Update Geometry & Transport

Developments in 11.1, 11.2.beta; fixes; ongoing

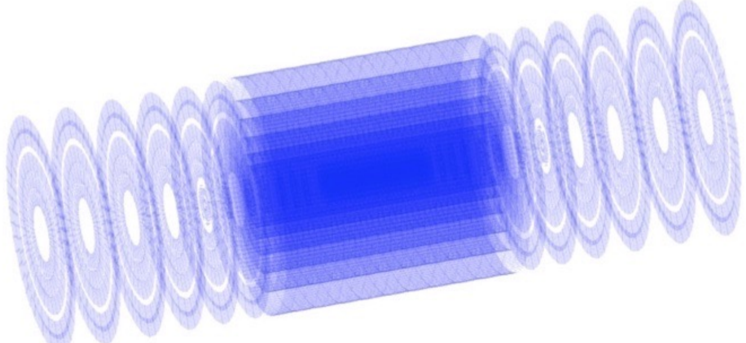
John Apostolakis & Gabriele Cosmo (CERN EP-SFT)

for the Geometry & Transport WG

Outline

- Features introduced in releases 11.1
  - VecGeom updates; Symplectic integrator; Coupled Transportation redesign
- VecGeom evolution (to version 2.0)
  - Simplification
- VecGeom: Development of surface-based modeler
- Integrated Quantum State Simulation (QSS2/3) – 11.2 beta

VecGeom – updates in v1.2.1



- Improvements and optimisations to BVH acceleration (for CPU and GPU)
  - Added *surface area* heuristic for BVH construction
  - Added implementation of *marching cubes* algorithm
- Extended *GDML reader* to support all existing shapes
- Improved ‘infrastructure’
  - Selection for enabling use made at configuration
    https://gitlab.cern.ch/VecGeom/VecGeom/tree/v01.02.01
  - Modernised Cmake usage and settings; switched to C++17 by default
  - Improved CUDA support in configuration and memory allocation
  - Bug fixes

11.1 developments: Navigation, Volumes, Transport

- Revised implementation of *G4CoupledTransportation* (Jonas Hahnfeld)
  - Inherits from G4Transportation; consolidated common variables and methods
  - Allowed G4Transportation to be base class for *G4TransportationWithMsc*, which combines transport with multiple scattering
  - Simplifies future maintenance challenge; it was already a concern.
- New class *G4TransportationParameters* for fine grain control of parameters for killing charged particles **looping** in a field
  - Optional, but it applies to all stable charged particles if created.
- New option to check for **overlaps** in parallel geometries
  - Through `/geometry/run/test` UI command
- **Improved** computation of surface **area** and cubic **volume** in specific solids

11.1 developments: Field

- Revised *G4FieldManager* to ensure robust behaviour of the integration
  - Keep **`epsilon_min/_max`** parameters for relative step accuracy between 'minimum and maximum accepted' values
  - Motive: poor accuracy of G4DormandPrince for epsilon > 0.001 (diverged by 10x)
- New 2nd order symplectic integration method *G4BorisDriver* *
  - Symplectic methods aim to conserve energy & phase space volume
  - This is first method in Geant4, and delivers low-order 'conservation' – deviations are proportional to (d/R)^3 , so step size must be kept low for accuracy
  - Note: Further work is required to finish development of the higher order (4th) methods needed by muon (g-2) and other accelerator-based use cases.

* GSoC 2022 project by Divyansh Tiwari

2023 Planned Developments
Geometry and Field

✔ In progress…
✔Achieved already in development releases

Propagation in Magnetic Field – 11.2 beta

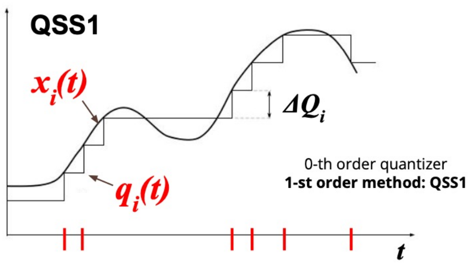
- Quantum State Simulation (QSS) method introduced
  - Alternative integration method, currently only for pure magnetic field
  - Second (QSS2) and third (QSS3) order methods
  - Provides Interpolation capability
- Refined control of very-long steps (typically in vacuum)
  - Prompted by challenge for drivers with interpolation

