FERMILAB-SLIDES-23-302-CSAID

Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

# In-memory Geometry Converter

**Guilherme Lima**

Geant4 Collaboration Meeting
September 25-29, 2023

# In-memory Geant4 geometry converter

- What is it?
- How does it work?
- Validation details
- Status and plans

# Geometry converter: what is it?

- Converter developed within the context of Celeritas, which goal is to allow a Geant4 job to offload some of the tracking to a GPU device
  - see Seth Johnson's talk for more details on the Celeritas project

- VecGeom (VECtor GEOMetry) was developed to promote SIMD-vectorized algorithms. Since its algorithms could also run well on GPGPUs, it became a natural choice for HEPsim-on-GPU prototypes like Celeritas and AdePT.
  - Celeritas uses VecGeom v1.x for now, at least while surface-based systems (VecGeom 2.x and ORANGE) are being developed

- In order to offload some tracking to the GPU, the Geant4 geometry needs to be made available in VecGeom.

# Converting geometry from Geant4 to VecGeom

- Temporary shortcut: Geant4 geometry → GDML file → VGDML parsing → VecGeom geometry
  - Not ideal: limited precision (ASCII representation of floats in the GDML file), extra configuration steps and human error modifying GDML files.

- **Ideal: in-memory Geant4-to-VecGeom geometry converter**
  - Started from a preliminary converter, developed by S.Wenzel, J.Apostolakis *et.al.* as part of an effort to integrate VecGeom's SIMD-accelerated navigation into Geant4 (module G4VecGeomNav).
  - We have adapted this (CPU-only) converter to the Celeritas (GPU) environment
    - Implemented an overall scaling factor, to be multiplied by all length dimensions, in analogy to VGDML parsing scale:
      Default units: Geant4 (mm) → Celeritas+VecGeom (cm)
    - Added some (dummy) GPU interface required by GPU-enabled VecGeom classes.

# Geometry converter: how does it work?

- Use the geometry hierarchy:
  - Start from world volume → logical volume which holds placed daughters (G4PhysVol = G4Solid + transf.matrix), each daughter → G4LogicalVolumes which may hold more daughters, and so on…
    Note: construction usually starts from the leaves (daughters) placed inside mothers, which are placed on their mothers, and all the way up to the topmost world volume
  - Same hierarchy exists in VecGeom, with G4VSolid → UnplacedVolume, G4LogicalVolume → LogicalVolume and G4PhysicalVolume → PlacedVolume
  - Geometry object conversion is a recursive process:
    it starts from the world volume, loops over its daughters, converting each physVol→placedVol (e.g. converting its solid, and its transf.matrix) and its logical volume to VecGeom → converting each placed daughter recursively… until the whole geometry is fully converted.

# Validating the converted geometry

- Compare capacity calculations (from the upstream converter)

  G4Solid::CubicVolume() vs. UnplacedVolume::Capacity()
  → fixes to default units (scaling) and parameter interpretation

- Compare printouts for VGDML vs. converted geometries
  - Good for large-scale comparisons (e.g. CMS or Atlas-like detectors), dump all VecGeom geometry details, including volume dimensions and placement matrices

    ```
    world_PV->PrintContent();
    ```

  - Proof of concept: use a convenient GDML file with one of each shape available
    → significant cleanup for homogenized dimensions

