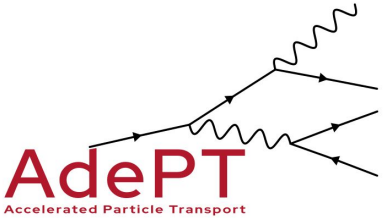







# VecGeom GPU plans

VecGeom Developers

May 4, 2022

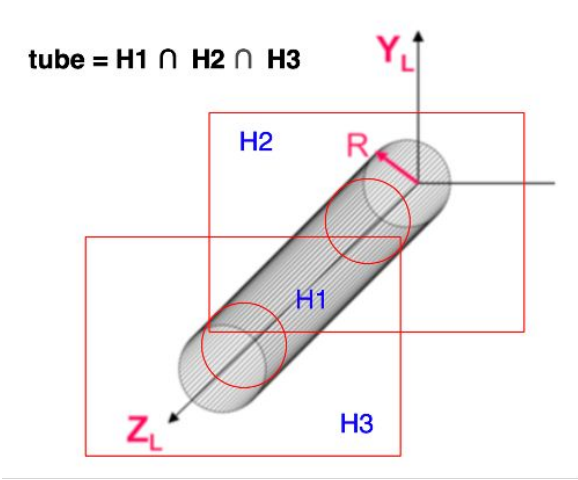

Detector Simulation on GPU Community Meeting

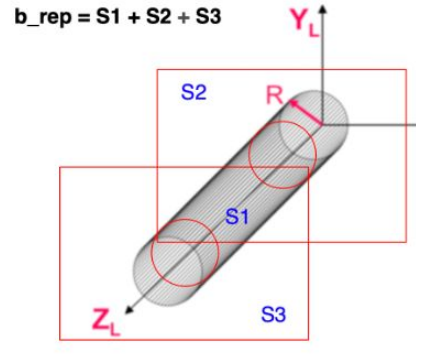
# VecGeom GPU support & evolution

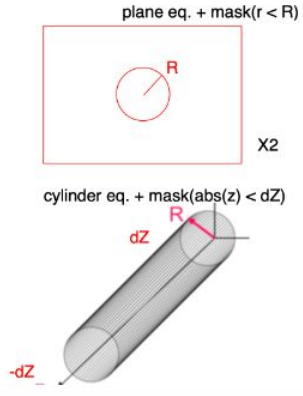
- A translation of the CPU C++ object model to GPU
  - Same code compiled in a *cuda* namespace in a different library
  - Numerically compatible code with CPU but:
    - not portable, polymorphic, with deep code stacks
    - highly branching, register hungry and divergent on GPU
- Main bottleneck in many AdePT workflows
  - Calls for a different approach for GPU to use less registers and be less divergent
    - simplifying the code and having more uniform algorithm complexity
    - the GPU model can be fully decoupled and constructed from the CPU model
  - Constraint: provide numerically compatible results with the current code
  - GPU requirement: allow for complexity reduction of current primitives code
    - provide the most uniform rather than the shortest code path
- A dedicated workshop earlier this year
  - Discussed candidates for a surface-based solid modelling approach

# Considered surface models

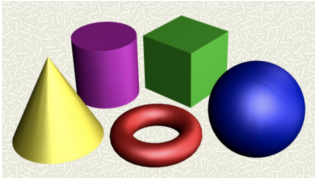
- Unbounded CSG models
  - Volume = intersection of half-spaces
  - Supporting Boolean operations
  - Example: *Orange* model (see session this afternoon)
- Bounded accurate surface models
  - Similar to tessellations but supporting also more complex surface types (mainly second order)
  - Surface + outline limit (mask)
    - E.g. (point - mask_origin).Mag2() < R * R
  - Example: *detray* model developed by ACTS
- Surfaces need binding to the touchable state



# Model types pros & cons



- Unbounded models:
  - Simplest surface representation (half-spaces)
  - Numerically robust for solid modelling
  - Boolean solids described naturally
  - A surface “hit” may be virtual, so a Boolean expression reduction is needed per volume
  - Specific problems for computing isotropic safety (larger overestimation)
- Bounded models
  - Surfaces must carry a “mask” condition -> larger (low-cost) combinatorics
  - The model must handle numerical “cracks” for joint edges
  - No need for volume-level reductions, surfaces are independent (think meshes)
    - Any ray-surface “hit” is a real solution
    - No need to track inside/outside a volume, the state can be inferred
      - So we can mix queries to surfaces coming from different volumes!
  - Isotropic safety can be computed in 2 steps: against the surface and against the outline

