Status of Geometry and Transport

John Apostolakis, Gabriele Cosmo, Andrei Gheata, CERN EP-SFT
for the Geometry & Transport Working Group

Geant4 Collaboration meeting 2024 - Catania, 8 October 2024

Outline

- Planned & (not) achieved in 2024
- Developments & Fixes in Beta4.11.3-1
  - Solids
  - Refactoring & Parallelisation
  - Navigation
  - GDML
  - Magnetic field
- VecGeom developments & migration path to version 2

Planned 2024 : Geometry

- VecGeom
  - Complete surface modeler prototype (portable & efficient GPU geometry) ✔
    - missing solids, integration in AdePT, optimisation structure ✔
    - Extension of BVH navigator to surfaces handling ✔
    - See Severin's talk today
  - Code simplification, removal of CUDA specific code (✔ solid CUDA library not needed on GPU)
  - Improve portability of SIMD-aware solids
- Reduce geometry initialization time using multi-threading ✔
  - Reduce time spent for voxelisation of complex geometry setups by adopting multithreading/tasking technique ✔
- Investigate use of multi-threading to speed up overlap checking and volume calculation
  - Study possibility to speed-up generation of random points on surface using multi-threading
- Investigate alternative implementation of navigation history
  - Review implementation of navigation history for possible memory and speed optimisation

Planned 2024: Field propagation

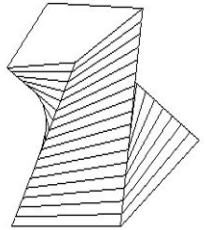
- Optimisation of QSS field driver (Quantized State Simulation) ✔
  - Review existing implementation of QSS for improving robustness and speed (QSS3)
  - See Rodrigo/Matias talk on Monday
- New UI commands for controlling field parameters ✔
  - Provide UI commands for simple applications (with one field manager / configuration only) to control parameters for propagation of charged particles in field and the accuracy of intersection of their curved trajectories with volumes
  - See Ivana's talk today
- Complete implementation of high-order "symplectic" integrator for accelerator applications
  - Review, complete and test existing Beta implementation of "symplectic" integrator
- Review accuracy of boundary crossing in field
  - ALICE and CMS requirement

Solids

- Improvements
  - 11.2.0 / 11.3-beta: Improvements in speed/stability for the calculation of cubic volume in Boolean solids
    - Refinement of Boolean mesh operations for calculating cubic volume
    - Implemented new methods *GetNumOfConstituents()* and *IsFaceted()*
    - More numerically-stable external processor for Boolean meshes
  - Completely revised and optimised implementation of *G4GenericTrap*.
    - Addressing problem report #2547
- Fixes:
  - 11.2.0: Fixed bounding box calculation in *G4VTwistedFaceted::BoundingLimits()*
  - 11.2.0: In *G4MultiUnion*, fixed problem in external Boolean processor, which resulted in nullptr
  - 11.2.1: Fixes in *G4UTrap* for a problem in CMSSW after migration to VecGeom 1.2.6
  - 11.3-beta: Removed internal cached state of *G4TwistedTubs* and *G4VTwistedFaceted*
    - Addressing problem report #2619
  - Fixed Coverity defects + clang-tidy (applies to all categories)

improvement

G4GenericTrap solid, motivation for revision



- *G4GenericTrap* will replace the ATLAS custom solid in the ATLAS Electromagnetic Endcap Calorimeter (EMEC) geometry description. The CPU performance is a critical matter for simulation in EMEC, currently it takes more than 30% of the total CPU time. Below is the benchmark result for the old and new versions (1 million calls, MacBook Pro M1)

G4GenericTrap	Inside	Surface Normal	Safety to In	Safety to Out	Distance to In	Distance to Out
Old	55 ms	155 ms	114 ms	116 ms	140 ms	114 ms
New	8 ms	23 ms	13 ms	13 ms	51 ms	72 ms

- The lateral faces of *G4GenericTrap* in general case are pieces of *parabolic hyperboloids*. Estimating the safety distance in this case is a non-trivial task and the result in the old version was often mistakenly overestimated (see issue #2547), which led to incorrect physics results.

Solids interface additions to speedup cubic volume estimation

- *GetNumOfConstituents()* returns the number of constituents used to construct the solid. This method was applied to optimise *volume* estimation in *G4UnionSolid* and *G4SubtractionSolid*.
  It can be also useful for statistical purposes, such as monitoring the complexity of Boolean solids.
- *IsFaceted()* returns *true* if the solid is bounded by planar faces, *false* otherwise. This method is currently not used, but in the future it could be used to detect polyhedra among Boolean solids. Volume of such solids can be calculated precisely and much faster using an analytical expression instead of the Monte-Carlo method.

