MixMax integration in CLHEP

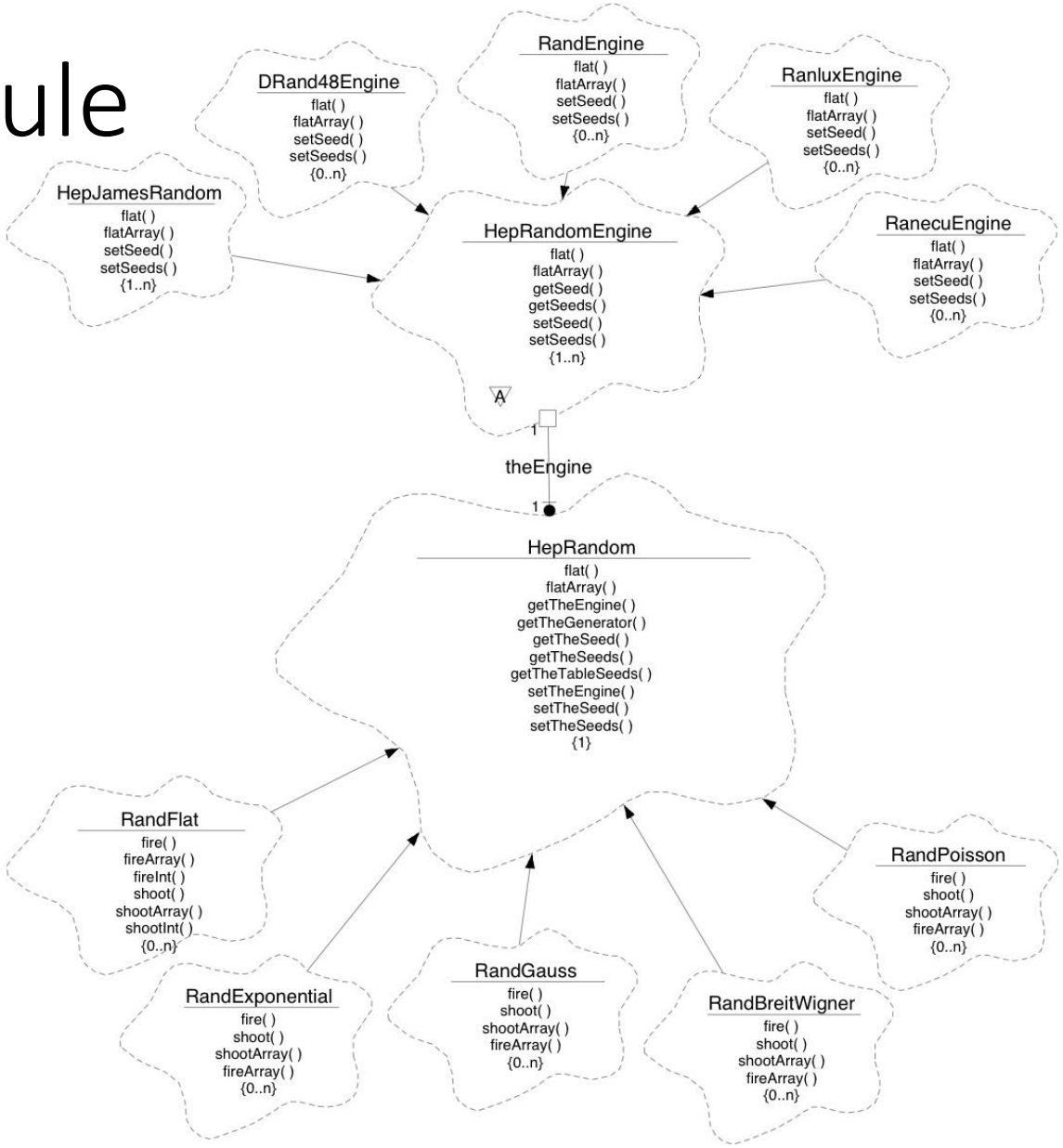
Gabriele Cosmo, CERN-EP/SFT

Contents

- The Random module in CLHEP
- Random engines available in CLHEP
- The testing suite
- MixMax in CLHEP, versions and current status

