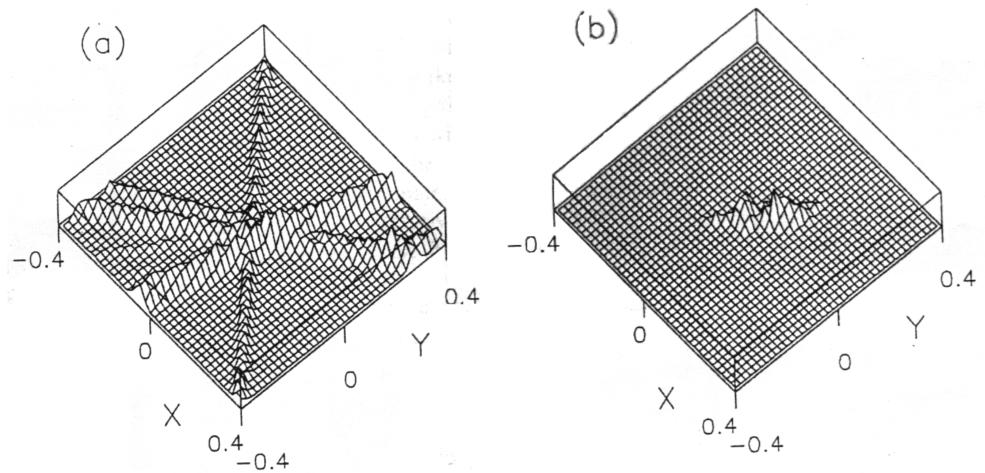



LCWS2000 A.Miyamoto

# ZVTOP vertex tagging

## Topological vertex finding

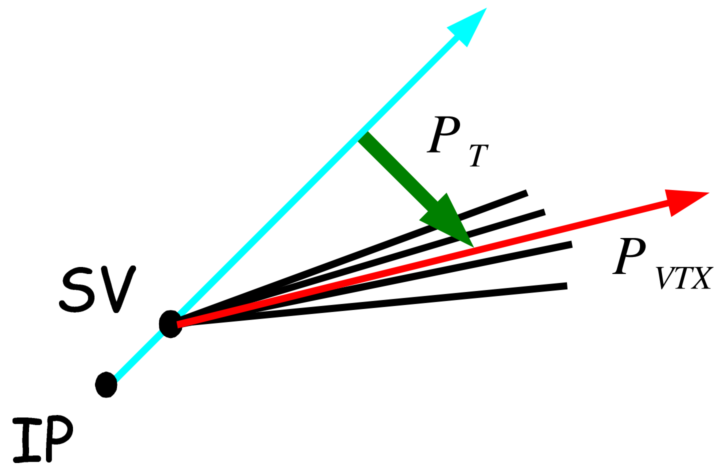
Define a tune of probability along the track trajectory and find points where trajectory overlapps




## Mass tagging

Use $p_t$ corrected vertex mass

$$M_{corr} = \sqrt{M_{VTX}^2 + P_T^2} + |P_T|$$



# Package Install

- ROOT
  - http://www-jlc.kek.jp/~miyamoto/linux/redhat/rh9.0/etc
    root-3.05.05.tar.gz
  - Compiled on RedHat 9.0
  - Package location is defined by ROOTSYS
  - run_config is used for configure
- CERNLIB for RedHat 8/9
  - http://www-jlc.kek.jp/~miyamoto/linux/redhat/rh8.0/etc
    cern-2002-bin/include/lib/share.tar.gz
- LCLIB, Physsim, JSF
  - Get from JLCCVS.KEK.JP

# JLCCVS

- Latest packages are available at jlccvs.kek.jp.
- How to get:
  $ cvs -d :pserver:anonymous@jlccvs.kek.jp/home/cvs/soft login <RETURN>
  Password: <RETURN>
  $ cvs -d :pserver:anonymous@jlccvs.kek.jp:/home/cvs/soft co jsf <RETURN>
- Update
  $ cvs update
- Web interface to see a code history
  http://jlccvs.kek.jp/cgi-bin/cvsweb.cgi/jsf/

# Build Packages

- Environment parameters
  - LCLIBROOT,  KFLIBROOT, JSFROOT
  - Set PATH, etc.  ref.  setup.bash

- LCLIB
  - $ cd $LCLIBROOT
  - $ ./configure  --pythia-major-version 6  --pythia-lib-dir $(JSFROOT)/lib --pythia-lib-name Pythia6
  - $ make install

- JSF
  - $ cd $JSFROOT
  - download pythia source file and save it in share/pythia directory
  - edit cnfig/configure.in  ( set Pythia version number )
  - $ make install    or   make fullinstall

- KFLIBROOT
  - $ cd $KFLIBROOT/Anlib/src
  - $ xmkmf -a
  - $ make
  - $ make install

- ~/.rootrc
  - Unix.*.Root.DynamicPath:  .:$(ROOTSYS)/lib:$(JSFROOT)/lib:$(KFLIBROOT)/lib
  - Unix.*.Root.MacroPath:      .:$(ROOTSYS)/macro:$(JSFROOT)/macro

- then ready to run   jsf gui.C

# Information on Web

- Home page of ACFA-Sim group
  http://acfahep.kek.jp/subg/sim
- Mailing list:   acfa-sim@acfahep.kek.jp
  - JSF update information
- JSF
  - http://www-jlc.kek.jp/subg/offl/jsf
  - Links to lclib, physsim, and documents in this page