Geant 4

GEOMETRY

Software Requirements Document

Status: Released

Version: 1.1
Project: Geant4-Geometry
Created: 20 February 2000
Last modified: 10 February 2004
Prepared by: Gabriele Cosmo (CERN)

1. Components

1. The geometry modeler must be organized in components:
   - *Navigation system, geometrical entities*
   - *Positioning of volumes, volume attributes, geometrical parameterizations*
2. Abstract interfaces should be provided at components level to facilitate future extensions to the system and provide for uniform treatment of functionalities at user level. Specific implementations must obey to the same abstract interface.
3. Interface definitions must be as much as possible decoupled from their specific implementations, which should be possibly deferred to subclasses.
   - *Example: generic definition of solid and its functionalities vs. specific box and algorithms*
4. Additional geometrical entities for specific use-cases can be added to the system provided that all expected functionalities are implemented according to their specifications.
5. Graphical representations should be decoupled from the geometrical entities to allow for reuse in software components not directly related to the geometry modeling.
6. Components in the geometry module should be independent from tracking and physics. The only external dependencies foreseen should be related to logical attributes associated to the geometrical entities (e.g. materials, graphical attributes) or to specific user-interface paradigms (e.g. messengers for UI commands).
7. Persistency of the geometrical objects should not be hard-coded in the modeler but defined through a data-model implemented in an external dedicated module.
8. Interface to CAD systems should be implemented by either following the STEP-EXPRESS standard or by any other data exchange format implementing a standard binding to such systems, either through file converters or dedicated parsers implemented externally to the geometry modeler.
   - *Eventually providing persistency capabilities for importing and exporting geometrical models.*

2. Geometry modeling

1. Modeling of a geometry should happen in stages, in the following order:

    i. *Definition of all necessary attributes relevant for the association to the geometrical entities (materials)*

    ii. *Definition of the geometrical entities (solids or shapes)*

    iii. *Definition of the logical entities (logical volumes) associating geometrical entities and relevant attributes*

    iv. *Definition of the real physical geometry model by positioning the logical entities in the space (physical volumes)*

    v. *Definition and association of additional attributes (geometrical regions, detector sensitivity, field managers, graphical attributes, etc.)*

2. It must be possible to achieve all above steps of the modeling of geometry by means of the defined abstract interfaces.

3. It must be possible to define multiple geometry models in the same setup and provide the ability to easily exchange them if necessary.

4. The system must take care of the management of the concrete objects (solids, volumes) allocated in memory, and eventually provide facilities for manually configuring their registration.

3. Optimization and performance

1. Navigation and relocation of points in a generic geometry model must be well performing at run-time. An optimization strategy must be implemented in order to reduce as much as possible unnecessary computation of intersections with volumes.

2. The optimization algorithms must be efficient and require little resources in terms of time of execution and memory consumption.

3. The optimization algorithms must apply independently from the geometry setup (geometrical shapes, kind of placements or hierarchical/flat structure).

4. It must be possible to selectively choose areas in the geometry model where to apply or not apply the optimization.

5. Navigation in optimized geometry setups must happen following the same interfaces defined for non-optimized geometry setups and must be transparent to the user.

6. Floating-point precision must be adopted in the implemented algorithms for the navigation, optimization and solids response.

- *Ad-hoc tolerances may be defined for the treatment of points at boundaries*

4. Testing

1. Adequate testing tools should be available for the intensive testing of geometrical entities response.

   - *Specific setups must be considered, verifying the response of the solids against known quantities*
   - *Known critical test conditions should be implemented and regularly verified*

2. Adequate verbosity levels should be implemented. It must be possible to easily activate them at run-time (or through pre-compile flags in the cases where performance is critical).

3. A comprehensive test suite based on a "real-case" application should be adopted in the evaluation and monitoring of run-time performance, under different conditions (i.e. transportation with and without magnetic field presence)

5. User interface

1. The public interface exposed to users must be as simple and intuitive as possible, with no ambiguities.

2. Interactive commands for the main functionalities provided by the modeler management may be provided through dedicated messengers.

6. Miscellaneous requirements

1. Coherent and uniforms coding style and editing conventions should be applied.

   - *Coherently indent the code (2-3 space characters per level)*
   - *Avoid usage of <TAB> keywords in the source code*
   - *Coherent usage of '{}' brackets in methods and classes*
   - *Coherent adoption of comments in the implementation*

2. The software should be compatible with Geant4 design and coding guidelines.

3. Usage of STL containers should be considered.

   - *Usage of C-style arrays should be avoided*
   - *Effective use of STL iterators and algorithms should be exploited*

4. Performance optimization should be based on quantitative measurements (profiling).

5. Specific unit tests should be provided for relevant new functionalities implemented. The tests should be designed in function of the different use-cases associated to such functionalities.