Pyg4ometry: Geometry preparation (manipulation, editing, compositing) for Geant4 / FLUKA

Stewart Boogert and Laurie Nevay

HSF Meeting 7th June 2021

Zoom

Andrey Abramov, William Shields, Benjamin Shellswell , Stuart Walker

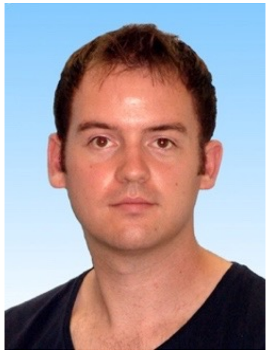
https://bitbucket.org/jairhul/pyg4ometry/src/develop/
http://www.pp.rhul.ac.uk/bdsim/pyg4ometry/

Introduction

Stewart Boogert

- Accelerator physicist (beam instrumentation, ILC, simulations)
- HEP PhD and post-doc (ZEUS@HERA)

Laurie Nevay



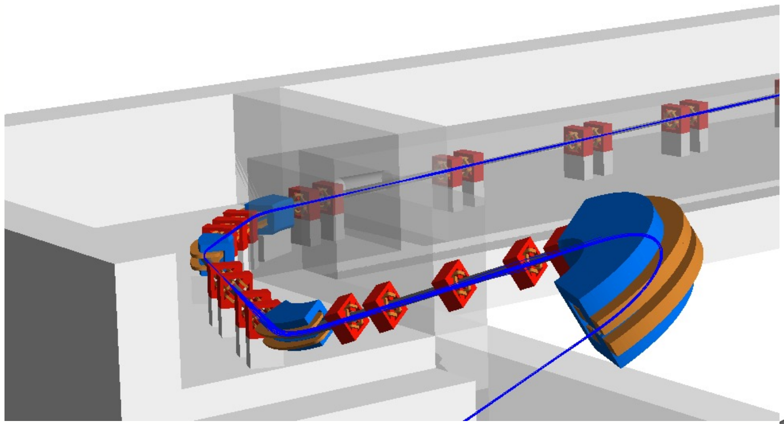
- Senior post-doc
- Background in accelerator beam instrumentation, high power fibre lasers
- Lead developer of BDSIM - Geant4 application for accelerator models

Royal Holloway Physics Department:

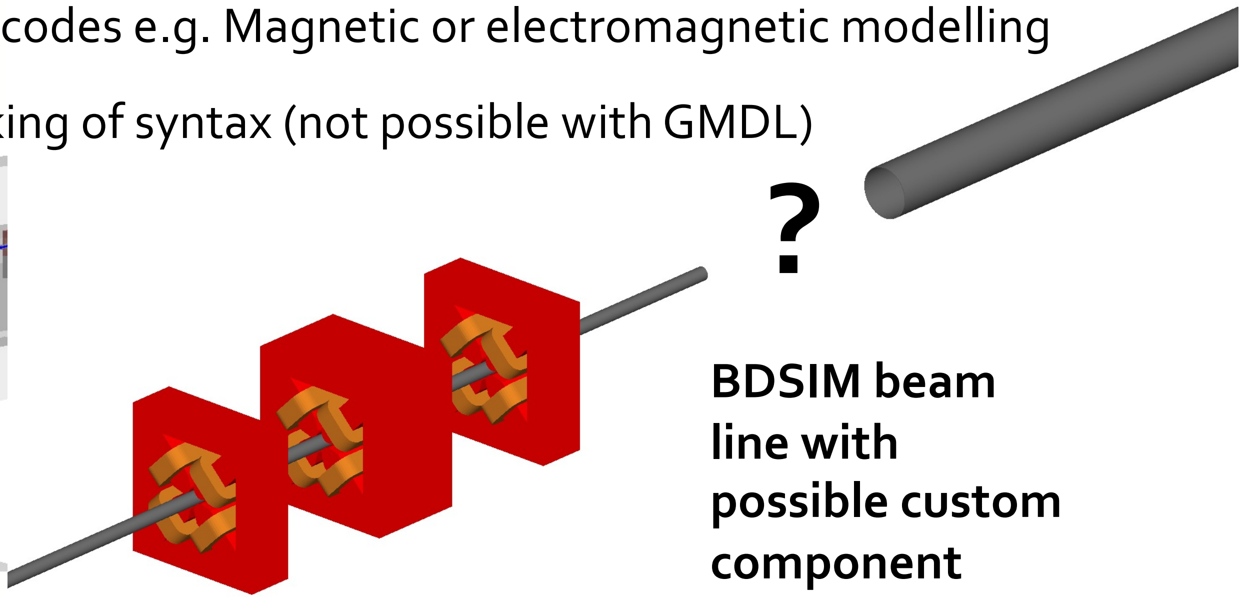
- Particle Physics & Condensed Matter
- PP: Theory, ATLAS, Dark Matter & Neutrinos, Accelerators (LHC + medical)

Beam delivery simulation (BDSIM)

- RHUL group has developed BDSIM, a code to make Geant4 accelerator models
  - [Computer Physics Communications (252), July 2020, 107200](#) http://www.pp.rhul.ac.uk/bdsim
  - Presented at 3rd Geant4 International User Conference 2018 (http://geant4.in2p3.fr/2018/)
- Want to insert custom components / customise models
  - Geometry preparation takes a long time
  - Needed to make geometry preparation as quick as possible to compliment BDSIM
- Create geometry from other codes e.g. Magnetic or electromagnetic modelling
- Interpreter or compiler checking of syntax (not possible with GMDL)



BDSIM screenshot



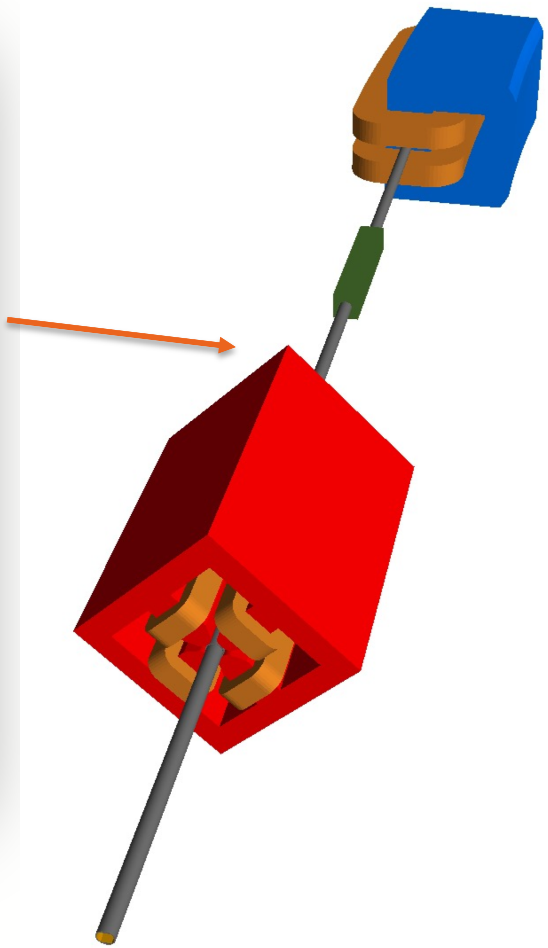
BDSIM beam line with possible custom component

Example BDSIM Syntax

"GMAD" - Geant4 + MAD (MAD is an accelerator optics code)

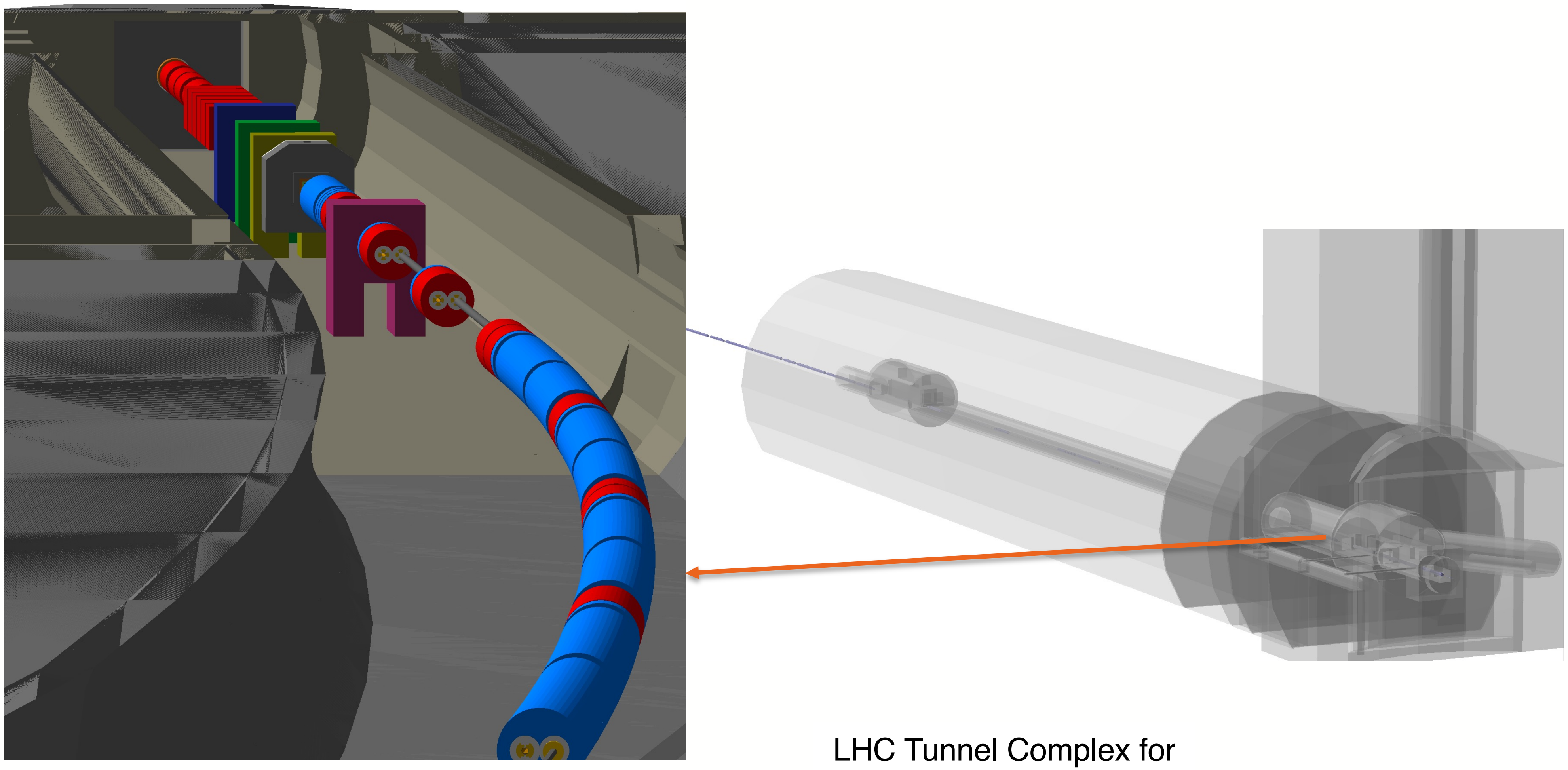
```
d1: drift, l=1*m;
q1: quadrupole, l=1*m, k1=1.3, magnetGeometryType="polesfacet";
c1: rcol, l=0.6*m, ysize=5*mm, xsize=5*mm, material="Copper", outerDiameter=10*cm;
s1: sbend, l=1*m, angle=0.10;

l1: line = (d1, q1, d1, c1, d1, s1);
use,period=l1;

sample, all;

option, ngenerate=1,
        physicsList="em";

beam, particle="proton",
      energy=10.0*GeV,
      X0=0.001,
      Y0=0.001;
```

```
-:---  sm.gmad      All L3    Git:develop  (Fundamental)
Wrote /Users/nevay/physics/reps/bdsim/examples/simpleMachine/sm.gmad
```



