

Geometry preparation for Geant4

51st Geant4 Technical Forum, CERN. 16th Jan 2020

Stewart Boogert

Andrey Abramov, Laurie Nevay, William Shields Benjamin Shellswell,
Stuart Walker



ROYAL
HOLLOWAY
UNIVERSITY
OF LONDON

<https://bitbucket.org/jairhul/pyg4ometry/src/develop/>

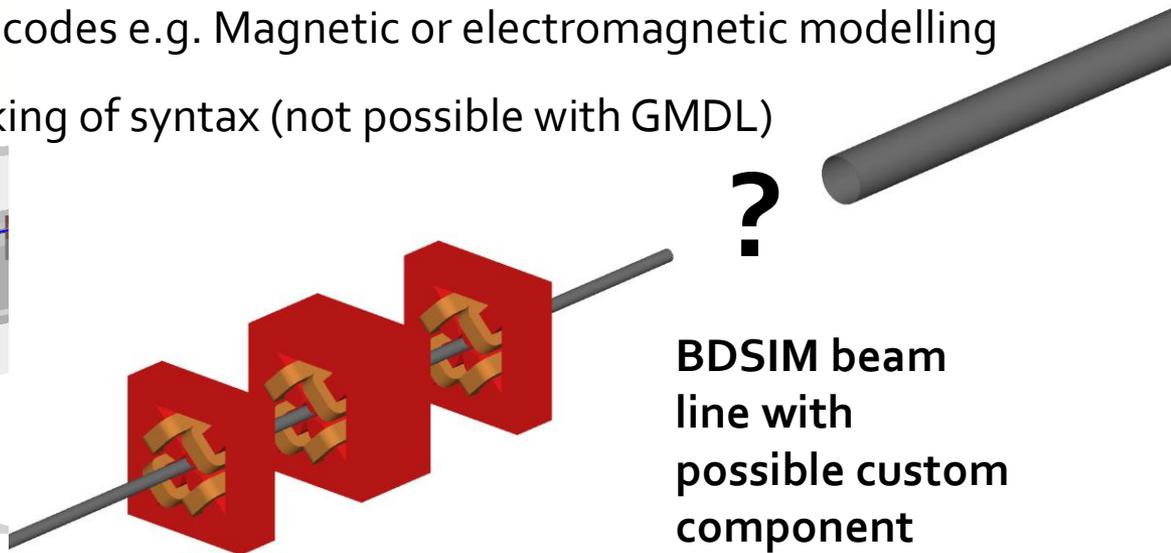
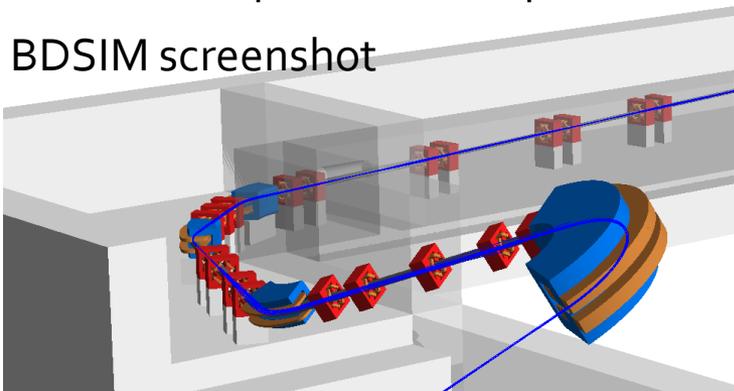
<http://www.pp.rhul.ac.uk/bdsim/pyg4ometry/>

Beam delivery simulation (BDSIM)



- RHUL group has developed BDSIM, a code to make Geant4 accelerator models
 - <https://arxiv.org/abs/1808.10745> <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

