Monte Carlo Generators
I. Hinchliffe

- General Remarks
- HepMC data record
- Integration into new Framwork
  - Sequence Diagram for Monte Carlo Generation and Simulation
  - Information in the Transient Store
  - Classes in the Framework
    - ∗ GenModule Base Class
    - ∗ Generator Specific Subclasses
    - ∗ Additional Classes that are Required

General Remarks

Each generator exists as an external package
/afs/cern.ch/atlas/offline/external/
– Aim to share work by having one person per generator/package.

Warning obsolete versions in /afs/cern.ch/atlas/software/cvs/offline

- General Purpose Herwig, ISAJET, Pythia
- Special Purpose PHOJET
- Special packages, *e.g* EvtGen (meeting on Friday at 12:30 pm)

HepMC used as interface between generator and simulation
and also between generators and decay packages

Need StdHep for old fortran generators (also an external package)
These will exist for several years at least
Herwig++ 2003?, Pytha7 2000 (limited functionality)

HepMC

Matt Dobbs and Jorgen Beck Hansen

Intended as a replacement for HEPEVT common

- Note StdHep++ (4.08) interface is similar to FORTRAN – not really adequate

Supports modularization of Generators, physical model

Event is described as a generic tree stucture with particles and vertices

8 Classes + 6 Utility classes (IO mainly)

Can store spin density matrices (if needed)

Iterators for navigation

Depends on STL and CLHEP

Has its own Particle data service, can be replaced

Currently available in /afs/cern.ch/atlas/software/cvs/offline/Simulation/HepM

Will other experiments use this?

ATLAS should ask GEANT-4 to use it.

General Package Structure

```
-------------------------------
| GeneratorEvent              |
-------------------------------
| signal_process_id: int      |
| event_number: int           | 1
| random_states: vector(int)  |_____ (association but no direct
| weights: vector(double)     |    |  navigability. To find ordinary
-------------------------------    |  vertices in an event you
| particle_iterator:*Particle |    |  navigate by means of a primary
| vertex_iterator:*Vertex     |    |  vertex and its "tree".)
-------------------------------    |
          | 1                      |
          |                        |
         \ / primary vertex        | ordinary vertex
          .  1..*                  | 0..*
        ---------------------------|-----------
        | Vertex                              |     0..1    end
        --------------------------------------- <--------  vertex
        | position: HepLorentzVector = 0      |         |
        | id: int                             |         |
        | weights: vector(double)             |         |
        ---------------------------------------         |
        | particle_iterator(Range): *Particle |         |
        | vertex_iterator(Range): *Vertex     |         |
        | check_***_conservation():bool       |         |
        --------------------------------------|         |
               ^  0..1                                 /
              / \ production vertex                   /
               |                                     /
              \ / outgoing particles                |
               .  1..*                              |
         ------------------------------------ 1..*  |  incoming
         | Particle                         |  /    |  particles
         ------------------------------------ <------
         | momentum: HepLorentzVector       |  \
       * | flow:  Flow = map<int,int>       |
 --------| polarization: Polarization = 0,0 |
 |       | status: int                      |
 |       ------------------------------------
 |
 |                                            -------------------------------
 |    --------------------------------        | ParticleDataTable           |
 |    | ParticleData                 |        -------------------------------
 |    --------------------------------        | description: string         |
 |    | name: string                 |0..*    -------------------------------
 \--->| pdg_id: int                  | /      | find(): *ParticleData       |
    1 | charge: double               |<-------| insert(ParticleData*): bool |
      | mass: double                 | \  0..*| erase(id): bool             |
      | clifetime: double            |        | erase(ParticleData*): bool  |
      | spin: double                 |        | delete_all():               |
      --------------------------------        | iterator                    |
      | is_hadron/EM/...(): bool     |        | const_iterator              |
      --------------------------------        -------------------------------
```

An aside to the Management

I am having a very hard time finding people to do the maintenence tasks.

Each job is small, two people could do them all

No sticks or carrots to induce people to do this uninteresting service work

Result

Private copies everywhere, work is being done several times over

ATLAS is still using pythia 5.7 (1997), current version is 6.140

Packages become orphans; *e.g.* what happens to HepMC when Dobbs graduates?