CERN IR1 example

H. Lefebvre (FASER) and S. Walker (ATLAS)



LHC Tunnel Complex for
accelerator background
in detector

History and more rationale

- RHUL group has developed BDSIM: a code to convert accelerator descriptions to Geant4 (geometry, material, fields etc.)
- Geometry preparation takes a long time
- Need to make geometry preparation as quick as possible to compliment BDSIM
- Create geometry from other codes e.g. Magnetic or electromagnetic modelling
- Interpreter or compiler checking of syntax (not possible with GDML)
- Rich language for creating geometry
- Parametrized construction (length safety)
- Lots of geometry exists in Flair/Fluka, STL and CAD
- Physics comparisons between Geant4 and Fluka (or other code)
- Facilitate geometry reuse and modification
- Started life as an internal group tool to accelerate geometric model development
- Set of python classes to aid geometry generation → **PYG4OMETRY**

Requirements

ROYAL HOLLOWAY UNIVERSITY OF LONDON

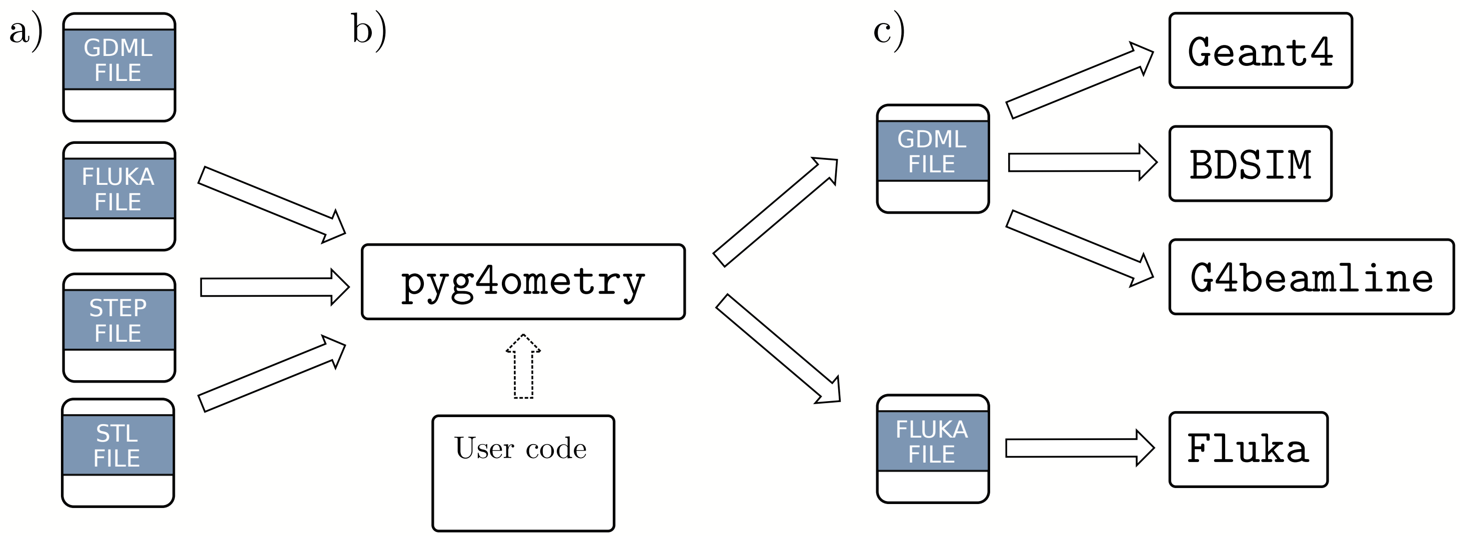
- Load (and convert) GDML, STL, STEP, FLUKA files
- Complete support (reading/writing) of GDML
- Visualize geometry
- Check for overlaps and geometry issues
- Composite (load and place) geometry from different sources
- Rendering for data analysis
- Modify geometry (cut holes, remove material etc.)
- Testing ground for Geant4 developments
- Leverage modern tools and programming
- Lightweight
- Open source and simple to install
- Simple to use API (think of a summer student)
- Simple to contribute to (think of a PhD student)
- Reasonable performance (I could not render ATLAS in GDML, partly an issue with the GDML)

Legitimate questions

- **Why not just write C++ using Geant4 API?**
- Compilation cycle is comparatively long (5 mins)
- Hard to debug geometry in some instances (voxelization will crash because of overlaps, but how to find the overlaps)
- **Why don't you just include this functionality in ROOT?**
- Not all users of Geant4 are particle physics experts
- Hard to prototype in ROOT and scripting languages are quick for ECRS to pick up and use
- Lots of packages exist with python bindings and can be collected under pyg4ometry
- **Why don't you just expand Geant4?**
- This is already being done and VTK is being developed as a visualization driver
- CGAL Boolean processing is already implemented and performs well compared existing G4 implementation
- **Why don't you just write GDML?**
- Quite hard to debug when bugs are introduced

Guiding principles and implementation

- Follow patterns of Geant4 (object interfaces, methods and internal data)
- Use GDML as a fundamental file description of geometry
- Use existing codes/libraries wherever possible
- Aim for 100% test coverage
- **Create python class representation for geometric data (other data too)**



Technology tools and dependencies

All dependencies are all open source and well maintained

Python

ANTLR

Visualisation Tool kit

FreeCAD

NGSolve

FEniCS

Open CASCADE TECHNOLOGY

SymPy

Important design features (G4/GDML)

- GDML interpreted via ANTLR grammar
- Dynamic (late) evaluation of expressions
- Complete coverage of GDML including mathematical expressions
- All different types of physical volume (placement, replica, parametrized etc)
- All solids (G4) can generate a tri/quad mesh
- Simple Boolean (union, intersection, subtraction) library based on BSP trees
- Meshes created once per LV and placed as instances in rendering pipeline (ok different for param and replica volumes)
- Scene tree created from PV-LV tree

Geant4 (python) example

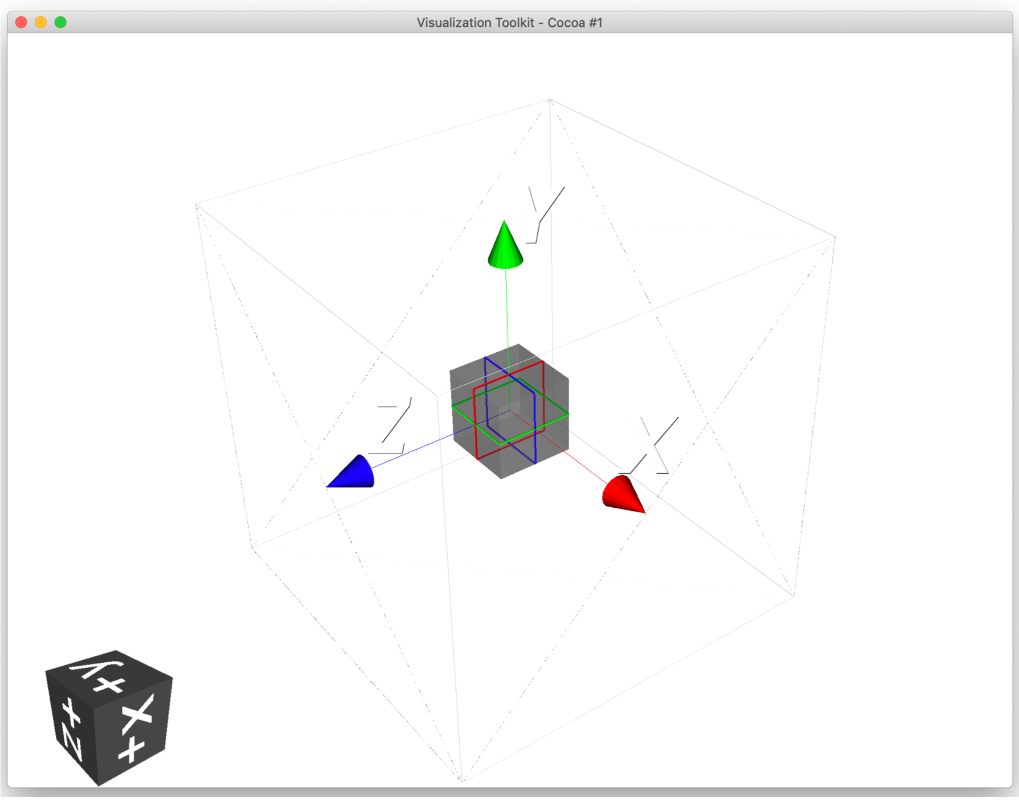
```
import os as _os
import pyg4ometry.gdml as _gd
import pyg4ometry.geant4 as _g4
import pyg4ometry.visualisation as _vi


def Test(vis = False, interactive = False) :
    reg = _g4.Registry()

    # defines
    wx = _gd.Constant("wx","50",reg,True)
    wy = _gd.Constant("wy","50",reg,True)
    wz = _gd.Constant("wz","50",reg,True)

    bx = _gd.Constant("bx","10",reg,True)
    by = _gd.Constant("by","10",reg,True)
    bz = _gd.Constant("bz","10",reg,True)

    wm = _g4.MaterialPredefined("G4_Galactic")
    bm = _g4.MaterialPredefined("G4_Fe")

    # solids
    ws = _g4.solid.Box("ws",wx,wy,wz, reg, "mm")
    bs = _g4.solid.Box("bs",bx,by,bz, reg, "mm")

    # structure
    wl = _g4.LogicalVolume(ws, wm, "wl", reg)
    bl = _g4.LogicalVolume(bs, bm, "bl", reg)
    bp = _g4.PhysicalVolume([0,0,0],[0,0,0],  bl, "b_pv1", wl, reg)

    # set world volume
    reg.setWorld(wl.name)

    # gdml output
    w = _gd.Writer()
    w.addDetector(reg)
    w.write(_os.path.join(_os.path.dirname(__file__), "T001_Box.gdml"))
    w.writeGmadTester(_os.path.join(_os.path.dirname(__file__),"T001_Box.gmad"),"T001_Box.gdml")

    # test __repr__
    str(bs)

    # test extent of physical volume
    extentBB = wl.extent(includeBoundingSolid=True)
    extent   = wl.extent(includeBoundingSolid=False)

    # visualisation
    v = None
    if vis :
        v = _vi.VtkViewer()
        v.addLogicalVolume(reg.getWorldVolume())
        v.addAxes(_vi.axesFromExtents(extentBB)[0])
        v.view(interactive=interactive)


    return {"testStatus": True, "logicalVolume":wl, "vtkViewer":v}
```