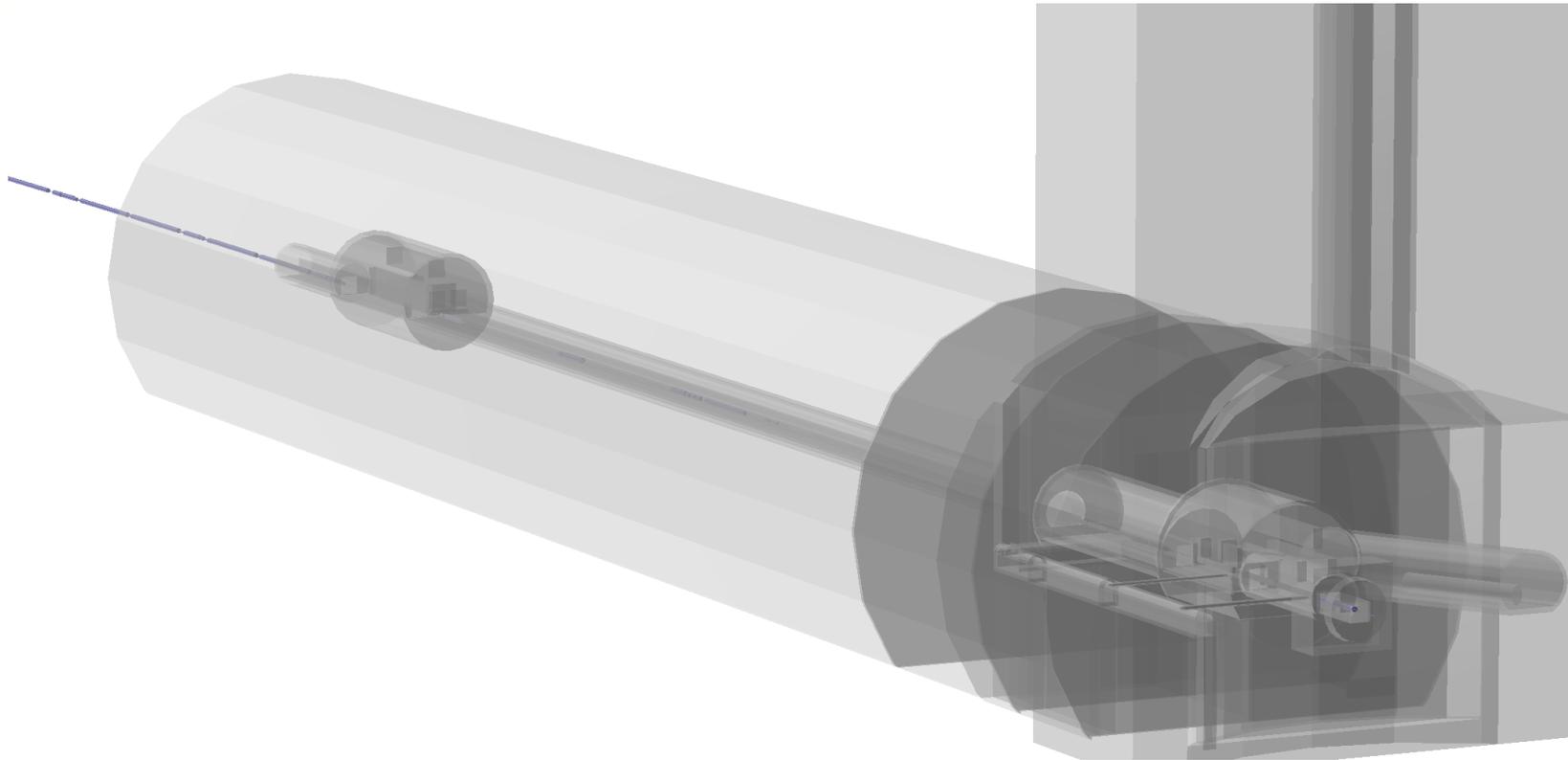


**BDSIM beam
line with
possible custom
component**

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

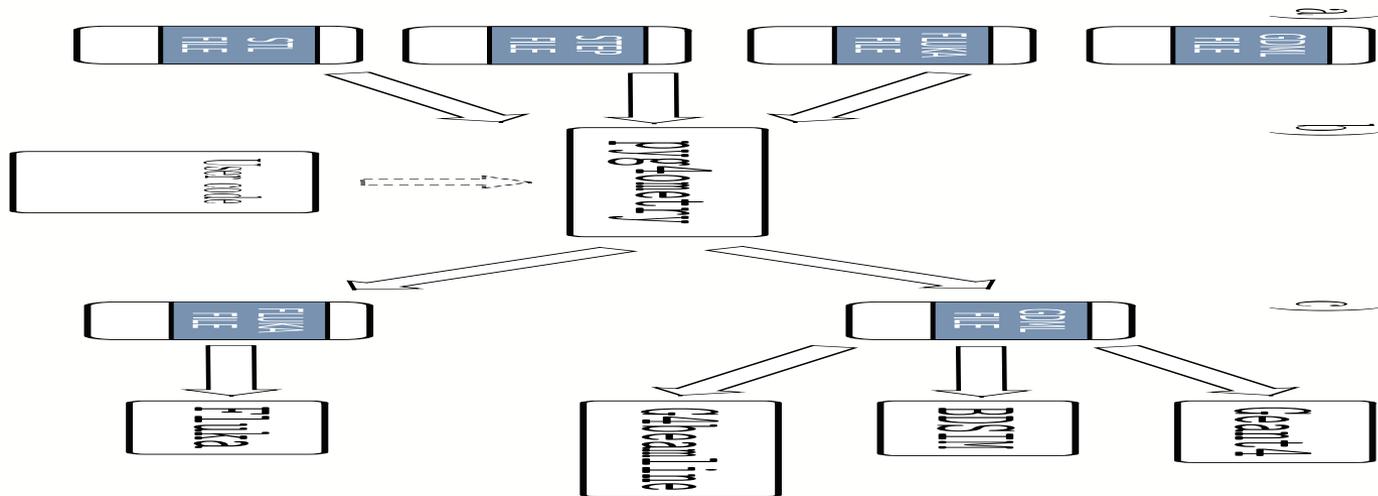


- Load (and convert) GDML, STL, STEP, FLUKA files
- Complete support (reading/writing) of GDML
- Visualize geometry
- Check for overlaps and geometry issues
- Composite (load and place) geometry from different sources
- Rendering for data analysis
- Modify geometry (cut holes, remove material etc.)
- 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 ;-)

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

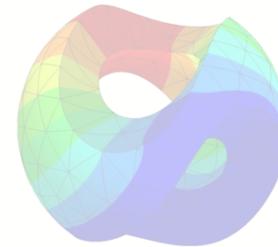
ANTLR



Visualisation
Tool kit



FreeCAD



NGSolve



FEniCS



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, "w1", reg)
    bl = _g4.LogicalVolume(bs, bm, "b1", 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)