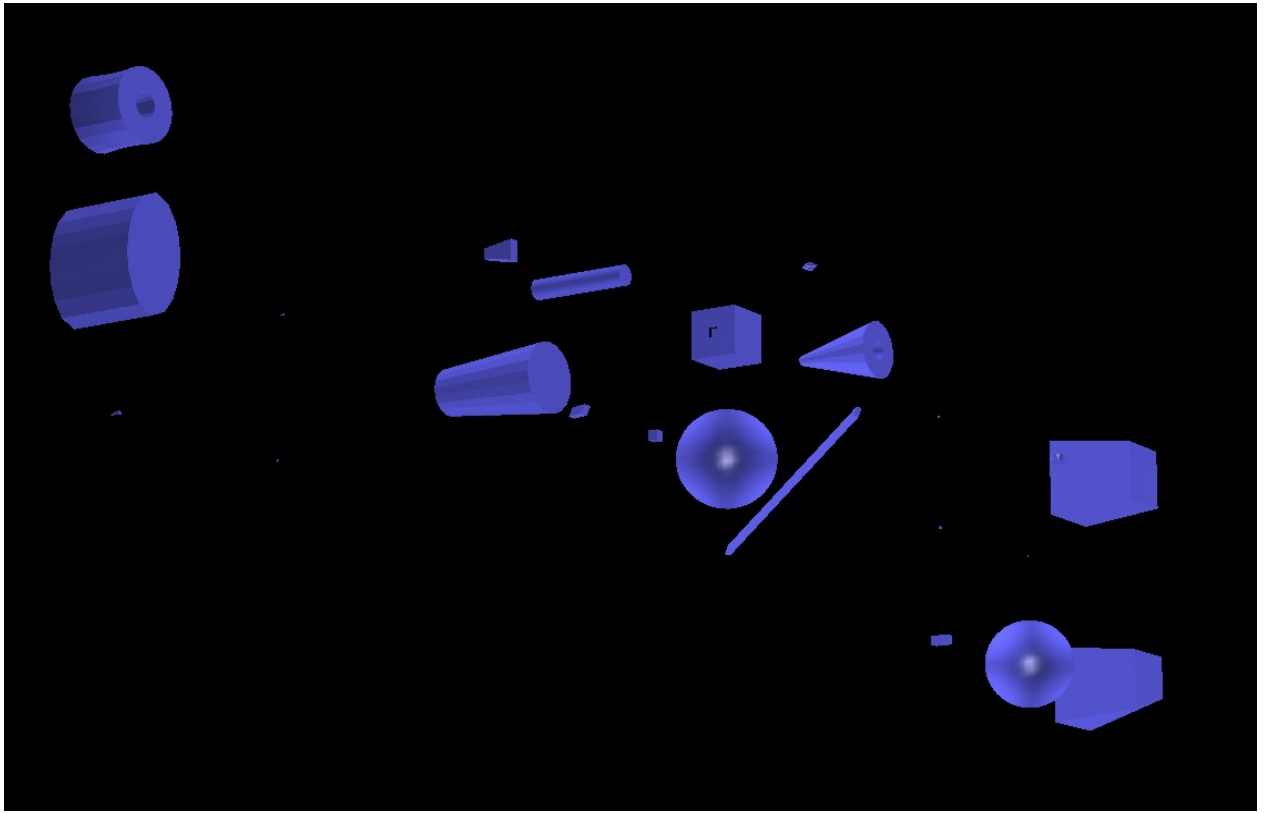
    → Some VecGeom bugs found and fixed:
    (1) Trap, Hype, Orb VGDML parsing ignoring length and angle units;
    (2) Tet and BooleanVolume parameter dumping improved an homogenized;
    (3) Specific interpretation of EllipticalCone parameters (independently developed shape codes)

    → Several shapes added (tessellated, extruded, ellipsoid, ellipticals, generic polycones, etc.)

- Final validation test: detailed tracking, comparing boundary-crossing coordinates

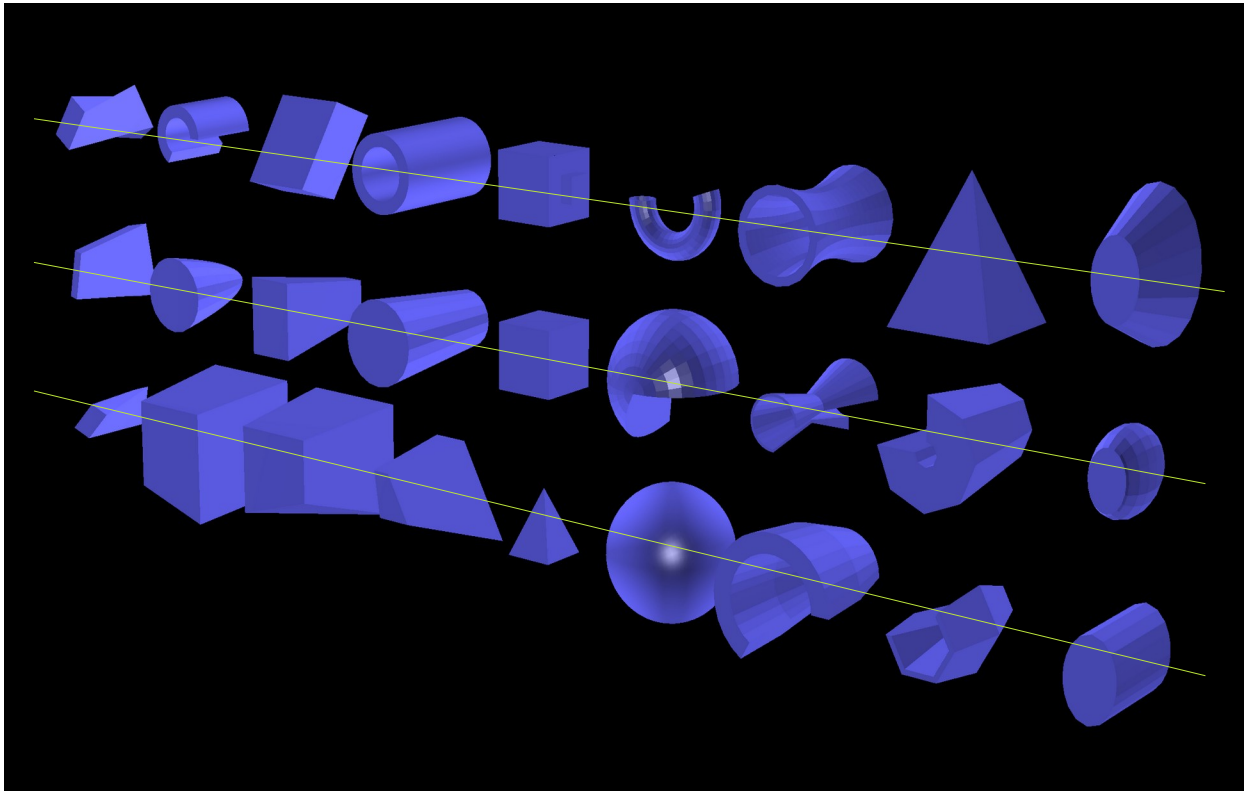
# Validation: original state of solids.gdml