Defines

Materials

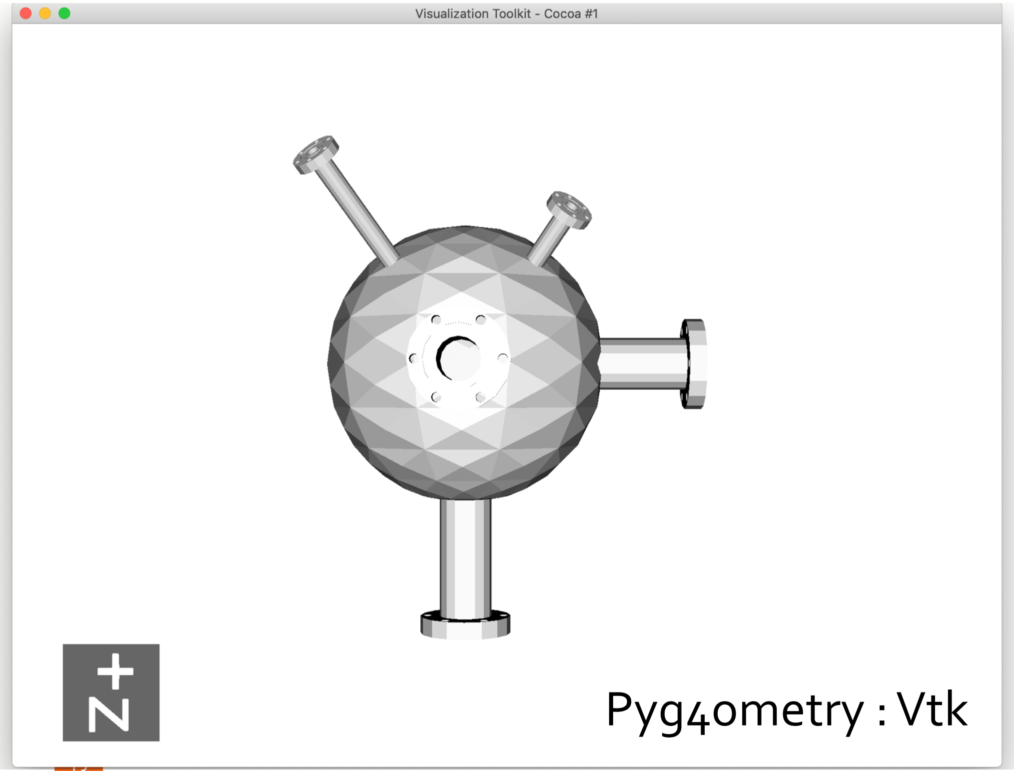
Solids

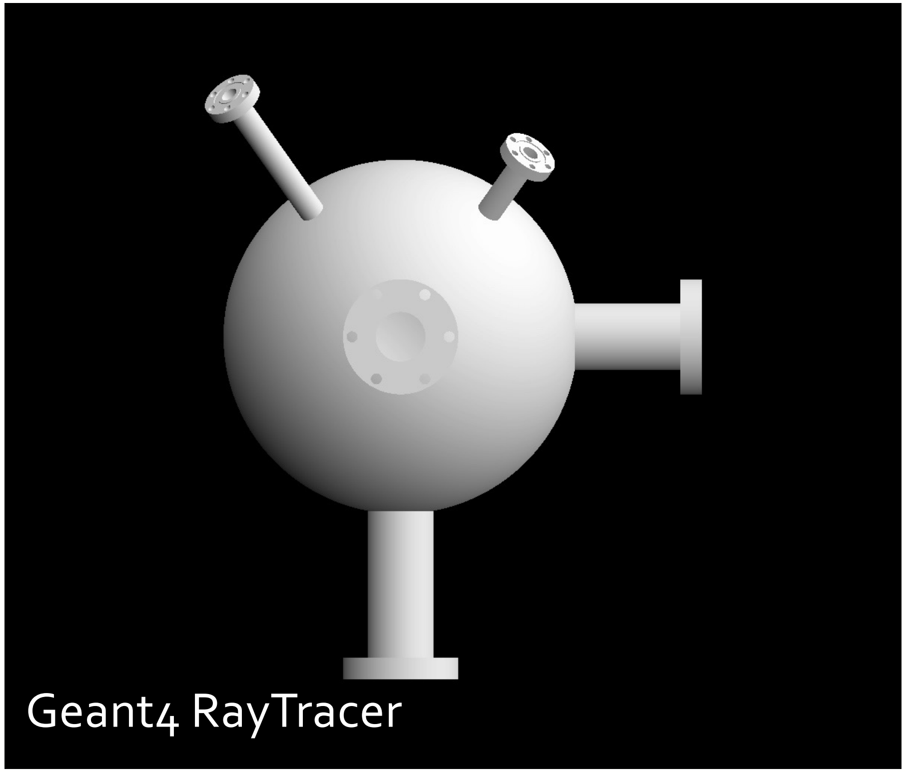
Structure



Geant4 (advanced python) example

- Generic python vacuum chamber builder (CF). Arbitrary sphere with arbitrary number of ports (flanges, beam pipes, spherical chamber)
- Pyg4ometry code : 229 lines
- GDML code : 385 lines

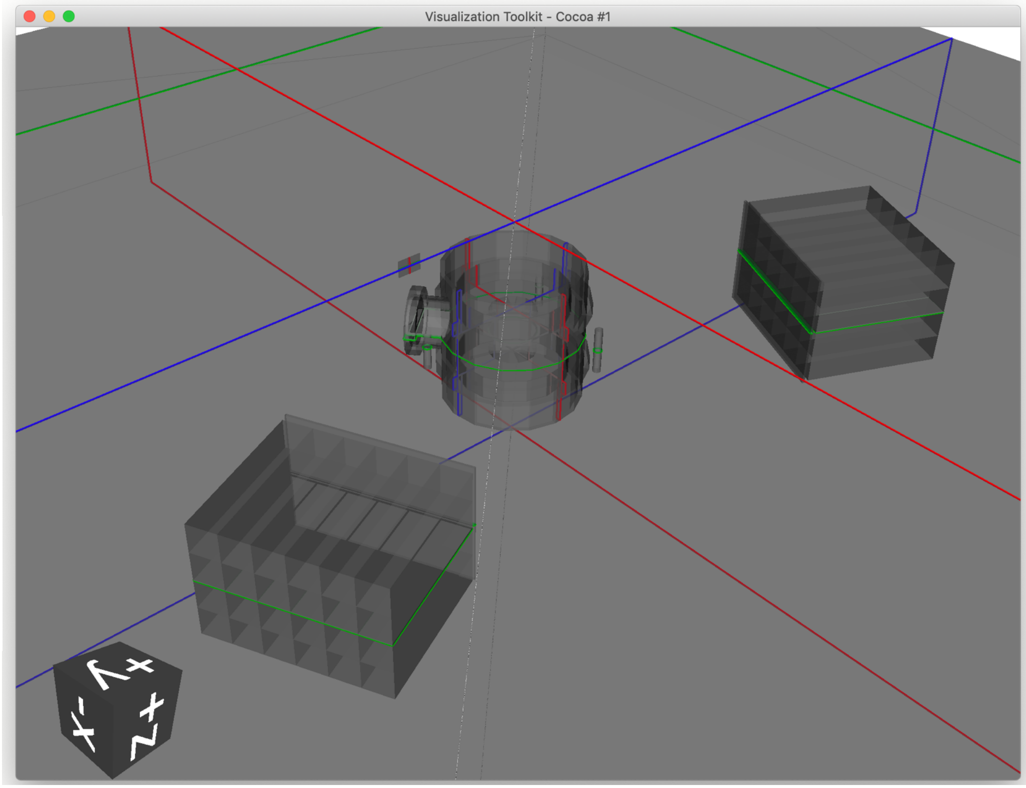




Geant4 (GDML) example

- Load of the GDML examples distributed with Geant4
  - Take a more complex example

```
[In [1]: import pyg4ometry
 FreeCAD 0.17, Libs: 0.17RUnknown

[In [2]: r = pyg4ometry.gdml.Reader("./lht_fixed.gdml")

[In [3]: l = r.getRegistry().getWorldVolume()

[In [4]: v = pyg4ometry.visualisation.VtkViewer()

[In [5]: v.addLogicalVolume(l)

[In [6]: v.view()
```



- Whole file is loaded and can be manipulated in the python terminal
- Dimensions can be changed, holes cut etc.

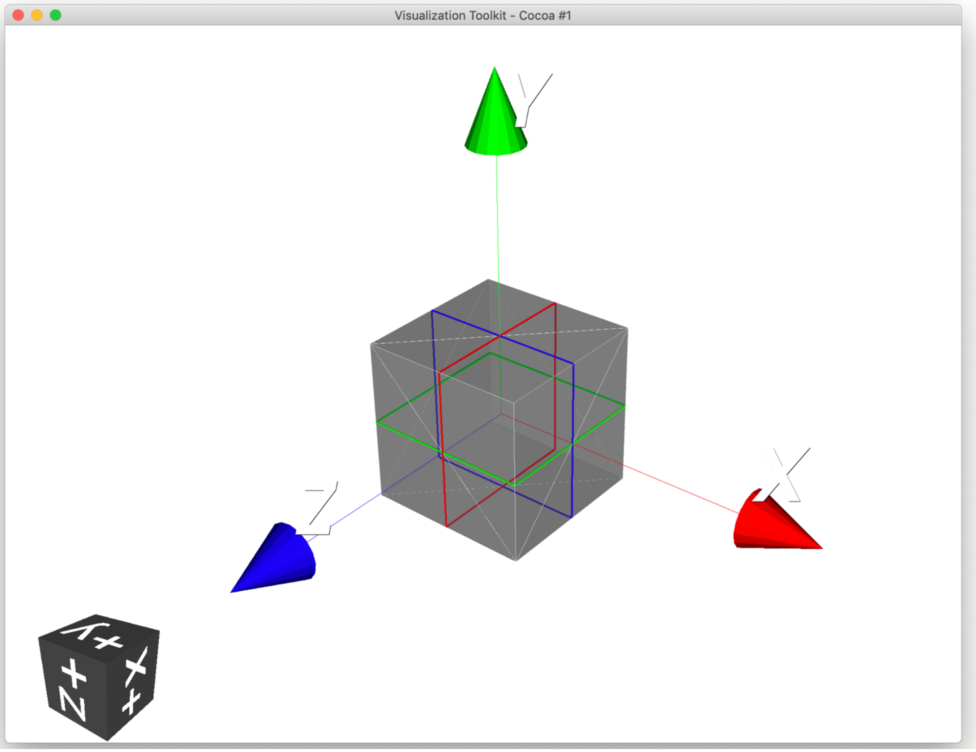
Important design features (Fluka ⟶ Geant4-GDML)