# Strategy

- Ideally take the best of the two worlds
  - Independent surfaces w/ outlines, but using components from Orange
    - Surface queries, Booleans implementation, relocation after crossing
    - Common algorithms for surface de-duplication (compacting common surfaces)
  - Discussing commonalities with the Orange team
- The estimated effort for a first implementation is ~1 year
  - A preliminary data model being investigated
  - A basic prototype assembling the model, as starting point for understanding better requirements and commonalities/differences with Orange
- Make a portable implementation by construction
  - Header-based approach with no run-time polymorphism, fully reusable as device code
- A development plan for a GPU prototype exists

# Development plan

- Stage 1: Project bootstrapping (now)
  - Data model, decide on common functionality with Orange
- Stage 2: Generate the surface description
  - A subset of VecGeom primitive shapes + example demonstrating it
  - Summer student project
- Stage 3: Model aggregation and base (looper) navigation
  - Data structure aggregation on host and copy to device
  - Simple demonstrator on host (device)
- Stage 4: Navigation and preliminary performance
  - Adapt BVH optimiser to work with surfaces
  - Performance analysis conditioning pursuing the project
- Stage 5: Full functionality and performance optimization
  - Missing solids and navigation functionality + optimizations

# Model concepts (details in the backup)

- Unplaced surfaces: surface “local” equation
  - Transform points & directions to the local frame, then use the simplest possible equation
- Outlines: providing the condition for hitting a surface
  - Can be also a set of triangles in case of complex planar surfaces
  - Usage of outlines require conversion to the local surface frame
- Placed surfaces: unplaced surface + outline + transformation on a scene
  - Store also a navigation state identifying the *VecGeom* touchable on each side
- Common surfaces: de-duplicated placed surfaces (more states per side)
  - Allow transitioning from one navigation state (touchable) to another
- Scenes: define a global reference frame for surfaces
  - The touchable global matrix is computed for the scene frame
  - Multiple levels of scenes possible if needed
  - Navigation optimizations (BVH) applied per scene

# Current status

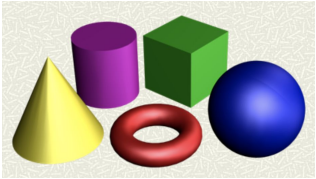
- Working on converting simple geometries, to understand better the needs and practical details
  - Started with *TestEm3* geometry of a sampling calorimeter
  - Planning to inspire from Orange for parts of the implementation
- We should try to figure out the commonalities and sharing the development
  - Could be a process converging from two sides to the same product, or two products sharing common code
- Started already discussing with the Celeritas team
  - Looking forward to closely collaborating on the new R&D

# Backup

# Solid modelling - relevant models

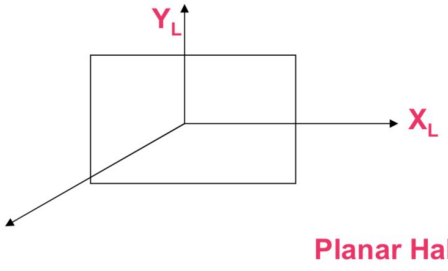
- Subclass of geometry modelling targeting **accuracy**
  - Mathematical description of equations defining the solid
- Primitive-based modelling
  - Bounded - Set of 3D solid classes (primitives) taking parameters to represent families of solids
  - Unbounded - Composition of half-spaces $\psi=\{(x,y,z): g(x,y,z) \geq 0\}$ *where:* $g: R^3 \rightarrow R$
- Constructive Solid Geometry (CSG)
  - Boolean combinations of primitives (binary Boolean tree)
  - Geant modelling uses a CSG with hierarchical containment constraints
- Surface modelling
  - Solid description making closed solids based on surface composition
  - Unbounded surfaces delimiting half-spaces
  - Bounded surface models (B-Reps) bodies limited by surfaces, **joint by edges** defined by vertices (**or mask operations**)
    - Similar to mesh representations, but the elementary surfaces accurate, not just planar faces.