QSS

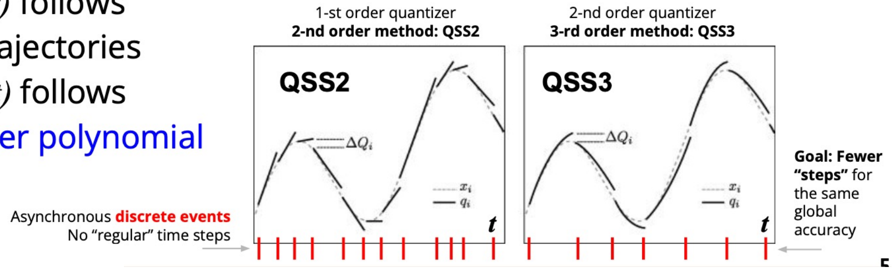
- Integrated into 11.2-beta
  - Second level text
  - Details

Higher order QSS

$$q_i(t) = \begin{cases} x_i(t) & \text{if } |q_i(t^-) - x_i(t)| \geq \Delta Q_i \\ q_i(t^-) & \text{otherwise} \end{cases}$$



- ΔQ_i is the **quantum**
  - **Maximum deviation allowed** between x_i and q_i (error control)
  - Derived from the **accuracy** demanded by the user
- **Higher order QSS methods** (**QSS*n***) follow a similar principle
  - In a **QSS1** method, $q(t)$ follows piecewise constant trajectories
  - In a **QSS*n*** method, $q(t)$ follows piecewise (*n*-1)-*th* order polynomial trajectories



Summary of results: QSS vs. DOPRI

Example	Method	QSS accuracy parameters: dQrel	dQmin	% of Intersections per G4 Step	QSS Substeps per G4 Step	User Time (seg)	System Time (seg)	Real Time (seg)	Average Time per G4 Step (seg)	Speedup (QSS vs. DOPRI) Real Time
B2a	DOPRI	N/A	N/A	3.79%	N/A	2.052	0.175	2.614	1.3E-04	N/A
B2a	QSS	1.0E-02	1.0E-03	3.75%	10.191	2.067	0.176	2.654	1.3E-04	-1.53%
B2b	DOPRI	N/A	N/A	3.73%	N/A	2.081	0.178	2.651	1.3E-04	N/A
B2b	QSS	1.0E-02	1.0E-03	3.77%	10.209	2.107	0.178	2.680	1.3E-04	-1.09%
B4c	DOPRI	N/A	N/A	4.31%	N/A	1.623	0.180	2.202	1.1E-03	N/A
B4c	QSS	1.0E-02	1.0E-03	4.02%	2.517	1.603	0.182	2.170	2.1E-03	1.43%
B4d	DOPRI	N/A	N/A	4.31%	N/A	1.637	0.183	2.217	1.1E-03	N/A
B4d	QSS	1.0E-03	1.0E-04	4.19%	5.026	1.605	0.178	2.164	1.1E-03	2.39%
B5 SingleBeam	DOPRI	N/A	N/A	2.78%	N/A	3.442	0.257	4.004	1.1E-01	N/A
B5 SingleBeam	QSS	1.0E-03	1.0E-04	2.78%	1,494.940	3.259	0.245	3.841	1.1E-01	4.06%
Extended Field 01	DOPRI	N/A	N/A	6.51%	N/A	1.020	0.096	1.347	7.4E-04	N/A
Extended Field 01	QSS	1.0E-02	1.0E-03	5.99%	37.787	1.014	0.096	1.333	6.7E-04	1.03%

(*)

- Performance
  - Tuning accuracy parameters
  - Compared with Dormand-Prince

b.