Refactoring & Parallelisation

- **New feature**
  - 11.3-beta: Added new capability to run voxel optimisation in threads in MT mode.
    - Parallelises only over volumes. See next slide.
- **Refactoring/cleanup**
  - 11.2.0: Simplified inheritance for touchables: *G4VTouchable* is not a base class any more
  - 11.3-beta: Moved *G4NavigationHistory* and helpers from volume to the management category
  - 11.3-beta: Removed references to unused classes *G4GRSSolid* and *G4GRSVolume*
  - 11.3-beta: Access *G4GeometryManager* singleton through its GetInstance() in stores
- **Improvements**
  - 11.2.2: Using *std::map* to speedup search of skin surfaces in large tables

new feature

Parallelising the voxelisation

- Creating the optimisation ‘voxels’ is time consuming in complex geometries
  - In each volume with multiple daughters, six choices of axis are tried
  - Opportunity to parallelise over volumes in large geometries (with many logical volumes that contain daughter volumes.)
- New capability to do voxel optimisation using threads / tasks.
  - Parallelises the optimisation over (logical) volume
  - First version was included in 11.3-beta release (June 2024)
- User can call new method to turn it on/off
  - *G4GeometryManager::RequestParallelOptimisation(optimise, verbose)*
- Implementations / issues
  - Current it uses *G4WorkerRunManager* - not ideal, and glitched in first CMS tests
- Refinements
  - Identified needed improvement - move to *G4WorkerRunManagerKernel*
  - Topic of working ‘Breakout’ session (Tuesday 16:00 - Parallelisation of Initialisation)

Navigation developments

11.2.0
- Added new *G4VNavigation* common navigation interface class
  - 11.2.0 Added new *G4SafetyCalculator* class, auxiliary to *G4Navigator* to avoid saving/restoring state
  - Added *RelocateWithinVolume()* method to *G4VoxelNavigation* and to *G4ParameterisedNavigation*.
    - 11.3-beta Updated existing navigators to make use of the new common interface
- *G4TransportationManager* now notifies *G4FieldManager* about the global field
  - 11.3-beta Done to avoid a dependency on the navigation module
  - Needed to implement new capability to set the 'parameters' for field propagation (see Ivana's talk later in this session)
- Fixes:
  - 11.2.0 Fixed */geometry/test/check_parallel* UI command

GDML

- Added possibility to set alternative grammar for schema validation while reading a GDML file (11.2.0)
  - Validate based on local file w/o network
- Fix in schema module gdml_solids.xsd for tessellated solid semantics, to correctly reference facet types (11.2.2)
  - Schema updated to GDML-3.1.7
- Disabled default GDML schema validation as XercesC currently does support only 'http' (11.2.2)

Field propagation

- Added QSS integration method

11.2.0

  - Improvements on the QSS Stepper: the main QSS integration loop is interrupted when the number of substeps exceeds a predefined threshold (set to 1000 by default).
- *DormandPrince745* remains the default stepper.
- Fixed potential uninitialised data, detected in ATLAS LTO builds

11.2.1

- Added new classes for automatic field construction from parameters

11.3-beta

  - *G4FieldBuilder[Messenger], G4FieldParameters[Messenger]* and *G4FieldSetup*.
  - Ivana's talk will give the use case and explain

VecGeom developments

- Main development is focused on the new surface modeler targeting GPUs
  - It is being validated and optimized for several complex setups
  - ***surface_model*** branch is the development branch for the new surface modeler
    - Run-time dependencies on the CUDA solid model recently removed
- Small changes in the master branch
  - CI/CMake and logging improvements
  - bug fix polyhedron safety
- Old interface is now in ‘v1’ branch
  - In the v1-patches branch we ported only important fixes from master

VecGeom version 2: portable GPU geometry support

- Merging the surface model in the master once fully validated
  - Creating a new patches branch where both surface + solid will be allowed on GPU, for comparisons
  - Only surface model changes and solid model fixes pushed in the patches
- Implementation of a portability layer on top of the surface model for GPU support
- Discontinuing solid model support for GPU (prevents a portable v2)
    - The solid model will continue being available on CPU
- Further interfaces cleanup on the solids part, before tagging v2
  - Cleaning-up the *cuda* namespace and specific methods

Thank you!