# UnplacedSurface – half-spaces



- Realized we need to be able to transform surfaces between reference frames anyway
  - Why not using the simplest common surface representation in terms of half-spaces
  - Surfaces become “unplaced” and “placed”
  - Transform points & directions to the local frame, then use the simplest possible equation
- Plane half-space: XY plane with normal along Z; Tube/cone = Z axis aligned
  - *UnplaceSurface* has type and id in a data store
  - Provides an **Intersect** function with a ray

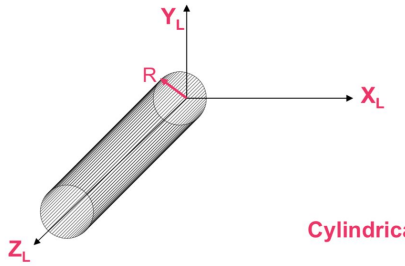
$$H = \{(x, y, z) : z < 0\}$$




Planar Half Space

A cylindrical Half Space is a given by:

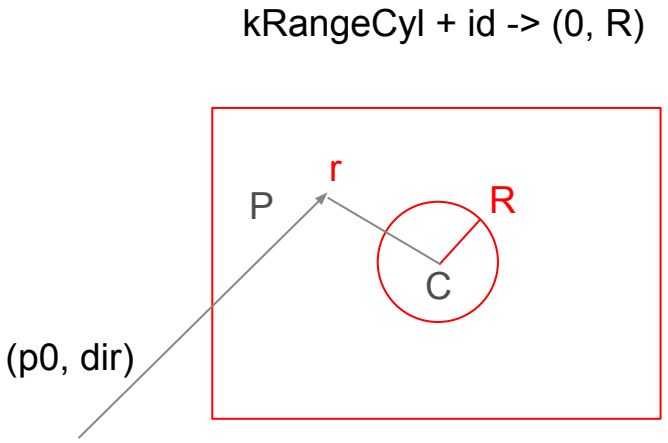
$$H = \{(x, y, z) : x^2 + y^2 < R^2\}$$




Cylindrical Half Space

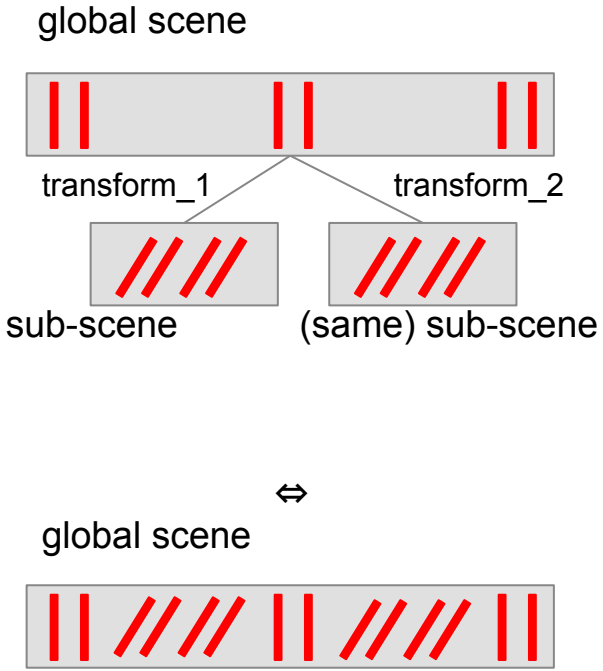
# Outline

- Defines surface boundaries on the UnplacedSurface support, in the **local** surface frame
- Provides Contains(localpoint) functionality
  - E.g. A box uses 6 ***kWindow*** masks, a window is always a range in X,Y, centered in the origin
  - Outline::Contains is typically a very simple function
  - For Booleans, rather use individual component outlines to solve a combined query
- Outlines can be at the limit triangles, for the more complex cases
- Outlines have just type and id in a data store



# PlacedSurface

- An unplaced surface + outline become “real” when placed in a common scene
  - A scene can be top-level (global), but also at lower (volume) levels
- It has an UnplacedSurface + outline, but also transformation with respect to a scene, and navigation state associated to the corresponding touchable
  - For single global scenes, the transformation is the multiplication of all local volume transformations, the state matches the current navigation state
  - The implicit state on the back-normal side is the one corresponding to the mother volume