Improved control of long steps - issue

- Integrators with interpolation need to keep the full state for all intermediate substeps
  - G4InterpolationDriver<> creates & keeps state of 61 interpolation segments
  - QSS currently manages segments dynamically – currently without a maximum number (to fix)
- These integrators provide ’dense’ output used to intersect boundaries
  - No extra ’derivative’ (field) evaluations needed – just interpolation of existing values
- Field integration must treat steps with very large distance to next physics interaction
  - E.g. in vacuum more than 10^4 meters in a HEP collider experiment with larger field – O(tesla)

Improved control of long steps - changes

- G4PropagatorInField: turned hard coded values into parameters to control the longer steps
  - MaxStepSizeMultiplier is a multiplier for the ‘diameter’ of the current volume
  - MinBigDistance – a minimum ‘additional’ distance
- Chosen small(er) default values:
  - MaxStepSizeMultiplier = 0.1 (originally 100)
  - LargestAcceptableStep = 100 * meter (originally was 1000.0 m)

2023 plans: Geometry & Navigation

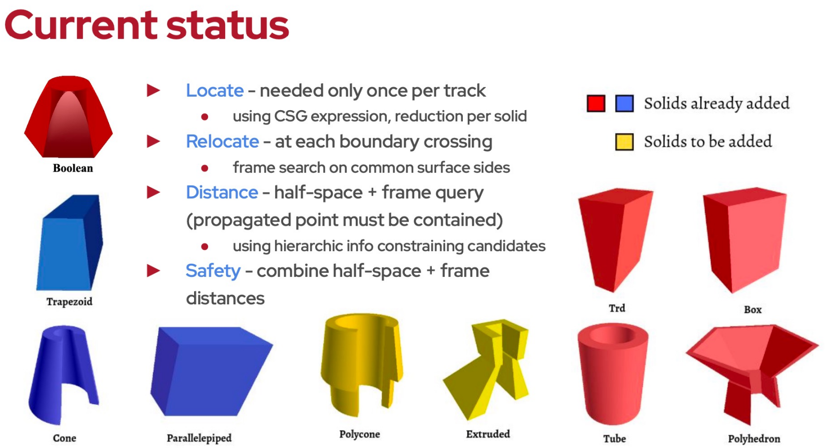
- Separate safety computation and its state from navigator ✔
  - Loose coupling of navigator in the computation of the safety distances from geometrical boundaries
  - Prototype is under testing (very small differences observed in full setups)
- Investigate simplification of touchables classes ✔
  - Code optimisation: removed unused specialisations (of G4VTouchable) and inheritance
  - Now *G4VTouchable* is a typedef to *G4TouchableHistory*

VecGeom – 2023 developments overview

- Improve portability of SIMD-aware solids
- Simplified VecGeom – eliminating unused, vector elements
  - Code simplification, removal of unused API/backends/specialisations
  - Mini-workshop/sprint at CERN in March to refine plans, deliver a first version
  - Refined and it is now the master branch of VecGeom: (no GPU support)
- Created a branch for 1.x patches (with old capabilities)
  - For use with existing Geant4 versions, e.g. in the next months, (and other use cases).
- Current master branch (2.0.0-rc1)
  - Removed vector APIs
  - Simplified Implementation Helpers.
  - Removed transformation specialization
  - Will make release (2.0) once GPU surface modeller is ready