- For each Fluka body create a G4/GDML solid
- Create large but finite G4/GDML solids instead of Fluka infinite solids
- Modify the sizes of bodies to create a length safety between solids
- Create CSG tree from Fluka regions
- Determine is CSG tree creates disjoint solids
- Shrink large solids once extent of Fluka region is determined

Fluka (python) to Geant4-GDML example

- In a very similar way to GDML, classes are created to represent Fluka concepts

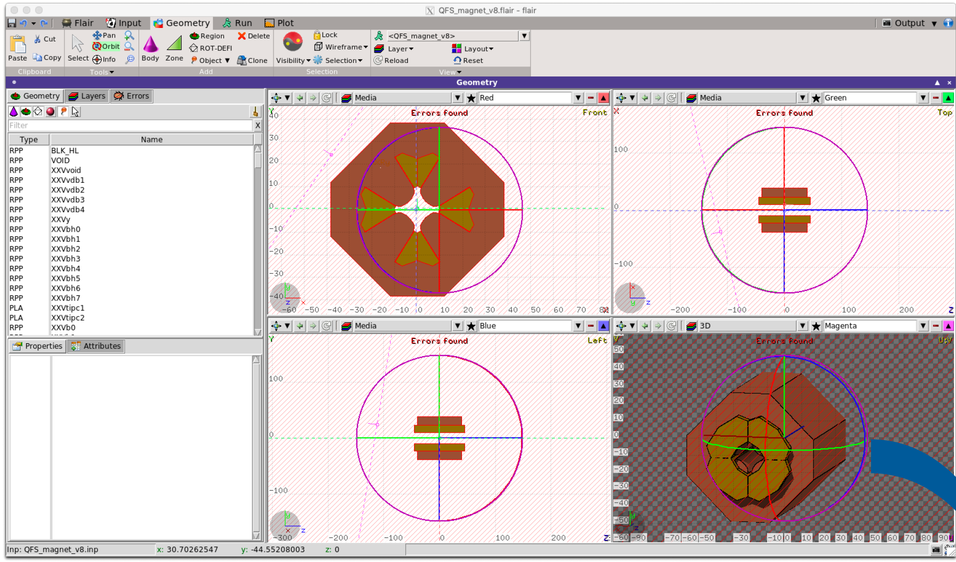
```
import pyg4ometry.convert as convert
import pyg4ometry.visualisation as vi
from pyg4ometry.fluka import RPP, Region, Zone, FlukaRegistry

def Test(vis=False, interactive=False):
    freg = FlukaRegistry()

    rpp = RPP("RPP_BODY", 0, 10, 0, 10, 0, 10, flukaregistry=freg)
    z = Zone()
    z.addIntersection(rpp)
    region = Region("RPP_REG", material="COPPER")
    region.addZone(z)
    freg.addRegion(region)

    greg = convert.fluka2Geant4(freg)

    greg.getWorldVolume().clipSolid()

    v = None
    if vis:
        v = vi.VtkViewer()
        v.addAxes(length=20)
        v.addLogicalVolume(greg.getWorldVolume())
        v.view(interactive=interactive)

    return {"testStatus": True, "logicalVolume": greg.getWorldVolume(), "vtkViewer":v}
```

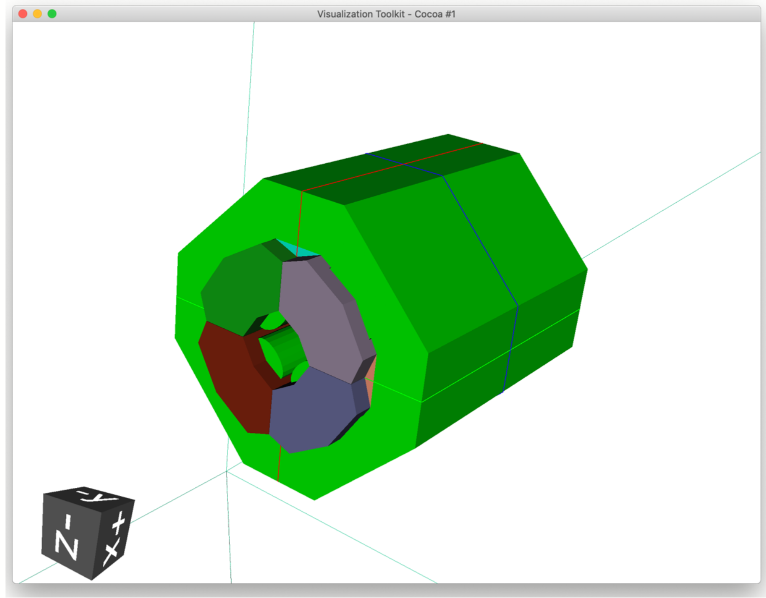


Fluka (Flair) to Geant4-GDML example

- Magnet created by CERN-RHUL PhD student (Gian Luigi D'Alessandro EN-EA-LE / KLEVER)

```
In [1]: import pyg4ometry
FreeCAD 0.17, Libs: 0.17RUnknown

In [2]: r = pyg4ometry.fluka.Reader("./QFS_magnet_v8.inp")

In [3]: greg = pyg4ometry.convert.fluka2Geant4(r.flukaregistry)

In [4]: wl = greg.getWorldVolume()

In [5]: v = pyg4ometry.visualisation.VtkViewer()

In [6]: v.addLogicalVolume(wl)

In [7]: v.setOpacity(1)

In [8]: v.setRandomColours()

In [9]: v.view()
```



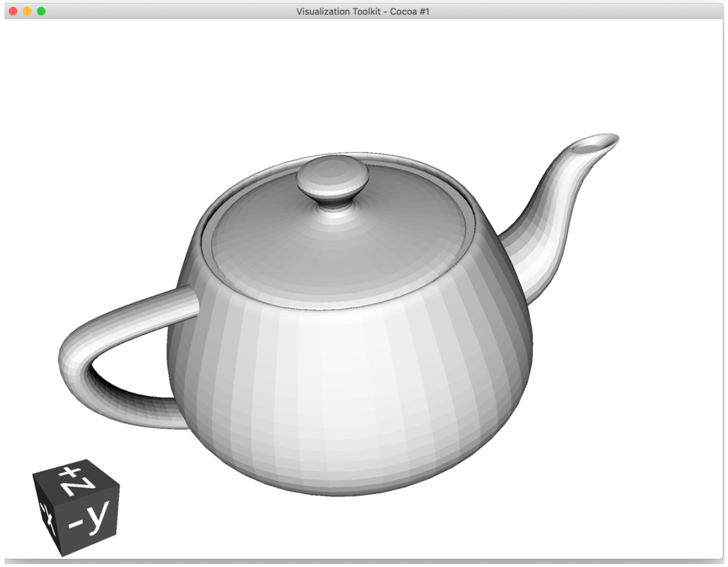


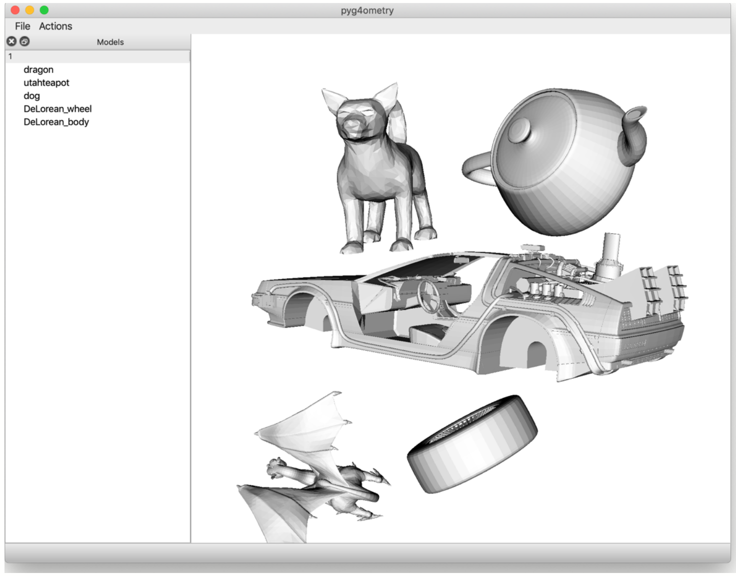
STL (tessellated solid) example

- Standard Tessellation Language
- Many 3D authoring programmes will produce STL
- Difficult format for GDML

```
In [1]: import pyg4ometry
FreeCAD 0.17, Libs: 0.17RUnknown

In [2]: reg = pyg4ometry.geant4.Registry()

In [3]: r = pyg4ometry.stl.Reader("./utahteapot.stl")

In [4]: l = r.logicalVolume("test","G4_Cu",reg)

In [5]: v = pyg4ometry.visualisation.VtkViewer()

In [6]: v.addLogicalVolume(l)

In [7]: v.view()
```

- Incorperate MeshCAD or other tools?