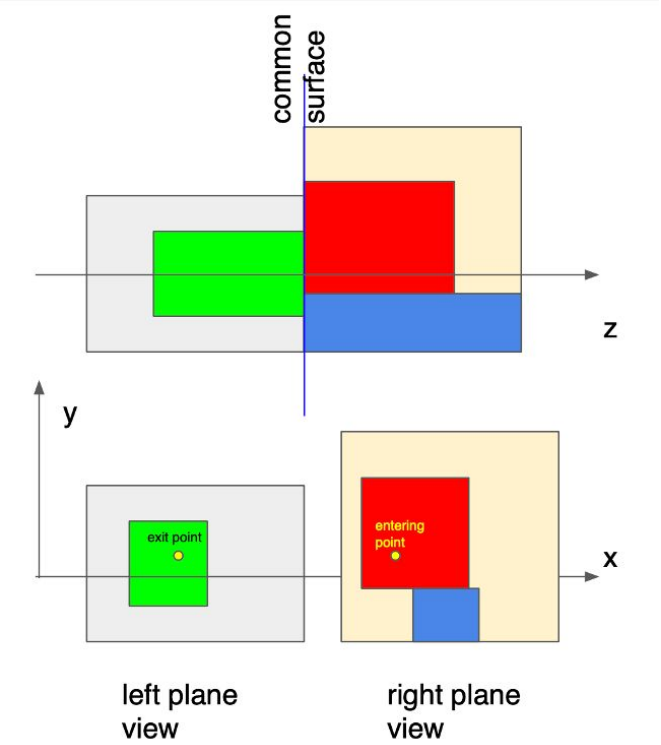


# PlacedSurface functionality

- Transform - transform global to local point
- InsideOutline - calls Inside with local coordinates for the outline
- Clone - clones the surface as defined by a solid to a PlacedSurface on a scene
  - A logical box volume will create 6 PlaceSurfaces with transformations in the local frame;
  - Placing the volume (or its ancestors) several times will clone all these surfaces as many times as placements, with the correct **composed** transformations in the volume hierarchy
- Once an individual placed surface is crossed, the state can be automatically inferred

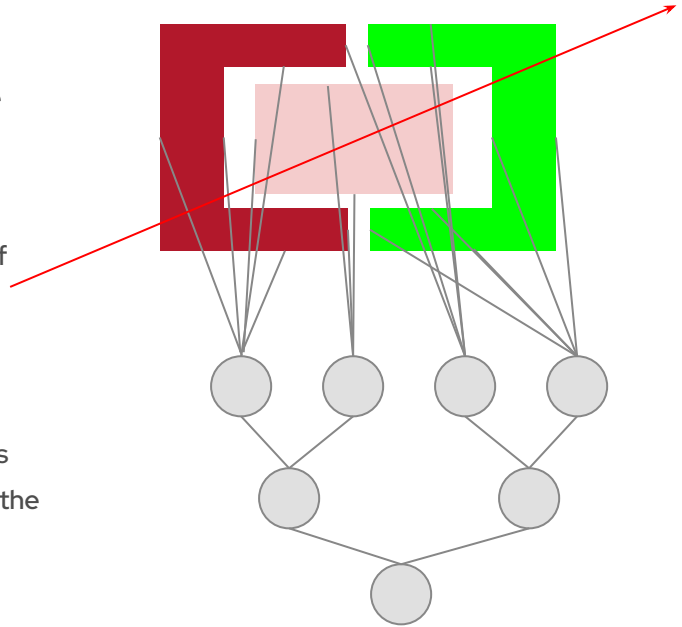
# Relocation with common surfaces

- Ancestor-daughter, neighbour volumes can have common surfaces
- Allow faster relocation after crossing boundaries
  - An exit point on one **side** of the common surface becomes an entry point on the other **side**
  - Need to find surface outline with highest depth containing the entry point
  - Still a point of divergence (variable loop over competing outlines)



# Scenes

- Representing all surfaces for a geometry sub-tree, flattened at a given logical volume level
  - Surfaces from a different volume tree can be placed in the scene
- There can be a hierarchy of scenes, but for practical purposes we can live with 2 levels
  - The idea is to use less memory, but also to balance the number of surfaces per scene
- PlacedSurface objects on a scene use a local scene navigation state
  - The global navigation index for a 2-level world is a tuple of indices
  - When converting global points to a sub-scene, one needs to use the scene transformation
- We can use one navigation optimizer per scene



BVH

# BrepHelper and SurfData

- The data model works with id's making composition easy, and size of objects static
- To access the actual data pointed by id's, a common storage is currently used, providing interfaces to access typed data (surfaces, masks, transformations) by id
  - Simple struct (SurfData), usable as is on both host/device
    - Templated by a Real_t type, to support easier single-precision in future
  - The SurfData pointer is percolated in the model interfaces, making the model fully backend independent
    - SurfData can be "cooked" on host by BrepHelper, then just copied and used on device.
- BrepHelper, provides assistance for converting the volume hierarchy model to the surface model
  - Generation of PlaceSurface per solid can be eventually done in the UnplacedVolume code