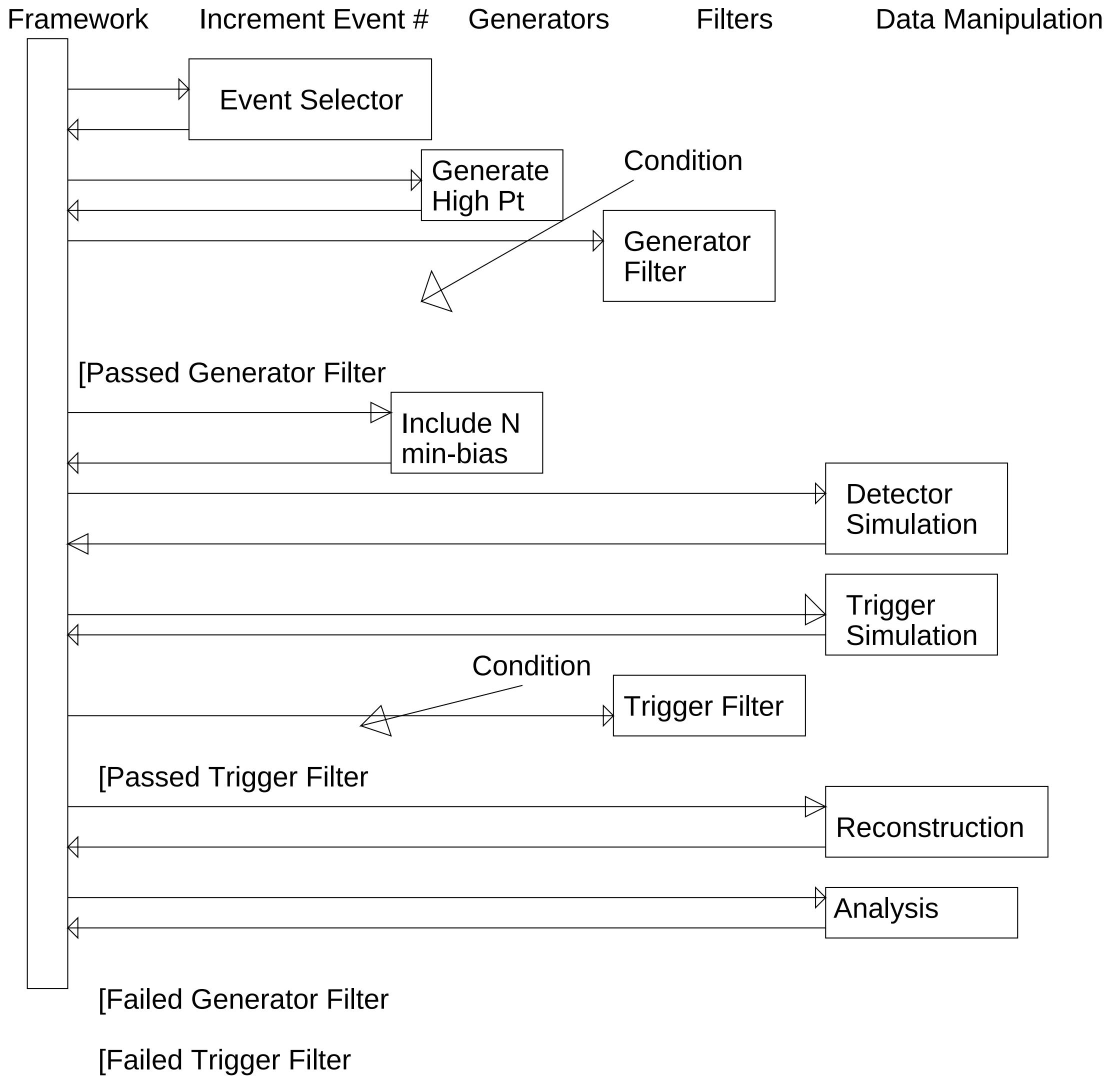
Integration into new Framework

M Shapiro and IH

General Requirements

- Output from all generators in common format
- Use one generator for High P_t process and another for MinBias
- Read either High P_t or MinBias events from pre-existing files OR generate the events on the fly
- Use same interface for full or fast simulation
- Stop at any stage in Generation/Simulation/Analysis chain, write output and then continue in separate job
- Add selection modules (filters) to stop event processing as early as possible if event fails to meet requirements

- Set Generator parameters at run-time
- Optionally write of parts of GeneratorEvent (requirement on convertor)



Framework

Increment Event #

Generators

Filters

Data Manipulation

Event Selector

Generate High Pt

Condition

Generator Filter

[Passed Generator Filter

Include N min-bias

Detector Simulation

Trigger Simulation

Condition

Trigger Filter

[Passed Trigger Filter

Reconstruction

Analysis

[Failed Generator Filter

[Failed Trigger Filter

Sequence Diagram for single physics event

Base Class for Generator Modules

- Base class GenModule implements common functionality:
  - Access HepMC ParticleTable
  - Instantiate CLHEP Random Engine
  - Throw Poisson (if required) for Number of Events
  - Call Generator (see below)
  - Load event into Transient Store
- Provides hooks for child class (virtual methods):
  - genInitialize() [Once at start of job]
  - genFinalize() [Once at end of job]
  - callGenerator() [Every event]
  - fillEvt(GeneratorEvent* evt) [Every event]