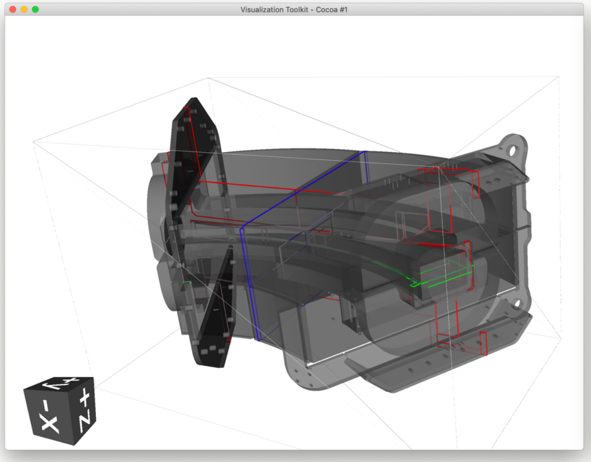


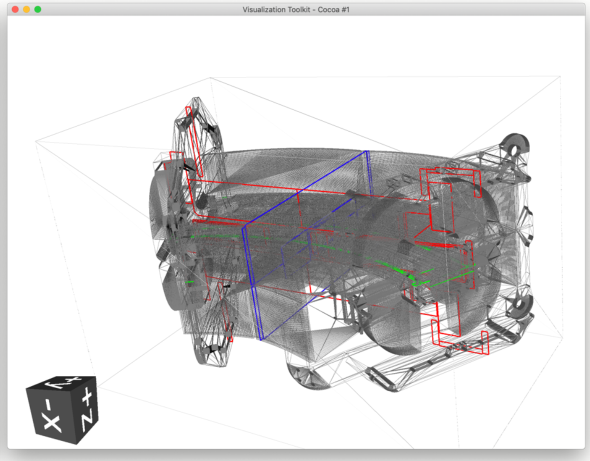


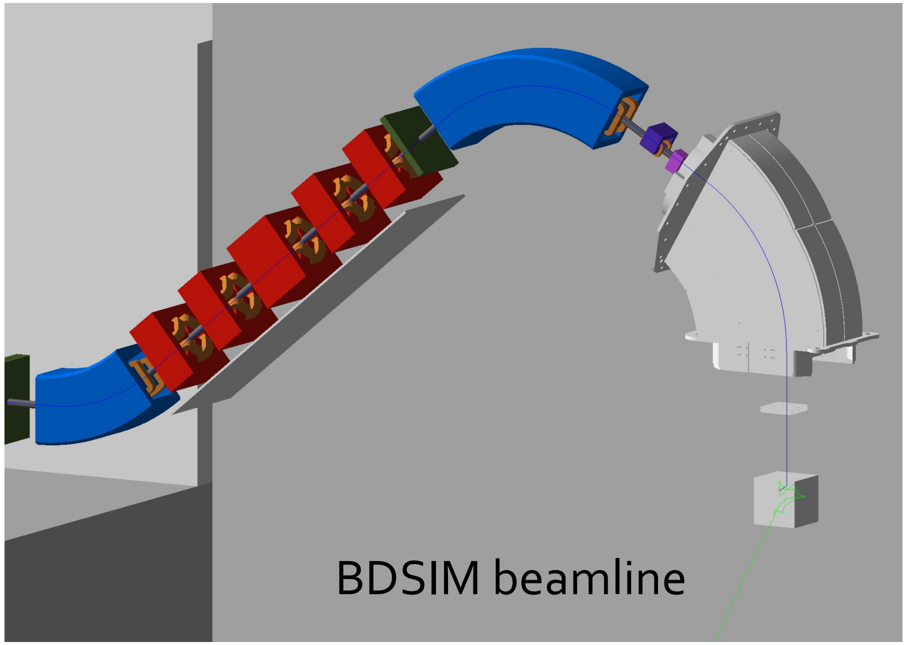
STP/STEP to Geant4-GDML example

ROYAL HOLLOWAY UNIVERSITY OF LONDON

- Load STEP file using FreeCAD-OpenCascade
- Still need to simplify CAD file
- Bodies and Parts map well to LV and PVs respectively. Convert bodies to triangulated mesh and place
- Based on STL loading
- Very advanced proof of principle (working in our workflow)
- Need to account for material
- (what about CadMesh, DagMC, McCAD)



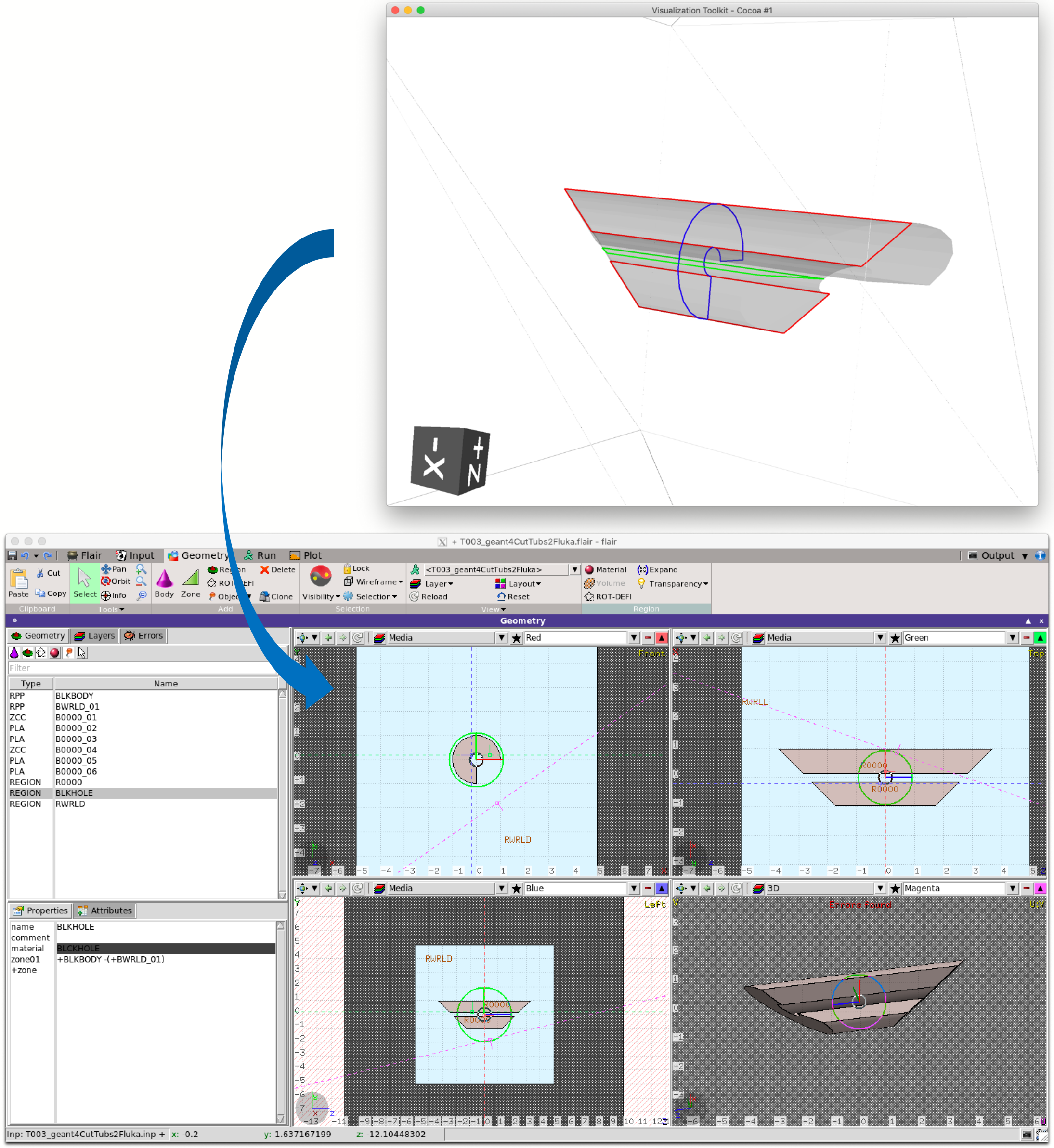




BDSIM beamline

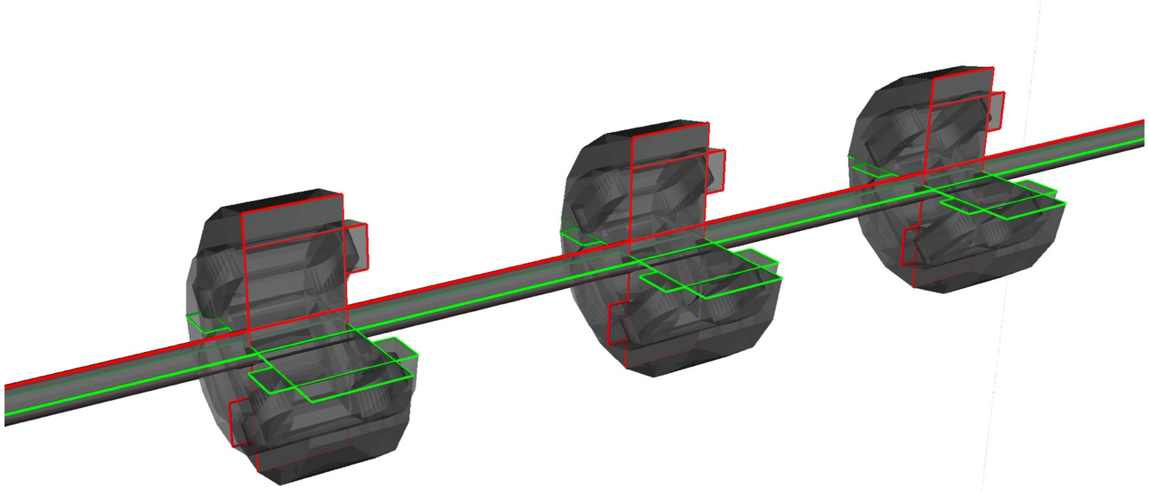
Geant4-GDML to Fluka example (**Primitive**)

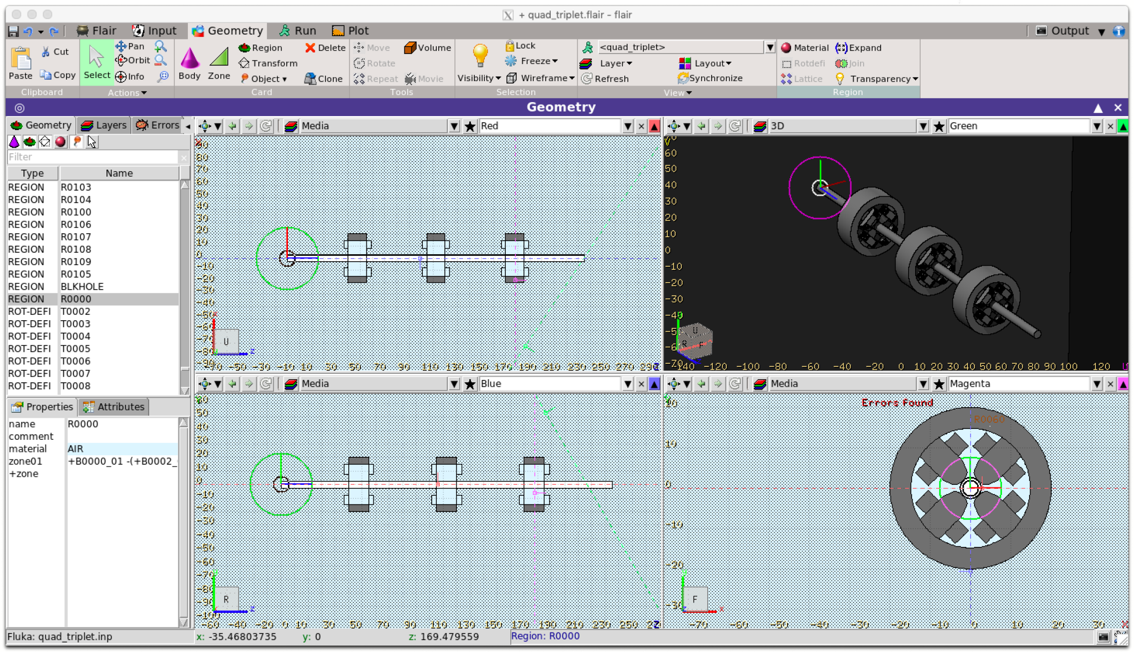
- Decompose G4-GDML primitive solids to Fluka bodies, then zones and join into regions
- Cut daughter volumes from mother to create flat hierarchy
- Working through G4 solids
- Need to implement union, intersection and subtraction G4 solids
- Least developed area of pyg4ometry but making rapid progress
- Scales an issue [-1,1,1] as does not exist in Fluka



Geant4-GDML to Fluka example (**Medium**)