The CLHEP Random module

- Originally part of the Geant4 simulation toolkit
  - First implementation back to 1995
- Becomes part of CLHEP in 1997
  - http://cern.ch/clhep
- Set of C++ classes implementing
  - Random number generators (engines)
  - Random distribution classes
  - Common interfaces for seeding/shooting random values, saving/restoring engines' status, etc…



Random engines in CLHEP

- Groups 15 generator classes all written in C++
  - Some engines now rather obsolete, which could well be removed
  - Most engines working only with 32-bits precision
  - Several of them now also available through the C++11 Standard or the GNU Scientific library
- HepJamesRandom
  - Implementing the algorithm by Marsaglia-Zaman (RANMAR), described also in "F.James, Comp. Phys. Comm. 60 (1990) 329”
  - Historically chosen as the default random engine
- RanecuEngine
  - Multiplicative Congruential generator using formula constants of L'Ecuyer, derived from the F77 implementation of “ranecu” by F.James in CERNLIB
- RanluxEngine
  - Derived from the F77 implementation by F.James and described in “M. Lüscher, Comp. Phys. Comm. 79 (1994) 100”, original ralnlux-24
  - Exists also in double-precision version (Ranlux64Engine, using double-precision floating point arithmetic internally) as described by Lüscher in 1997 (2nd generation of Ranlux Lüscher algorithm, ranlux-48).

More random engines in CLHEP …

- MTwistEngine
  - Mersenne-Twister (MT19937) generator by M.Matsumoto and T.Nishimura, as described in “M.Matsumoto and T.Nishimura, *ACM Trans. on Mod. and Comp. Sim.*, Vol. 8, No. 1 (1998) 3–30”
  - Using original seeding procedure
- Hurd160Engine, Hurd288Engine
  - Engines based on a Linearly Interconnected Shift Registers algorithm, as described in “W.J.Hurd, IEEE Trans. on Comp., Vol. c-23, No. 2 (1974) 146-152
- RanshiEngine
  - Based on the algorithm described in “F.Gutbrod, Comp. Phys. Comm. 87 (1995) 291-306”

The CLHEP testing suite for Random

- Set of fine-grained tests for provided interfaces
  - Sequence generation reproducibility (streams, files, engine's copies,…)
  - Reproducibility of distributions
  - Generation of vectors of numbers
  - Thread safety
- No tests for randomness quality

MixMax in CLHEP

- MixMax included in CLHEP version 2.3.1.0 (November 2015)
  - Implementation of MixMaxRng engine wrapper in C++
  - Based on mixmax v1.1 (N=256, s=487013230256099064, m=1)
  - Part of Geant4 release 10.2 series (http://cern.ch/geant4)
  - Showing performance issues when setting seeds and shooting vectors of numbers (issues even more evident on multi-threaded applications)
- Corrections included in CLHEP version 2.3.3.0 (May 2016)
  - Based on mixmax v1.1 (N=17)
  - Corrected performance issues (thanks to smaller N and use of seed_spbox() for seeding MixMax with single seed); adopt simple loop for vector generation
  - Using wrong values for multiplier constants
- Corrected version included in current development version of Geant4
  - Based on mixmax v1.1 (N=17, s=0, m=2^36+1)
  - Ready to be included in next CLHEP release and Geant4 release

Testing MixMax from CLHEP

- ✓ Passing all tests in the CLHEP testing suite
- ✓ Validated in Geant4 simulations through Geant4 testing suite, including multi-threaded applications
- ✓ Smart seeding algorithm
- ✓ Excellent CPU performance
  - ❖ Shooting 700 millions numbers (sequential) grouped in vectors

    (MacOS, 3.06 GHz Intel Core 2 Duo)

    HepJamesRamdom - 9.5s
    Ranlux64Engine (luxury 2/3) - 76.81s
    RanecuEngine - 10.44s
    RanluxEngine (luxury 4/5) - 67.32s
    MTwistEngine - 8.32s
    RanshiEngine - 6.61s
    MixMaxRng - 6.51s

Next…

- Jump on testing new mixmax 2.0 !
- Aim to include C++ version in CLHEP/Geant4 if robust enough
- Should set MixMax as the default generator in CLHEP/Geant4