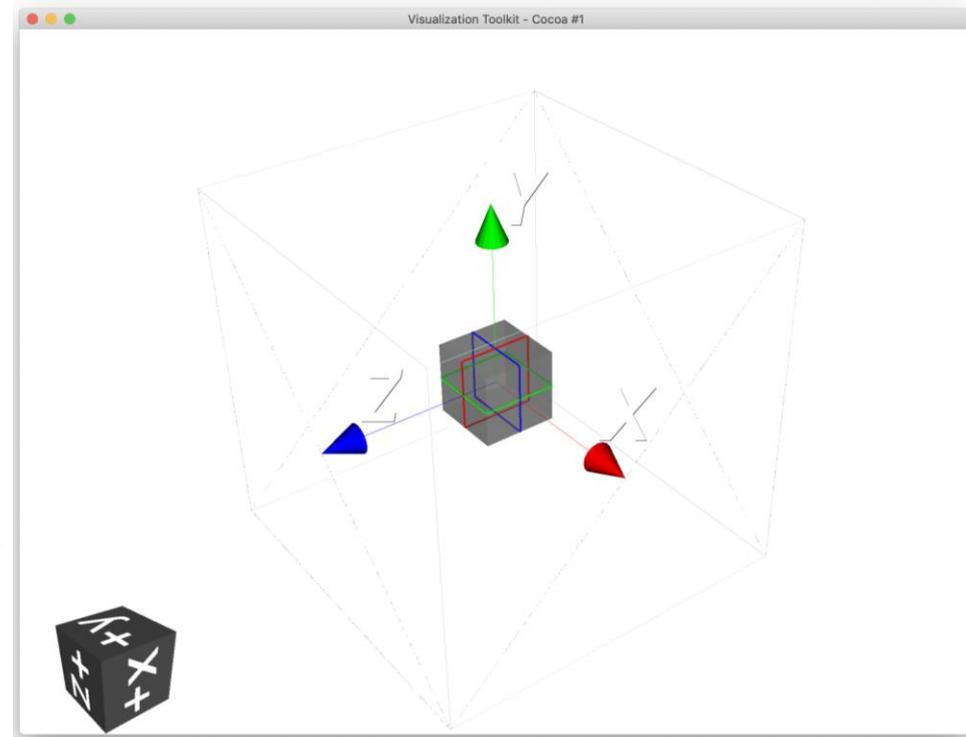
    return {"testStatus": True, "logicalVolume":wl, "vtkViewer":v}
```

Defines

Materials

Solids

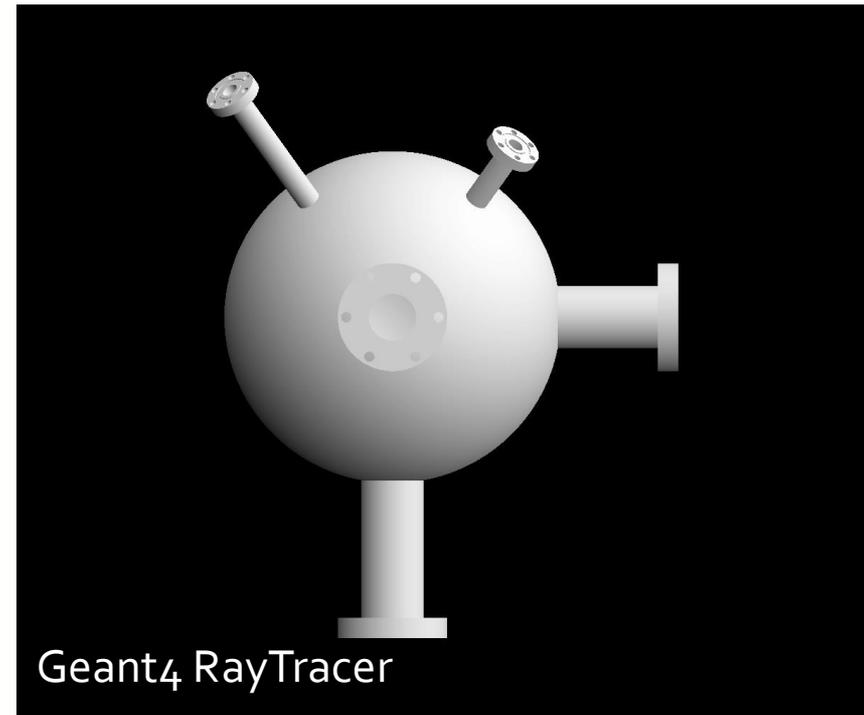
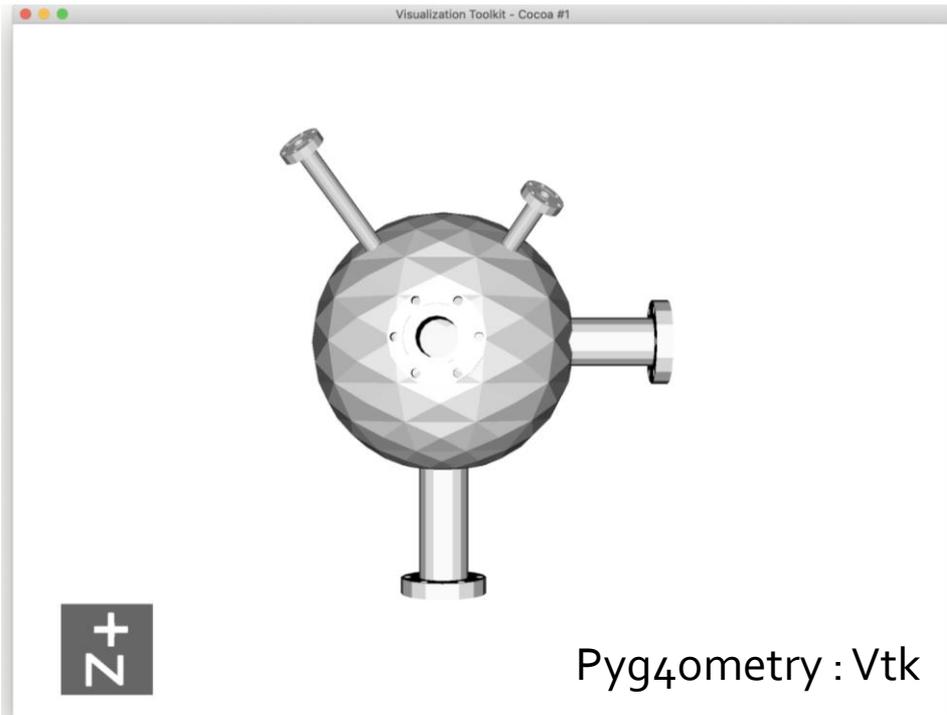
Structure



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")

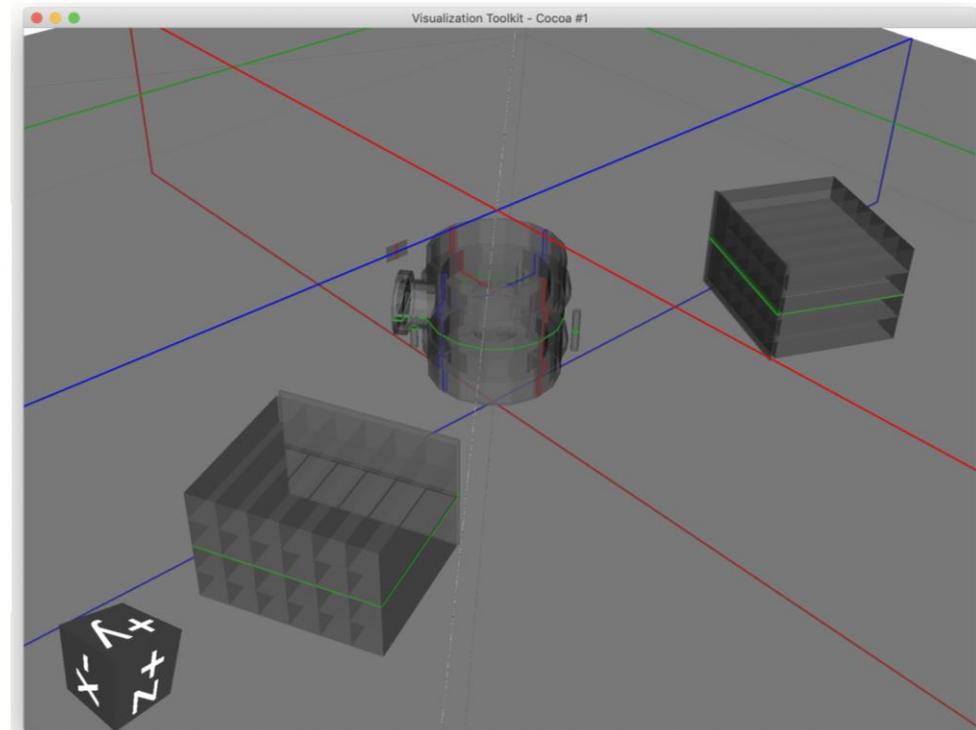
[In [3]: l = r.getRegistry().getWorldVolume()

[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 if 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()