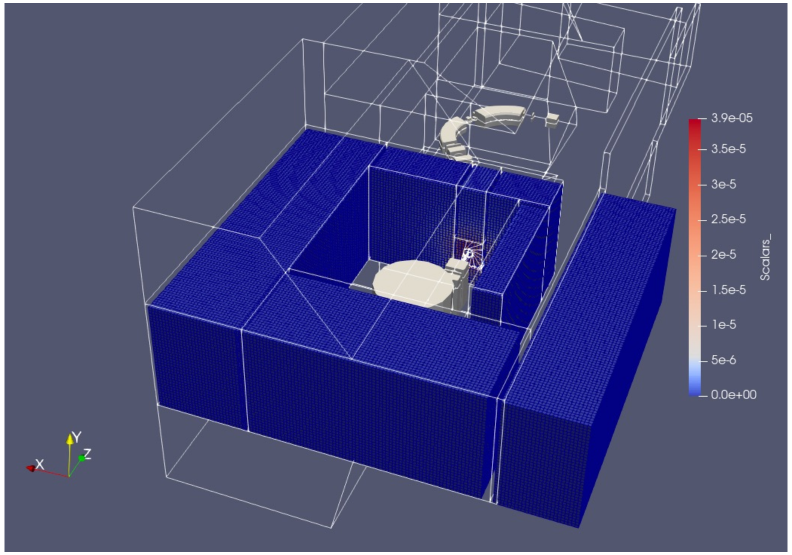
- Possible to convert larger models at the scale of a small experimental region
- Complex solids, CSG trees, booleans
- Issue with large CSG trees in G4 converting to Fluka
- Fluka geometry in form of disjunctive normal form (DNF) and can see large blow (computation time and memory) up when converting from Geant4

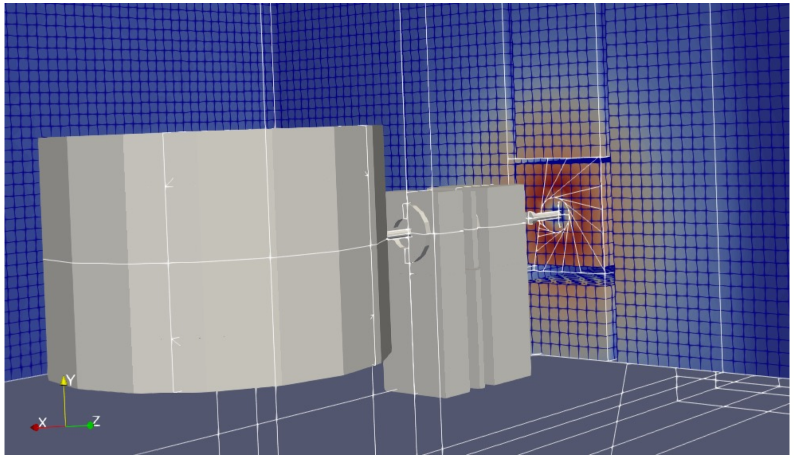




Geant4-GDML to **Paraview**

- Cedric Hernalsteens (CERN), Robin Tesse (ULB)
- Example of proton therapy system from Ion Beam Applications (IBA)
- Another potential target for 3D data is Paraview (built on VTK)
- “Industry” standard for visualisation of 3D data
- Use geometry data from pyg4ometry and output from **Geant4**/Fluka





Overlap detection example

Three classes in general

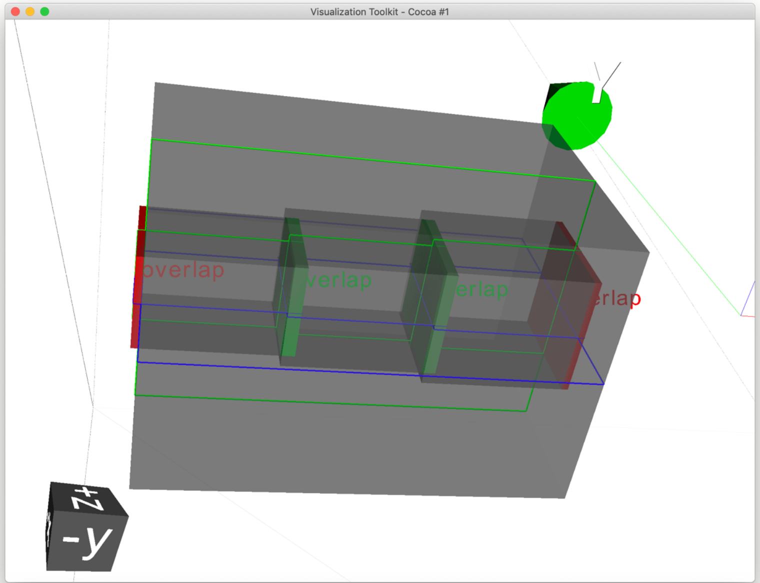
- Protrusion
- Daughter overlap
- Coplanar faces (can be a problem)

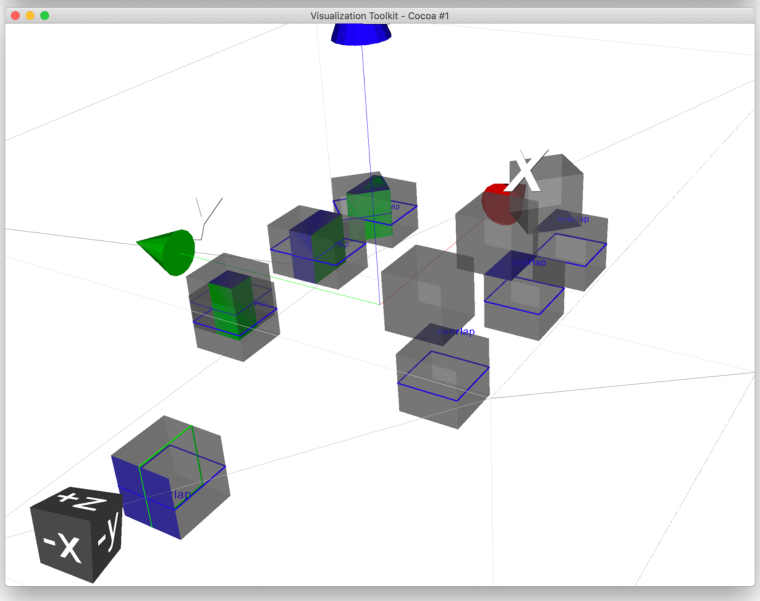
First two easily dealt with CSG operations (intersection)

Coplanar faces needs a dedicated algorithm (back up slides)

Search strategy

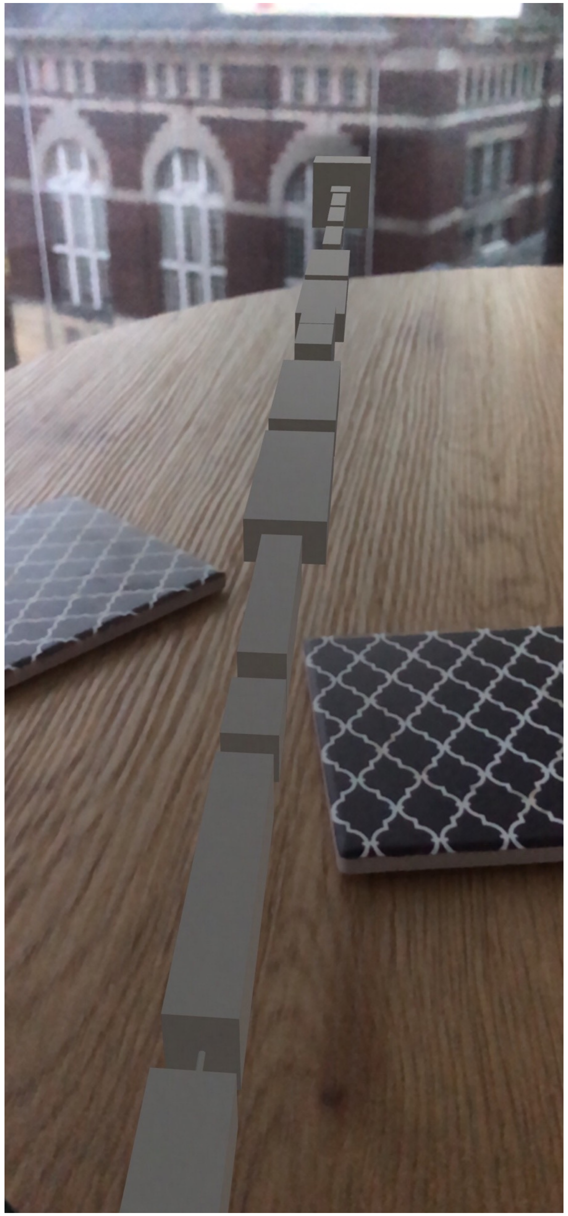
- Between daughters of LV
- Between daughters and LV solid





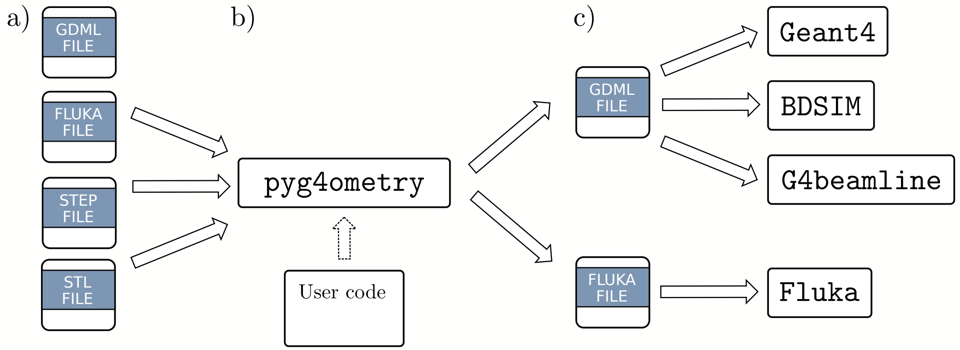
Geant4-GDML to **AR/VR**

- Have 3D mesh description for Geant4 and Fluka geometry
- Have possibility to create Augmented and Virtual reality models of beamlines and detectors
- Great potential public engagement and outreach potential
- Overlay particles, energy deposits etc. on AR/VR world
- Tools are being much more available (Unity, Unreal, **USDZ** file format)
- **Have running Geant4 on iOS and Android and prototype of Filament (google PBR) viewer : AR-Geant4!!**
- Open on your iPhone browser (or Mac)
- **www.pp.rhul.ac.uk/~sboogert/USDZ/K12_Small.usdz**