Adding Monte Carlo Information to Transient Store

- HepMC defines generator independent description of event
- In order to add to Transient Store, define a class

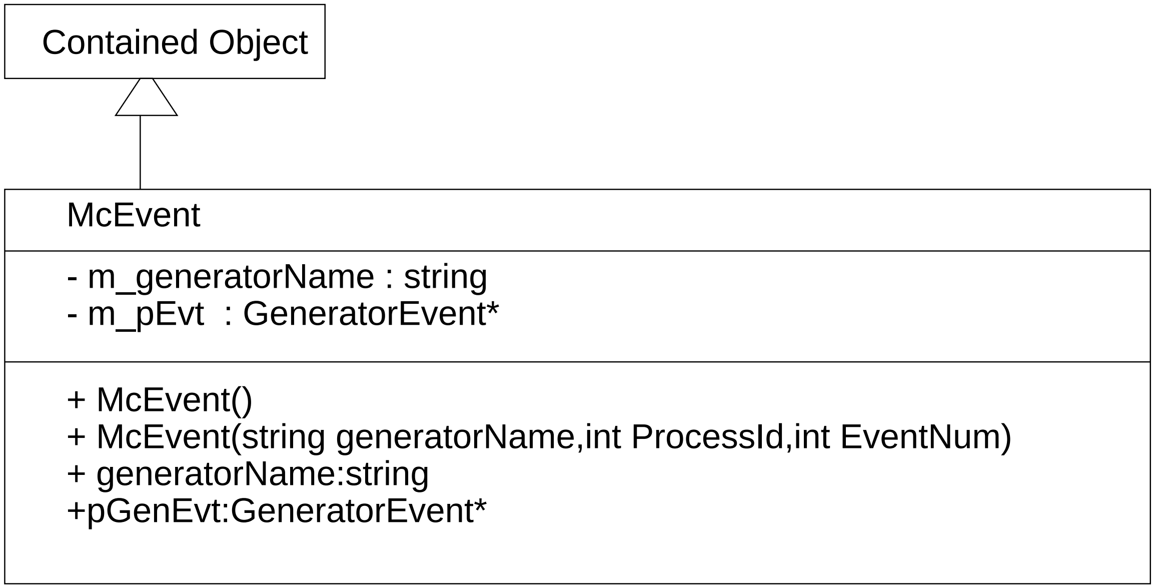
  McEvent:: public ContainedObject

  that contains generator name and HepMC::GeneratorEvent
- Since several McEvents within a given physics event (hard scatter plus N min bias):

  typedef ObjectVector<McEvent> McEventCollection

  Interface is identical to STL vector

McEvent Class Diagram



Contained Object

McEvent

- m_generatorName : string
- m_pEvt : GeneratorEvent*

+ McEvent()
+ McEvent(string generatorName,int ProcessId,int EventNum)
+ generatorName:string
+pGenEvt:GeneratorEvent*

Generator Specific SubClasses

- Status:
  - EventService for generating events (instead of reading file) In tar ball distributed this week
  - Single Particle Gun (Done)
  - Isajet (Done)
  - PhoJet (Almost Done)
  - Pythia Work needed for preliminary version of parameter setting
    - Need Interface to pass parameters from Framework
    - Get parameters via JobOptions service (Interactive job options will simplify this)
    - Pythia7 and herwig++ will use file for input *a la* ISAJET
    - Estimate 15 June for delivery
  - Herwig – September

  - Generator Level Filter Example June 1
  - Persistify HepMC (or parts thereof) $>$ 1 October

- Additional Actions:
  - Decide locate of interfaces in CVS
    /cvs/offline/GeneratorInterface ??? Move HepMC here ???
  - Integrate into releases.
  - HepMcParticleTable Service
    Requires policy decision (GEANT-4 service???)

Open Issues

- Random Number Mangement and Storage:
  - How do we store generator seeds?
    (Each Fortran generator uses different scheme)
  - Do we want a general CLHEP service for C++ generators (including seed storage)?
- Integration of Special Purpose Decay Monte Carlos:
  Are these called as sub-Modules under the GenModule class?
  - Tauola
  - Evtgen – Note there is a meeting regarding EvtGen on Friday

- What should we do with K_S and Λ?
  - Stable, decay at origin, or decay with finite lifetime??
  - Decay in generator with proper lifetime, simulator ignores long-lived decays if necessary
  - Straightforward with HepMC.