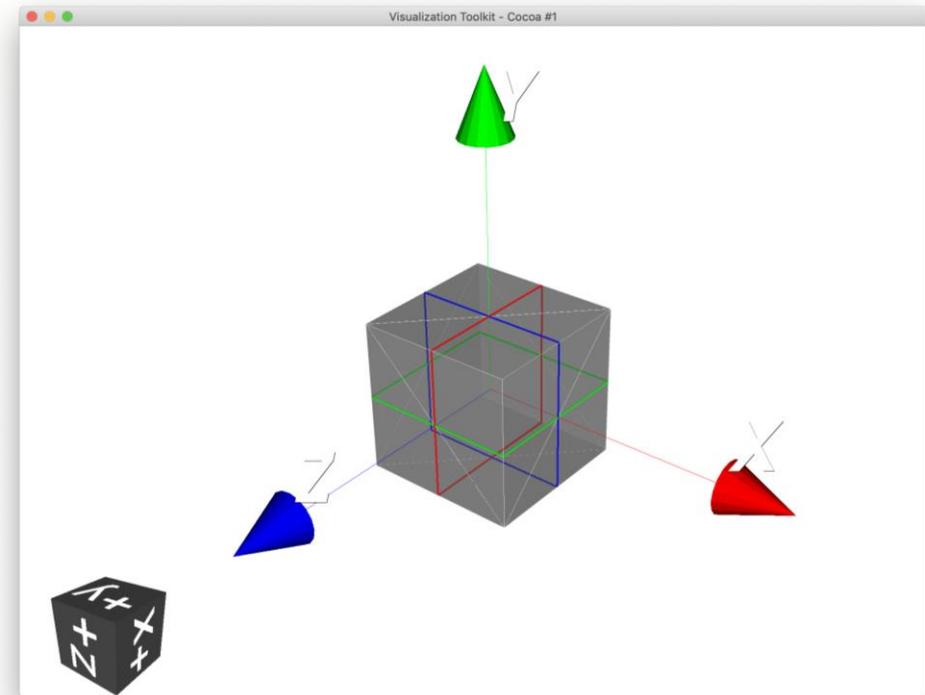
    rpp = RPP("RPP_BODY", 0, 10, 0, 10, 0, 10, flukaregistry=freg)
    z = Zone()
    z.addIntersection(rpp)
    region = Region("RPP_REG", material="COPPER")
    region.addZone(z)
    freg.addRegion(region)

    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)

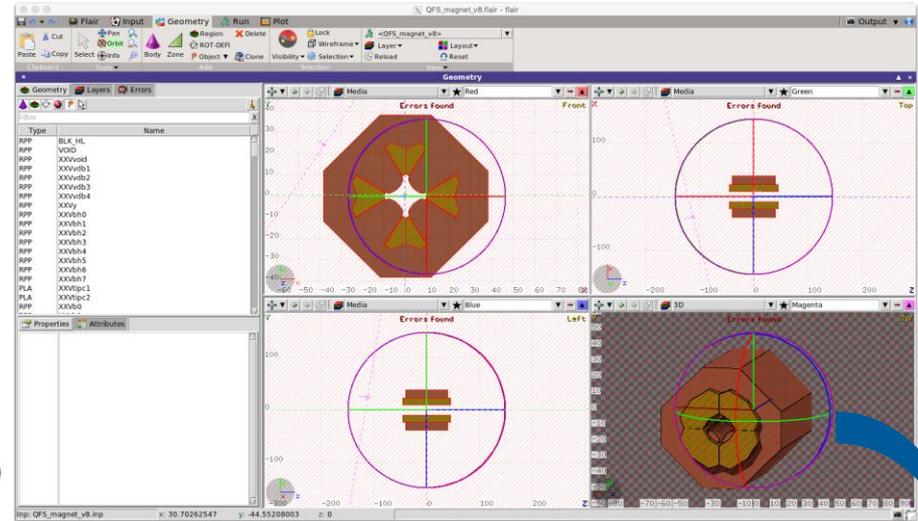
    return {"testStatus": True, "logicalVolume": greg.getWorldVolume(), "vtkViewer":v}
```