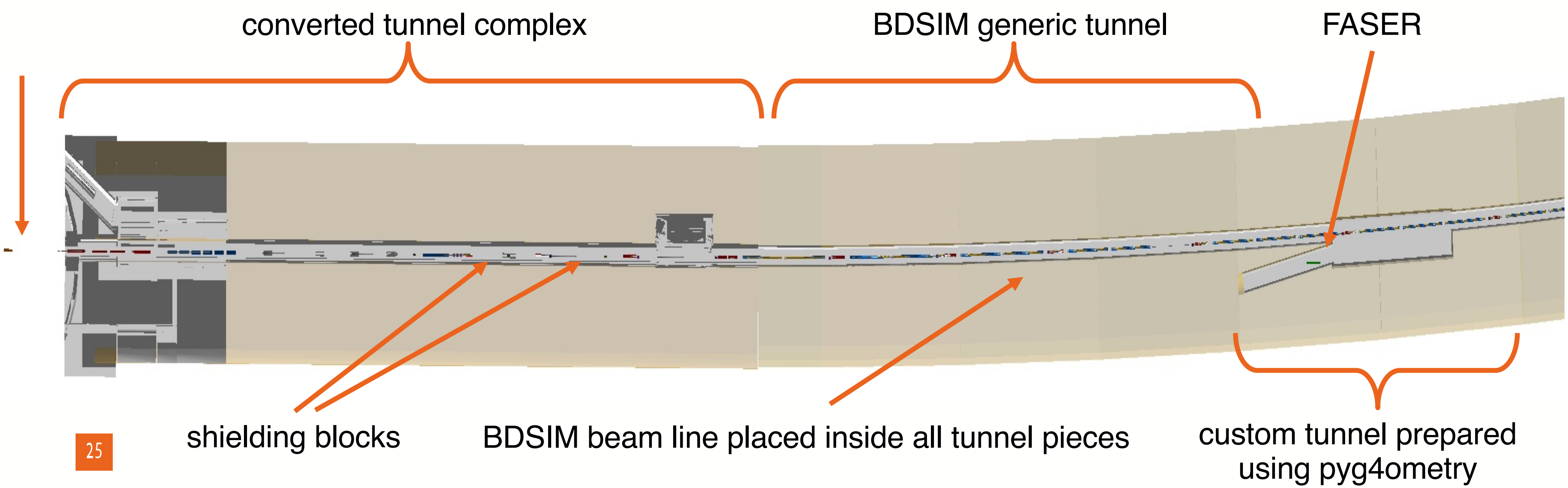


Compositing example

- A powerful workflow could be to take geometry from **multiple different sources** and composite in a GDML file



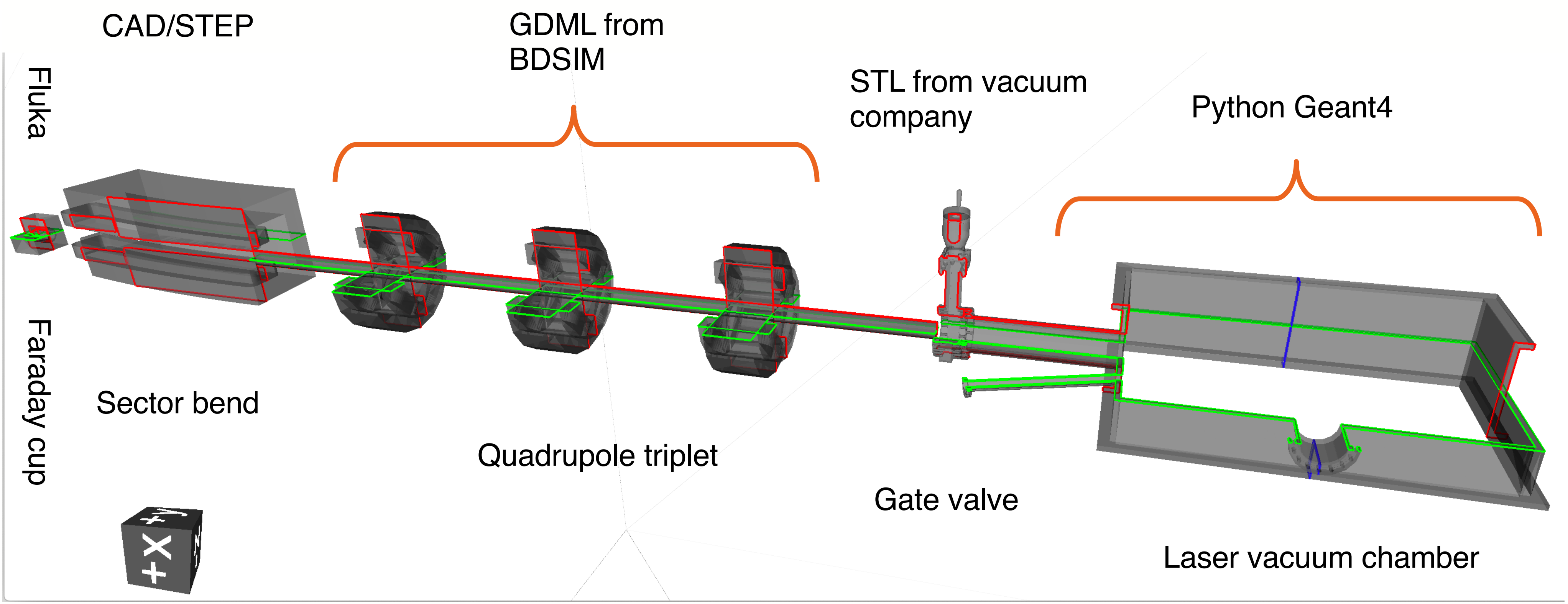
- Convert LogicalVolumes to Assembly volumes (excellent for placement)
- Intelligently merge two GDML files (dealing with name clashes)
- Example where Fluka/Geant4/GDML is used



Compositing example (Injector)

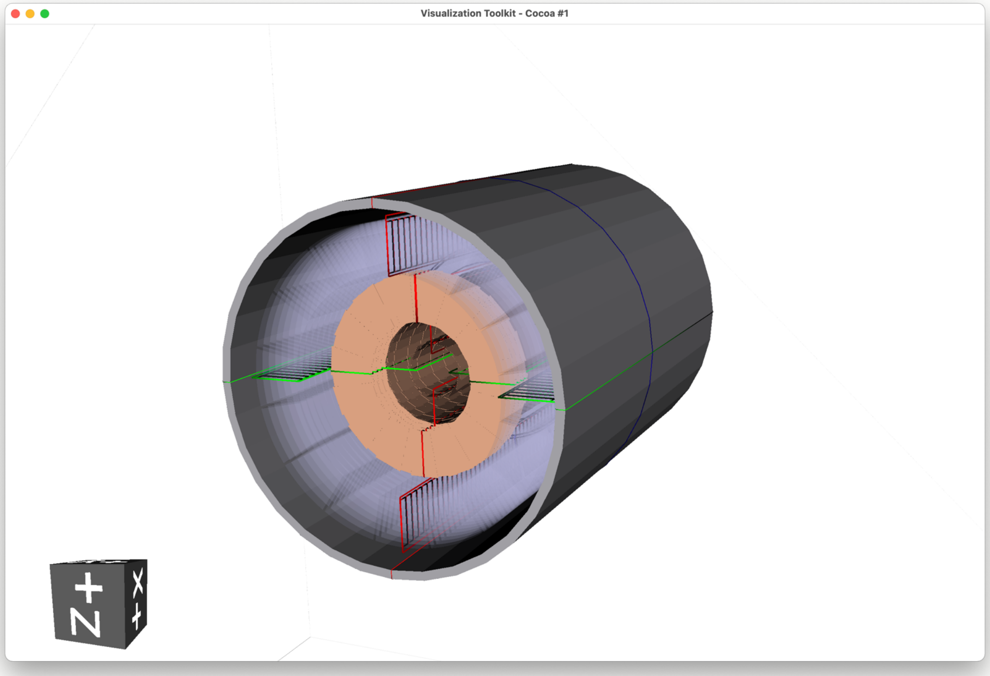
ROYAL HOLLOWAY UNIVERSITY OF LONDON

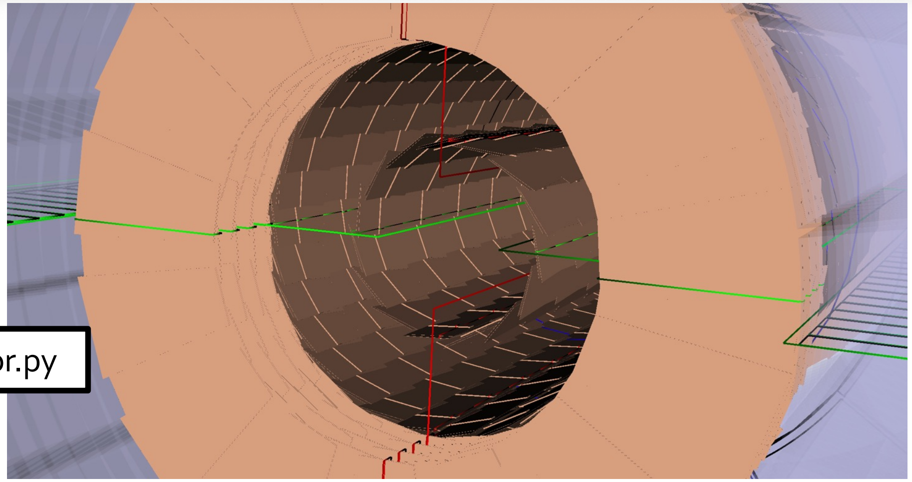
- Example using all geometries (CAD, STL, Fluka, Python and GDML)
- Imaginary photo-injector with different components



HEP Detector (1)

- Hypothetical example of detector
- Silicon tracker, solenoid and ECAL
- Written in Python using pyg4ometry
  - ~360 lines of Python
  - ~5500 output lines of gdml
- Functions for each sub-detector
  - programmatically designed
- About 8 hours of work
- Constants / variables propagate through python expressions to final GDML for parameterised output