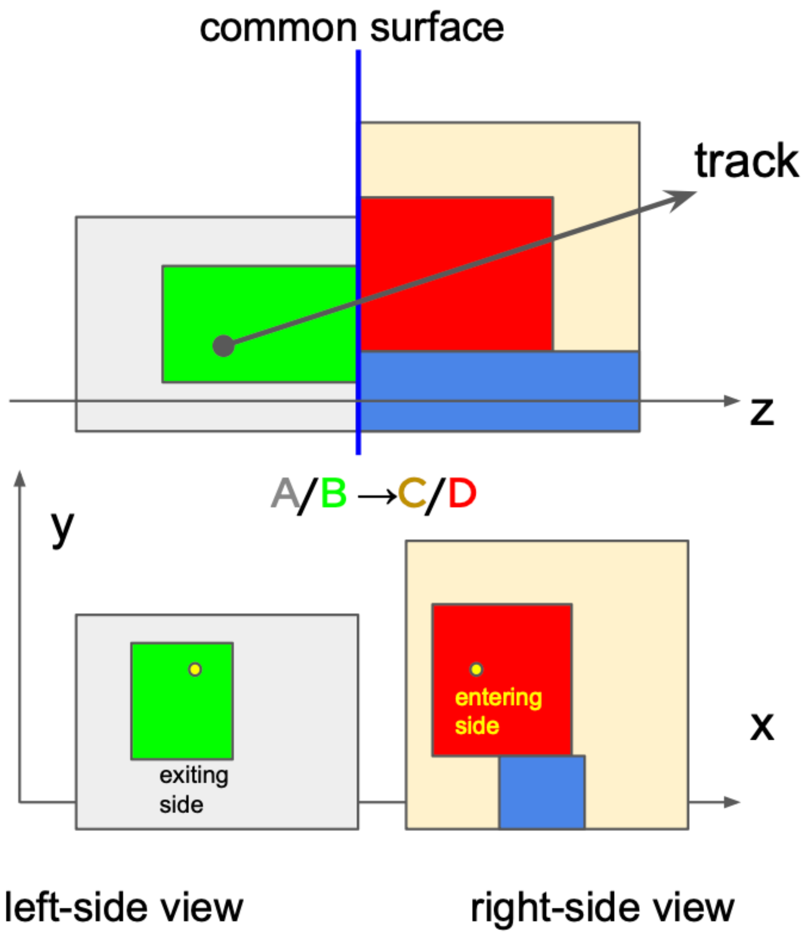
VecGeom: new surface-based modeller

- Development of VecGeom surface-based navigation
- Motive: avoid thread divergence on GPU
  - Reduce the large disparity between time to intersect simple and complex solids, which causes divergence and suppresses GPU performance
- Approach is to 'decompose' each solid into bounded-surfaces
- Each bounded surface has an infinite surface and an 'outline'/imprint of the solid on it
  - A box becomes 6 surfaces, so more data but simpler intersections
  - A Tube becomes 3 surfaces, a Tube-section can be 4 or 5
  - A polygon will have $N_{polygon} * M_{sections} + 2$ => can be large
- Many solids now converted, some remain
  - Done: Boxes, Trd, Tubs, Cones, Boolean, polyhedral
  - ToDo: polycone, extruded



VecGeom: new navigation

- Relocation at surface uses pre-computed information
  - Deposited ‘imprint’ of every solid that is on the common surface
- Algorithm to disentangle Boolean expression
  - Non-recursive method developed
  - Promising first results on GPU: 2x faster for many components, though it is 2x slower for few pieces (looping over surfaces.)
- Preliminary performance (looping over volumes)
  - Safety computation: ~2x slower on CPU, ~2x faster on GPU
  - Propagation + relocation: ~2x faster on CPU, ~6x faster on GPU
  - Memory: ~1 kByte per “touchable” volume
- Optimisations of memory and pruning candidate surfaces
  - Using ‘levels’ of geometry – full flattening => 3+ levels
  - First version of BVH optimisation
- Target is to run cms_2018 geometry working on GPU by end 2023
- Details in talk of Parallel Session 2B – Andrei Gheata (earlier Tuesday)



Field Propagation – remaining goals

- Review accuracy of boundary crossing in field
  - ➢ ALICE and CMS requirement

Bug Fixes

Patches in 11.1p01- Geometry

11.1.p01

- Solids/Boolean:
  - Fixed hang in G4MultiUnion, caused by oveflow of 'size-1' when 'size' value is zero
- Solids/Specific:
  - G4QuadrangularFacet: fixed references to triangles in the warning message issued when checking for collinear vertices
- Management:
  - G4LogicalVolume: use std::shared_ptr for handling visualization attributes. Ignore calls to SetVisAttributes() from worker threads
- Magnetic field:
  - Reduced printout for valid settings of epsilon_min/_max in G4FieldManager