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()

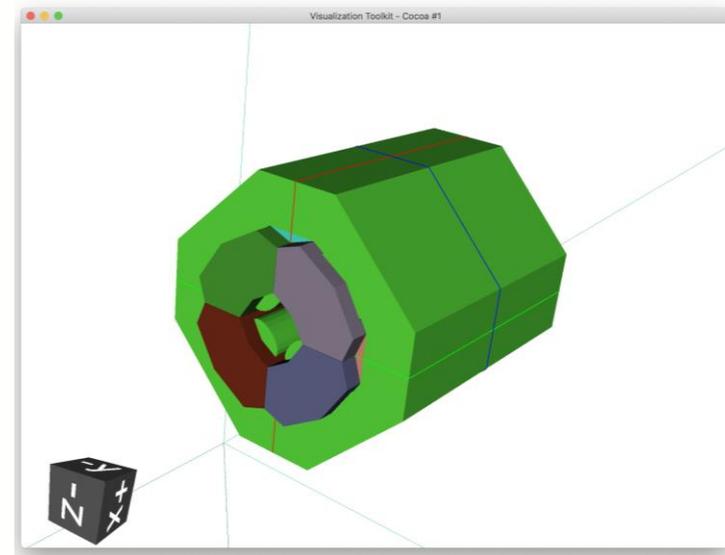
[In [5]: v = pyg4ometry.visualisation.VtkViewer()

[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")
```

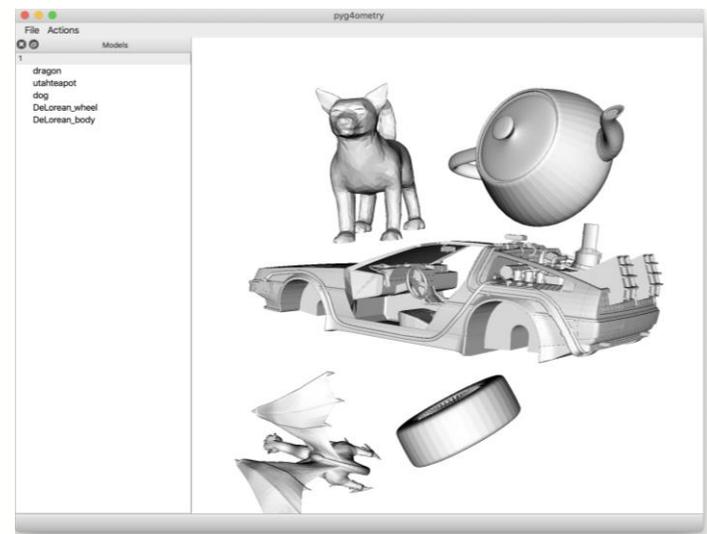
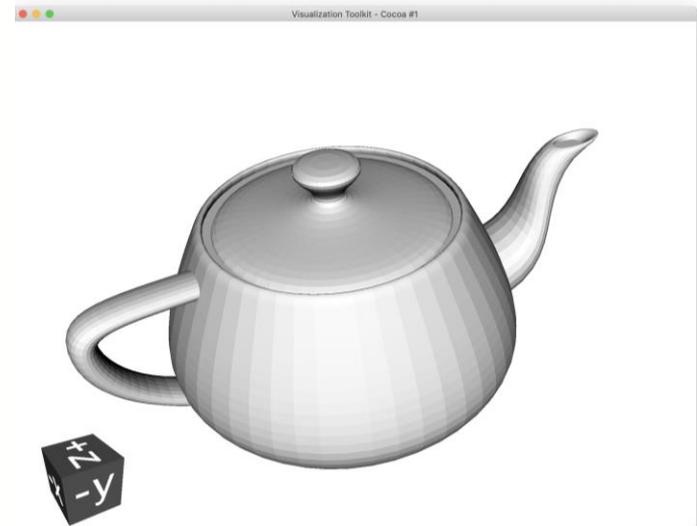
```
[In [4]: l = r.logicalVolume("test", "G4_Cu", reg)
```

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

