HepPDT and HepMC

MC4LHC
Lynn Garren
July 17, 2006

Where to find HepMC and HepPDT

- CLHEP
  - http://savannah.cern.ch/projects/clhep/
- LCG external packages
  - http://savannah.cern.ch/projects/hepmc/
  - http://savannah.cern.ch/projects/heppdt/
- Matt Dobbs
  - http://mdobbs.web.cern.ch/mdobbs/HepMC/
- FNAL web pages
  - http://cepa.fnal.gov/psm/heppdt/
  - http://cepa.fnal.gov/psm/HepPID/
- Discussions forum: LCG generator meetings

CLHEP versus LCG

- For some time, two version of HepMC
  - Matt Dobbs – used by ATLAS
  - CLHEP – used by CMS
- After discussions among experiments beginning 2003, there was an agreement to use the Matt Dobb’s version and have it supported as an LCG external package
- Single point of maintenance and development
- FNAL agreed to provide support
  - Both HepPDT and HepMC move to LCG
- No official mention of this agreement to CLHEP community

HepMC understandings

- Experiments
  - CMS most affected
    - Agreed to change code so that everyone would be using the same version.
    - Expectation that they can request necessary changes.
  - ATLAS and LHCb expect to migrate
- Monte Carlo Generators
  - Some happily using CLHEP HepMC
    - Unaware of proposed changes
  - Some using only their own code

What’s happening in CLHEP

- HepMC and HepPDT essentially frozen
- Bug fixes
- Status could change after this workshop
- CLHEP support has been waning
- Status should be clarified
  - Add appropriate notification to CLHEP website

LCG HepPDT

- Decay factory removed
  - Useful capability, BUT
  - No one used it
  - Everyone complained about the templates
- Two libraries
  - HepPDT
    - ParticleDataTable
  - HepPID
    - Free function translation methods
    - Can use these without any other part of HepPDT

Changes to HepMC

- 1.26.01
  - Last release from Matt (Sept. 2005)
  - libHepMC - core C++ code
  - libHepMCfio - Fortran IO code
- 1.27.02
  - HeavyIon class
    - Added after thorough discussion
    - First request from CMS
  - Bug fixes for g++ 4.x
- Request for Pythia-like event printout
- Proposed 2.00
  - Use MathCore vectors instead of CLHEP Vector
  - Much controversy

IO_AsciiParticles

- Proposed class for Pythia-like event output
- Also contains variable output accuracy
- Author: Mikhail Kirsanov
- Use instead of, or along with, IO_Ascii
- Discussed on project-lcg-simu@cern.ch
  - Positive reception
  - Non-controversial
- Additional functionality
  - Proposed for HepMC 1.28

Vectors

- CLHEP Vector
  - much maligned
  - used nearly everywhere
- MathCore GenVector
  - intended as replacement for CLHEP Vector
  - MathCore can be built as part of ROOT
  - MathCore can be built as a standalone package
    - No ROOT dictionary

CLHEP Headers in HepMC

- CLHEP/Vector/LorentzVector.h
    - HepLorentzVector
        - momentum 4 vector
        - position 4 vector
- CLHEP/Geometry/Point3D.h
    - HepPoint3D
        - position
- CLHEP/Geometry/Normal3D.h
    - HepNormal3D
        - unit normal vector
- Clear replacements for all of these

MathCore GenVector

- CMS requests that HepMC use GenVector
  - ROOT IO
  - Consistency with other code
  - Avoid user confusion
- Geant4 uses CLHEP Vector
  - May change - no commitment or timetable
  - May choose to become owners of CLHEP
- Herwig++ uses CLHEP
- Some function calls differ

CMS Request

- Use GenVector in HepMC
- The main advantage is that - apart from being very flexible when you make them persistent - they allow easy root browsing of HepMC events, something that is not possible with CLHEP vectors.
- Also, CMS has decided to drop CLHEP completely (as soon as an alternative random number package exists)

Response from Herwig++ and ThePEG authors

- Within the C++ event generator community CLHEP is used internally by many programs and therefore, at the moment, interfacing to HepMC as an output format introduces no further dependences. However a change to MathCore inside HepMC would do this and we are strongly opposed to such a move.
- As HepMC is essentially stable and has been successful for a number of years we do not see any need to change it in this way.

⇒ Notice that many other users echo this last comment.

More comments from the Herwig++ and ThePEG authors

- The original proposal to have the LCG take responsibility for HepMC was intended to move away from the situation where there were two versions of the code with the one in CLHEP not being updated fast enough. It was also hoped that by decoupling the projects bug fixes etc could happen faster.
- This proposal does nothing to deal with this problem and in fact only exacerbates it. As it seems impossible to reach a consensus the default should be as near as possible to the status quo but solve the original problem, i.e. that HepMC moves to LCG and continues to use CLHEP. If this is not possible, then we think that HepMC should at least be a neutral, stand-alone package, and let the users choose between CLHEP or MathCore.
- For a change to MathCore there needs to be broad support from the theoretical, experimental and Geant4 communities which obviously does not exist.

MathCore

- Expect lots of discussion at this workshop
- Don’t forget backwards compatibility
  - Is that handled by using 1.27?
- Next few slides show various proposed solutions.

1. Simplistic Solution

- Modify code such that it can be used with either GenVector or Vector.
- Lots of #if statements in the code
- Have to build two separate libraries
- Not acceptable

2. Use Templates

- Powerful
- Default would be to use CLHEP Vector
  - No changes to existing code
- All differences handled by templatization
- User chooses vector class
- Many people have reservations
  - How do you know which vectors were used?
  - What do you load into CINT?
  - Will you end up with separate libraries anyway?

3. Eliminate dependency

- Include some vector package with HepMC
- Dangerous
  - Most likely end up effectively linking to two different versions of the same vector package
- Which vector package?
- Doesn’t solve the problem

4. Status Quo

- CLHEP HepMC
  - Hidden users have surfaced
- LCG HepMC
  - Does this use GenVector or Vector?
- This is exactly the problem that we are trying to avoid

5. Bite the Bullet

- Build HepMC 2.0 with GenVector
  - Not backwards compatible
  - Using vectors that will be supported
- CMS is aggressively moving to MathCore
- ATLAS and LHCb are considering
- Continued support for HepMC 1.2x
  - Several versions in the same place better than several versions in different places
  - Still support headache - but acceptable

Conclusion

- We’ve already had far too much confusion with HepMC
- We’ve made a start on resolving this
  - Issues with communications
- Expect interesting discussions this week
- Remember that ongoing discussion will be at LCG Generator monthly meetings
  - project-lcg-simu@cern.ch