- Evd: ROOT-based visualization tool, part of the celeritas infrastructure, was used to produce this picture
- Original solids.gdml from a Geant4 example, edited → at least one instance of each shape supported by the converter
- Limited number of shapes supported
- Very unequal dimensions and positioning / spacing



# Validation: current status of solids.gdml

- Added support to several shapes
- Similar shape dimensions and adequate positioning and spacing
- Detailed tracking, comparing coordinates of each boundaries crossed



# Back-porting into G4VecGeomNav package

- A preliminary version of the geometry converter has been back-ported into the G4VecGeomNav repository
  - Includes several new shapes and bug fixes on the converter itself
  - The converter validation unearthed a few bugs on VGDML parser, which are now fixed.
  - Another MR, including extensions for GPU compatibility, has not been merged yet.

- More updates after MRs above (with Seth Johnson):
  - More shapes added: reflected shapes, multi-union
  - Further testing and functionality → more VecGeom fixes

    → **Perfect agreement in CMS2018 tracking using converted VecGeom geometry**, based on detailed Celeritas vs. Geant4 tracking tests using ~9k 10GeV $e^-$ tracks x 7 starting seeds.
  - Re-factored and cleaned up for cleaner code and better performance
  - To be eventually back-ported to G4VecGeomNav module as well

# Geometry converter: status and plans

- In-memory Geant4-to-VecGeom geometry converter is now available
  - From a preliminary prototype in G4VecGeomNav, further developed under the Celeritas environment
  - Debugged, fixed, validated and released: produces in-memory VecGeom model
  - The VecGeom model is readily available for tracking in the GPU
  - Has been (partly) ported back into the G4VecGeomNav module
    - Still to be ported: reflected shapes, multi-union, simplifying refactoring

- Prospects:
  - New: surface-based geometry approach, still under development – see previous talks

    → expected to be supported by this converter, no roadblocks anticipated
  - More shapes to be added as needed (e.g. triggered by other complex GDML files used)
  - TBD: long-term repository (requirements, dependencies, constraints)

# Backup slides

# Some code illustration

```
class G4VecGeomConverter
{
  private:
    /** Remember pointer to generated world from imported G4 geometry. */
    VPlacedVolume const* world_;

    // one G4 physical volume can correspond to multiple vecgeom placed volumes
    // (in case of replicas)
    BidirectionalTypeMap<std::vector<VPlacedVolume const*> const*,…
    BidirectionalTypeMap<VUnplacedVolume const*, G4VSolid const*>…
    BidirectionalTypeMap<LogicalVolume const*, G4LogicalVolume const*>…
    // fast O(1) lookup to get VecGeom or G4 placed volume based on index
    // FastG4VecGeomLookup fFastG4VGLookup;
    std::vector<Transformation3D const*> replica_transformations_;

    int verbose_;

  public:
    void set_verbose(int const verbose) { verbose_ = verbose; }
    int get_verbose_level() const { return verbose_; }

    /// Get placed volume that corresponds to a G4VPhysicalVolume
    std::vector<VPlacedVolume const*> const*
    get_placed_volume(G4VPhysicalVolume const* n) const
    {…
    }

    /**
     * Queries the G4 geometry for the top volume and recursively
     * imports and converts to VecGeom geometry.
     */
    void convert_G4_geometry(G4VPhysicalVolume const*);

  private:
    /**
     * @brief Deletes all VecGeom geometry generated by this class.
     */
    void clear_vecgeom();
```

```
/** @brief Converts a physical volume into a VecGeom placed volume.
 * Its transformation matrix and its logical volume are also converted,
 * making the conversion process recursive, comprising the whole geometry
 * starting from the top volume.
 * Will take care not to convert anything twice by checking the
 * bidirectional map between Geant4 and VecGeom geometry.
 */
std::vector<VPlacedVolume const*> const* convert(G4VPhysicalVolume const*);

/**
 * @brief Special treatment needed for replicated volumes.
 */
void extract_replicated_transformations(…
/**
 * @brief Converts G4 solids into VecGeom unplaced volumes
 */
VUnplacedVolume* convert(G4VSolid const*);

/**
 * @brief Converts logical volumes from Geant4 into VecGeom.
 * All daughters' physical volumes will be recursively converted.
 */
LogicalVolume* convert(G4LogicalVolume const*);

/**
 * @brief Converts transformation matrices
 */
Transformation3D* convert(G4ThreeVector const&, G4RotationMatrix const*);
```