```
[In [6]: v.addLogicalVolume(l)
```

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

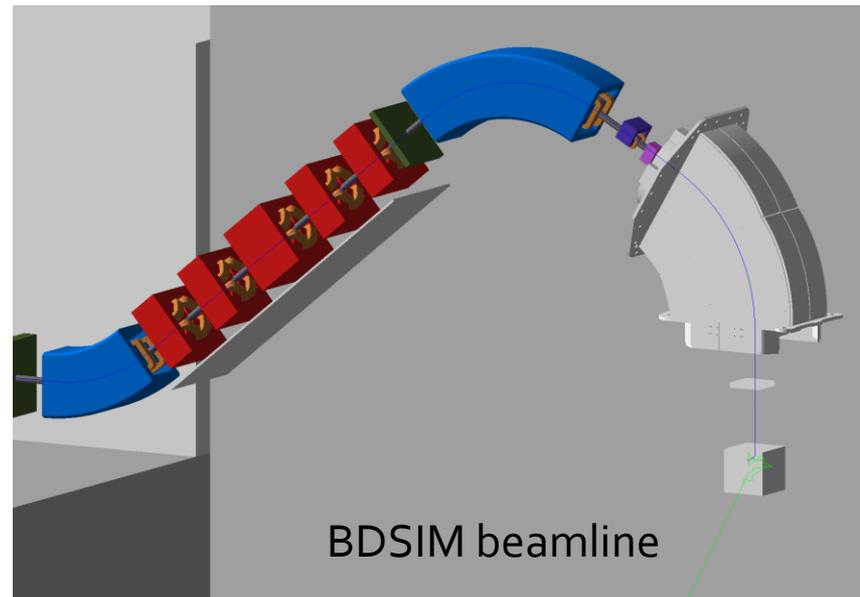
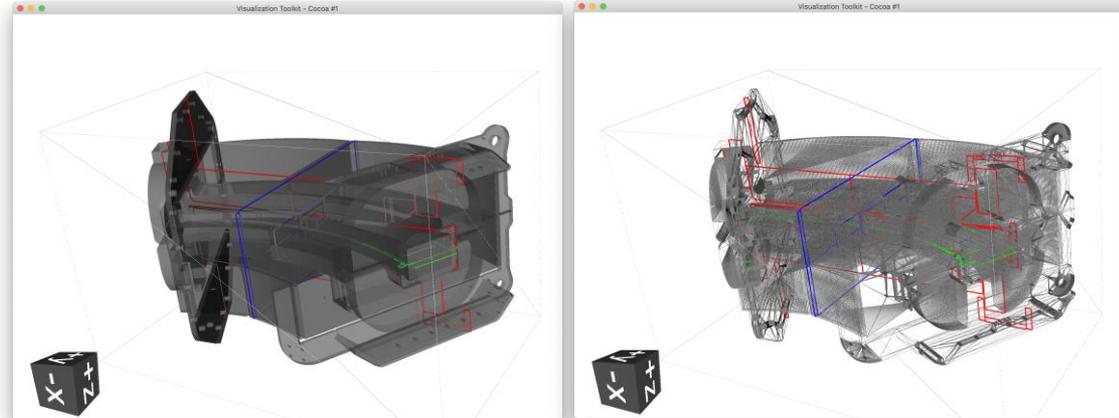
- Incorporate MeshCAD or other tools?



STP/STEP to Geant4-GDML example



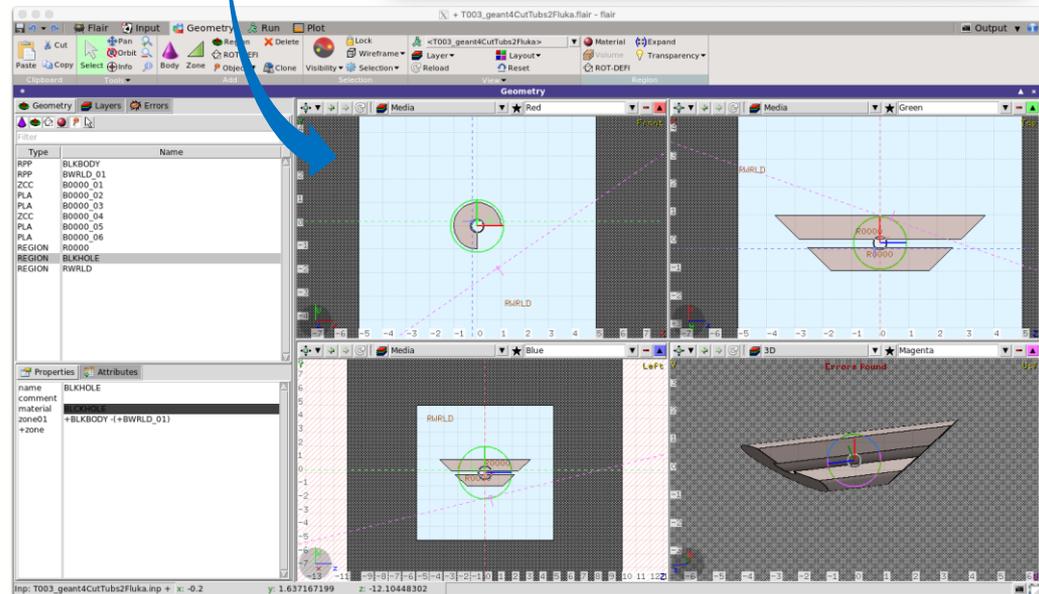
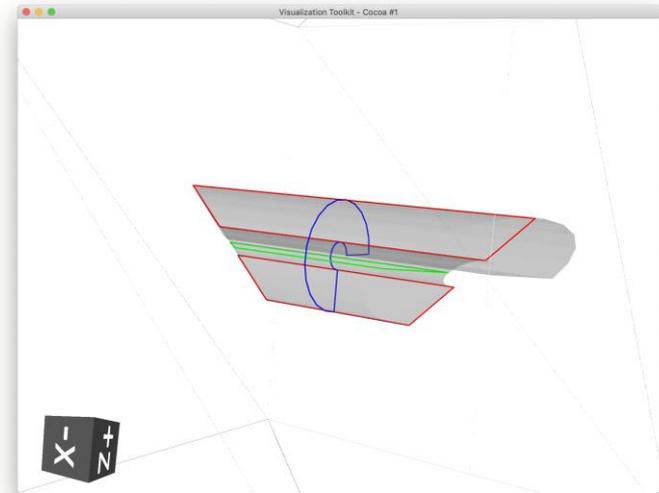
- Load STEP file using FreeCAD-OpenCascade
- Still need to simplify CAD file
- Bodies and Parts map well to LV and PVs respectively. Convert bodies to triangulated mesh and place
- Based on STL loading
- Very advanced proof of principle (working in our workflow)
- Need to account for material
- (what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



Geant4-GDML to Fluka example



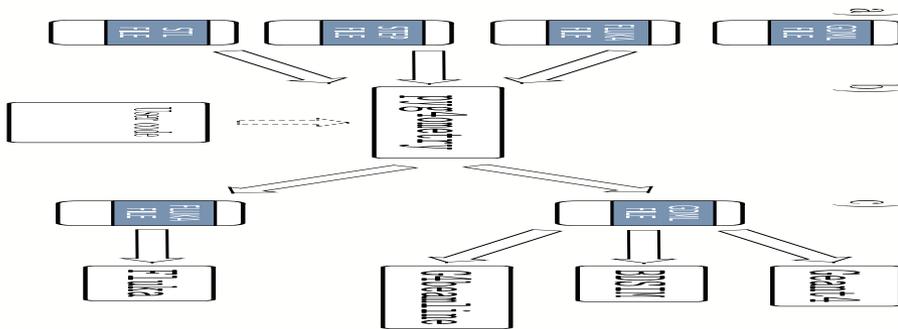
- Decompose G₄-GDML primitive solids to Fluka bodies, then zones and join into regions
- Cut daughter volumes from mother to create flat hierarchy
- Working through G₄ solids
- Need to implement union, intersection and subtraction G₄ solids
- Least developed area of pyg₄ometry but making rapid progress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



Compositing example



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

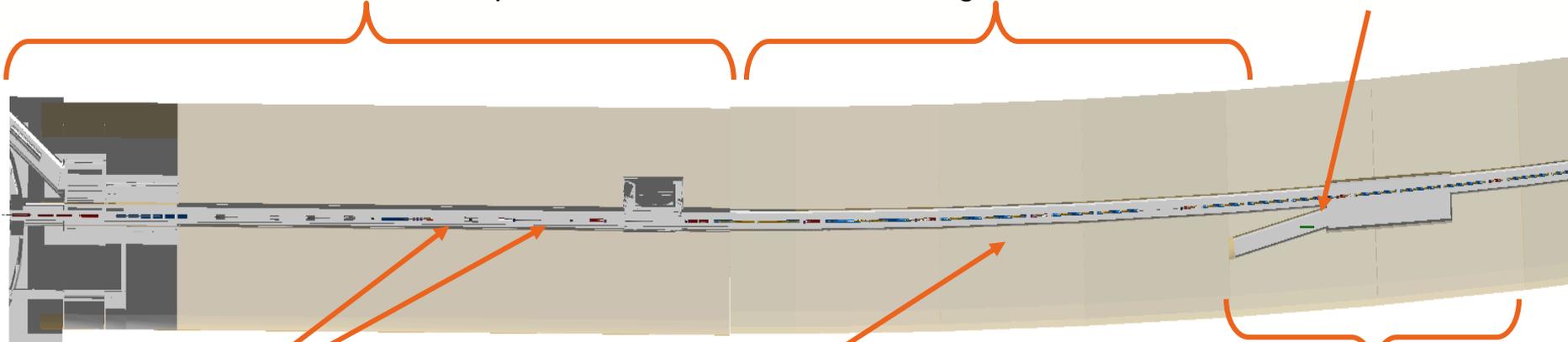


- Convert LogicalVolumes to Assembly volumes (excellent for placement)
- Intelligently merge two GDML files (dealing with name clashes)
- Example where Fluka/Geant4/GDML is used

converted tunnel complex

BDSIM generic tunnel

FASER



shielding blocks

BDSIM beam line placed inside all tunnel pieces

custom tunnel prepared using pyg4ometry

Overlap detection example



Three classes in general

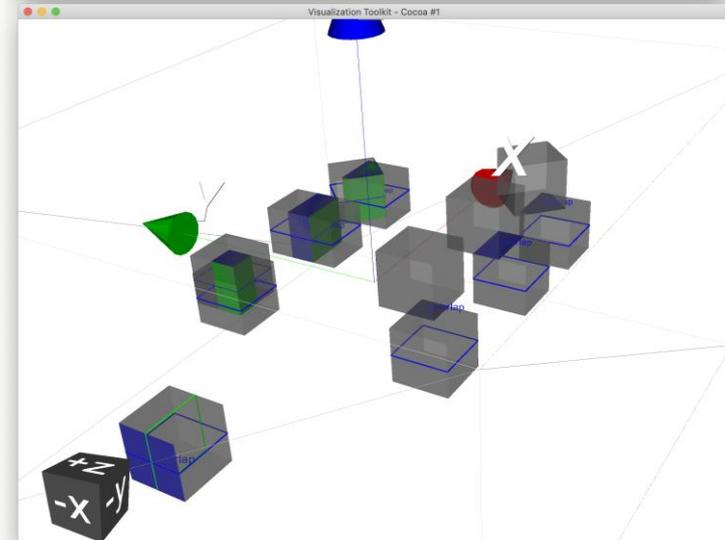
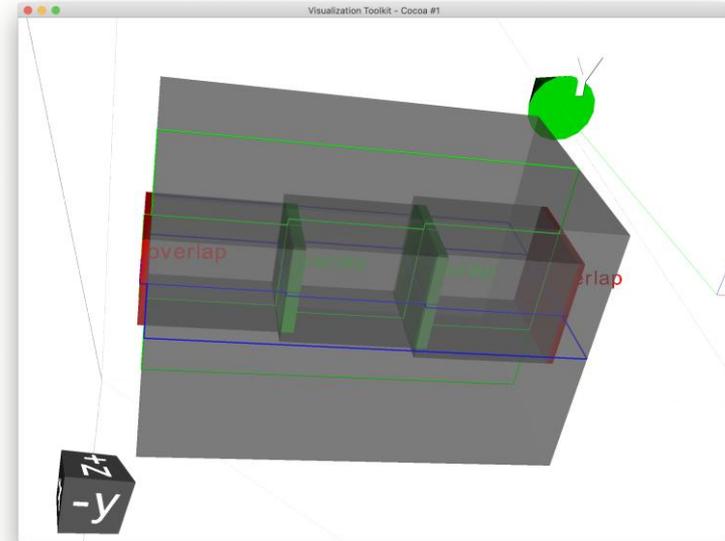
- **Protrusion**
- **Daughter overlap**
- **Coplanar faces (can be a problem)**

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





Testing

- Over 400 unit tests
- 86% test coverage

| | | | | |
|----|--|-------|------|------|
| 43 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/SkinSurface.py | 11 | 1 | 91% |
| 44 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/_init__.py | 12 | 0 | 100% |
| 45 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Box.py | 27 | 0 | 100% |
| 46 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Cons.py | 84 | 0 | 100% |
| 47 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/CutTubs.py | 116 | 10 | 91% |