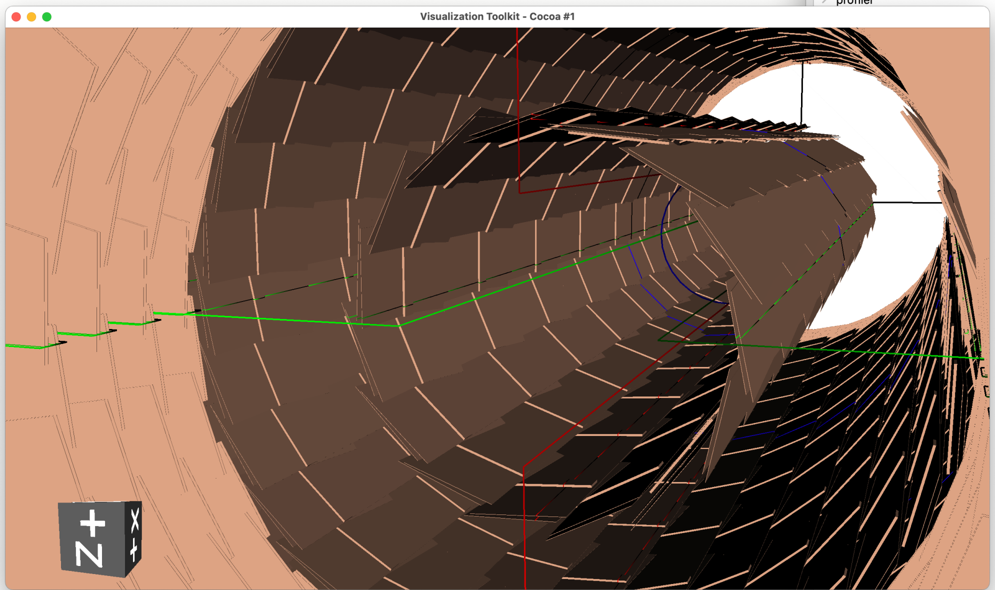


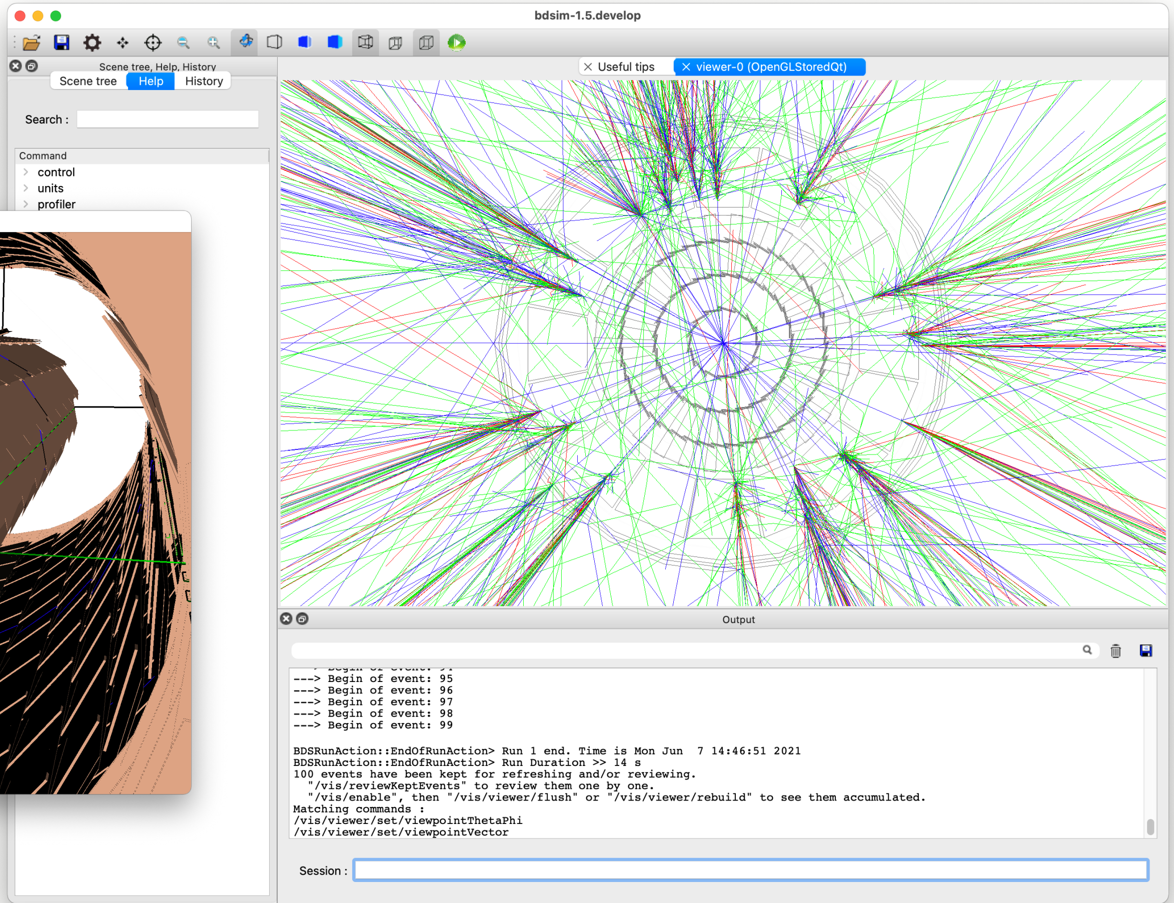
pyg4ometry/pyg4ometry/test/pythonCompoundExamples/HepDetector.py

HEP Detector (2)

ROYAL HOLLOWAY UNIVERSITY OF LONDON

In VTK in Python





In Geant4

"Good" house keeping

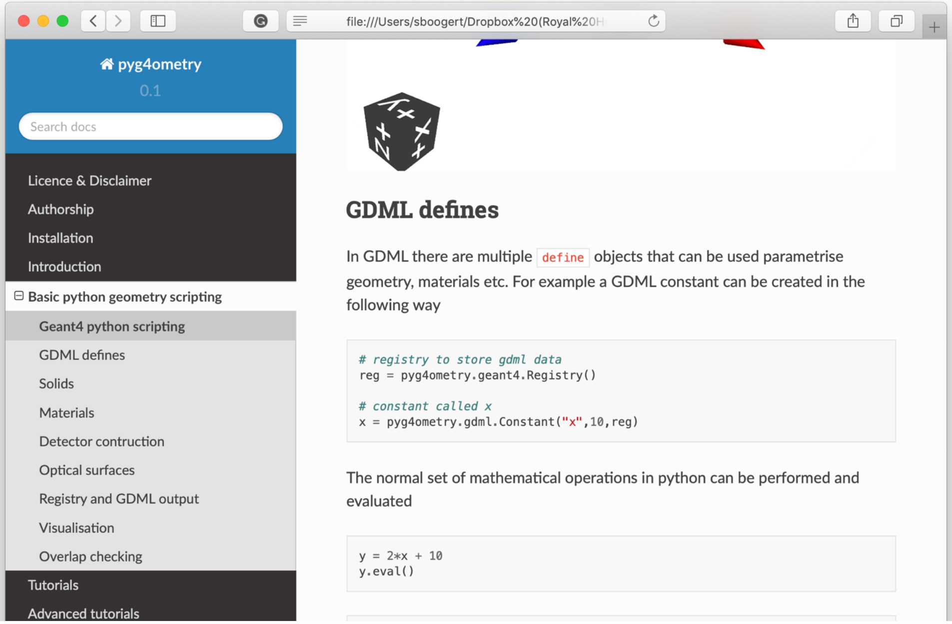
Testing

- Over 400 unit tests
- 86% test coverage

```
sics/coderepos/pyg4ometry/pyg4ometry/geant4/SkinSurface.py                  11     1    91%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/__init__.py                     12     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Box.py                    27     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Cons.py                   84     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/CutTubs.py               116    10    91%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Ellipsoid.py             109     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/EllipticalCone.py         97     4    96%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/EllipticalTube.py         60     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/ExtrudedSolid.py          63     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericPolycone.py        44     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericPolyhedra.py       47     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericTrap.py            82     1    99%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Hype.py                  132     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Intersection.py           42     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Layer.py                  39    10    74%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/MultiUnion.py             37     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/OpticalSurface.py         19     2    89%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Orb.py                    74     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Para.py                   46     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Paraboloid.py            106     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Plane.py                  26     4    85%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Polycone.py              149     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Polyhedra.py              46     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Scaled.py                 28     1    96%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/SolidBase.py              38     5    87%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Sphere.py                 79     7    91%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Subtraction.py            44     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TessellatedSolid.py       61     2    97%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Tet.py                    43     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Torus.py                 129     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Trap.py                  107     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Trd.py                    33     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Tubs.py                  103     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedBox.py             66     1    98%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedSolid.py           55     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTrap.py            98     1    99%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTrd.py             76     1    99%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTubs.py           134    15    89%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwoVector.py              50     8    84%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Union.py                  42     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Wedge.py                  37     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/__init__.py               40     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/pycsg/__init__.py                       1     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/stl/Reader.py                          45     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/stl/__init__.py                         1     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/transformation.py                     125     4    97%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Convert.py               26     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Mesh.py                  59     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/VisualisationOptions.py   7     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/VtkViewer.py            345    76    78%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Writer.py                16     0   100%
sics/coderepos/pyg4ometry/pyg4ometry/visualisation/__init__.py               4     0   100%
-----------------------------------------------------------------------------------------------
                                                                         10480  1428    86%
```

Manual

- Unit tests **are** an excellent documentation for features
- Sphinx documentation
- All examples in this presentation (apart from the complex Fluka and CAD) are in Git repository



Potential projects and future directions

ROYAL HOLLOWAY UNIVERSITY OF LONDON

- **Possible pyg4ometry developments**
- Need to speed up VTK (too many vtkActors)
- C++ output for compiled code
- Add "loop" GDML tags
- GUI for controlling geometry creation (FreeCAD interface)
- Output mode for Unity/Unreal/gltf/USD(Z) engines for outreach activities
- Build model hierarchy (LV,PV) from Fluka input
- Dynamic scene tree update
- Performance testing (physics comparisons) between Geant4 and Fluka
- Symbolic expression simplification (GDML equations, Fluka regions?)
- Simultaneous STEP geometry creation (cadquery)
- More export targets (gltf, vtk, usdz)

- **Geant4 developments**
- ~~Create VTK based visualizer for Geant4 (might help with solids which cannot be displayed in OpenGL visualization)~~
- Boolean processing and VTK already in geant4 now
- Contribute to Fluka/Geant4 interoperability projects
- Waiting for MOIRA as potential to load STEP directly into G4
- Update GDML to use different file formats (STL, CAD)
- Multiple GDML files per Geant4 application (currently only one world volume supported and lots of name collisions)

- **Flair developments**
- ~~GDML loader for Flair?~~
- Discussed with Vasilis and too many dependencies ;-(

Imagined workflows

ROYAL HOLLOWAY UNIVERSITY OF LONDON

- Simultaneous creation of geometry for multiple target simulations (G4, Fluka, MCMP, PHITS etc
- Export of meshes for Multiphysics and other applications (need gmsh, tetgen, CGAL, etc)
- Load of STEP, triangulate and compare to G4 (cross checking)
- Front end UI to create G4 simulation geometry (think Flair/FreeCAD)
- New generation of beamline builders for FCC-ee, FCC-hh, PBC etc
- Test new visualization algorithms and systems without need to recompile
- Develop detector geometry from start using tools like pyg4ometry (integrating activity of engineers, students, software experts together)
- Workflows using triangulated or quad meshes (DAGMC)

Conclusions

- Have developed a relatively powerful geometry manipulation tool(kit)
- Uses most up to date software packages and modern programming language
- Generic conversion back and forth between different tools is quite possible but probably not how a tool like this will be used (area of discussion)
- Need testing and refinement on larger and more complex models to home the algorithms and tests
- Potential to save a lot of user's time with generating geometry
- Programmatic interface to geometry creation and manipulation, lots of potential for different use cases we have not thought about
- Materials between different codes still needs some work
- Need to test more models, run timing tests in Geant4/Fluka

Backup discussion points

- **Global name space in GDML file**
- Merging GDML files needs to avoid collisions between names of nodes
- BDSIM for example preprocesses GDML to update all names (slow)
- Tessellated solid data could potentially be in a separate file
- Use python object ID opposed to name to key objects (similar to pointer usage in G4)