# Example code: simple G4Solid → UnplacedVolume conversions

```
// THE TUBESEG
else if (auto tube = dynamic_cast<G4Tubs const*>(shape))
{
    unplaced_volume
        = GeoManager::MakeInstance<UnplacedTube>(scale * tube->GetInnerRadius(),
                                                 scale * tube->GetOuterRadius(),
                                                 scale * tube->GetZHalfLength(),
                                                 tube->GetStartPhiAngle(),
                                                 tube->GetDeltaPhiAngle());
}

//
// THE CONESEG
else if (auto cone = dynamic_cast<G4Cons const*>(shape))
{
    unplaced_volume = GeoManager::MakeInstance<UnplacedCone>(
        scale * cone->GetInnerRadiusMinusZ(),
        scale * cone->GetOuterRadiusMinusZ(),
        scale * cone->GetInnerRadiusPlusZ(),
        scale * cone->GetOuterRadiusPlusZ(),
        scale * cone->GetZHalfLength(),
        cone->GetStartPhiAngle(),
        cone->GetDeltaPhiAngle());
}
```

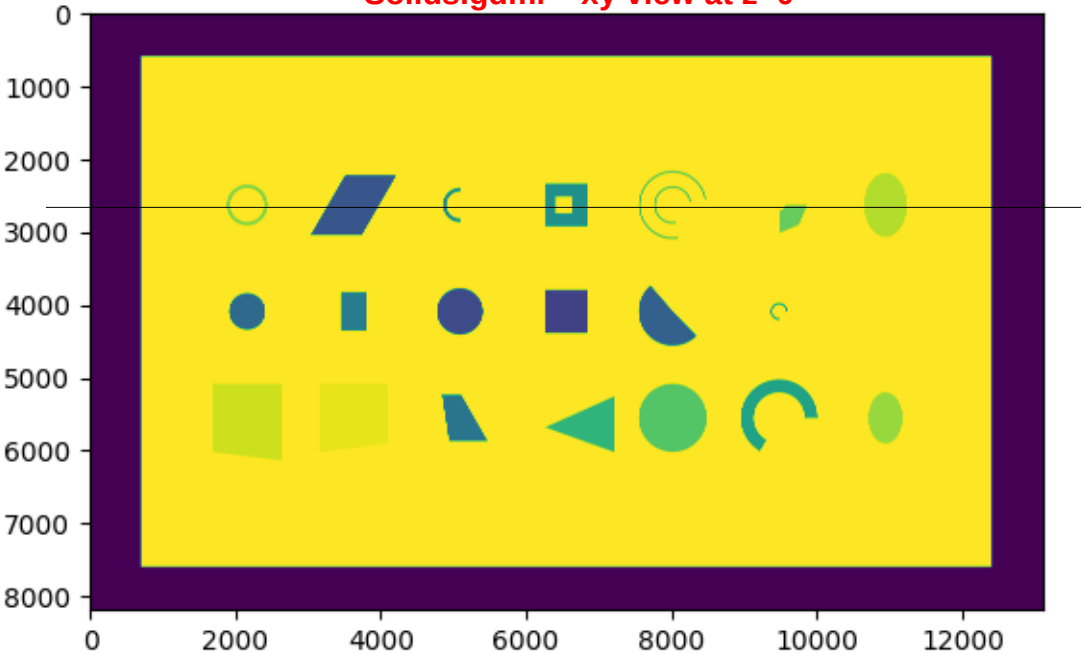
# Ultimate validation: full tracking

- In the simple geometry of solids.gdml, send tracks through several volumes, then perform full tracking
- Repeat this tracking for VGDML and converted geometries based on same input GDML file
- Compare detailed distances between volumes, and volume IDs as they are traversed

→ Fix to GDML files: ensure length and angle units always defined for all shapes (otherwise different default units in Root (evd resizing) and Geant4+Celeritas (tracing)

- The figures show the result of the celeritas rasterizer, which sends thousands of parallel tracks to “scan” the volumes as they are traversed.
  - All the tracking is fully performed on the GPU.
  - Volume IDs are used for coloring each part of each track, composing a “cut view” of the geometry

Solids.gdml – xy view at z=0




Solids.gdml – xz view at y = 125 (top shapes)