| 48 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Ellipsoid.py | 109 | 0 | 100% |
| 49 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/EllipticalCone.py | 97 | 4 | 96% |
| 50 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/EllipticalTube.py | 68 | 0 | 100% |
| 51 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/ExtrudedSolid.py | 63 | 0 | 100% |
| 52 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericPolycone.py | 44 | 0 | 100% |
| 53 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericPolyhedra.py | 47 | 0 | 100% |
| 54 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/GenericTrap.py | 82 | 1 | 99% |
| 55 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Hype.py | 132 | 0 | 100% |
| 56 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Intersection.py | 42 | 0 | 100% |
| 57 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Layer.py | 39 | 10 | 74% |
| 58 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/MultiUnion.py | 37 | 0 | 100% |
| 59 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/OpticalSurface.py | 19 | 2 | 89% |
| 60 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Orb.py | 74 | 0 | 100% |
| 61 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Para.py | 46 | 0 | 100% |
| 62 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Paraboloid.py | 106 | 0 | 100% |
| 63 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Plane.py | 26 | 4 | 85% |
| 64 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Polycone.py | 149 | 0 | 100% |
| 65 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Polyhedra.py | 46 | 0 | 100% |
| 66 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Scaled.py | 28 | 1 | 96% |
| 67 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/SolidBase.py | 38 | 5 | 87% |
| 68 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Sphere.py | 79 | 7 | 91% |
| 69 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Subtraction.py | 44 | 0 | 100% |
| 70 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TessellatedSolid.py | 61 | 2 | 97% |
| 71 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Tet.py | 43 | 0 | 100% |
| 72 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Torus.py | 129 | 0 | 100% |
| 73 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Trap.py | 107 | 0 | 100% |
| 74 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Trd.py | 33 | 0 | 100% |
| 75 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Tubs.py | 103 | 0 | 100% |
| 76 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedBox.py | 66 | 1 | 98% |
| 77 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedSolid.py | 55 | 0 | 100% |
| 78 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTrap.py | 98 | 1 | 99% |
| 79 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTrd.py | 76 | 1 | 99% |
| 80 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwistedTubs.py | 134 | 15 | 89% |
| 81 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/TwoVector.py | 50 | 8 | 84% |
| 82 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Union.py | 42 | 0 | 100% |
| 83 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/Wedge.py | 37 | 0 | 100% |
| 84 | sics/coderepos/pyg4ometry/pyg4ometry/geant4/solid/_init__.py | 40 | 0 | 100% |
| 85 | sics/coderepos/pyg4ometry/pyg4ometry/pycsg/_init__.py | 1 | 0 | 100% |
| 86 | sics/coderepos/pyg4ometry/pyg4ometry/stl/Reader.py | 45 | 0 | 100% |
| 87 | sics/coderepos/pyg4ometry/pyg4ometry/stl/_init__.py | 1 | 0 | 100% |
| 88 | sics/coderepos/pyg4ometry/pyg4ometry/transformations.py | 125 | 4 | 97% |
| 89 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Convert.py | 26 | 0 | 100% |
| 90 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Mesh.py | 59 | 0 | 100% |
| 91 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/VisualisationOptions.py | 7 | 0 | 100% |
| 92 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/VtkViewer.py | 345 | 76 | 78% |
| 93 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/Writer.py | 16 | 0 | 100% |
| 94 | sics/coderepos/pyg4ometry/pyg4ometry/visualisation/_init__.py | 4 | 0 | 100% |
| 95 | | | | |
| 96 | | 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

The screenshot shows the pyg4ometry website interface. The left sidebar contains a navigation menu with items like 'Licence & Disclaimer', 'Authorship', 'Installation', 'Introduction', 'Basic python geometry scripting', 'Geant4 python scripting', 'GDML defines', 'Solids', 'Materials', 'Detector construction', 'Optical surfaces', 'Registry and GDML output', 'Visualisation', 'Overlap checking', 'Tutorials', and 'Advanced tutorials'. The main content area is titled 'GDML defines' and explains that in GDML there are multiple 'define' objects that can be used to parametrise geometry, materials, etc. It provides an example of a GDML constant definition in Python: `# registry to store gdm data reg = pyg4ometry.geant4.Registry() # constant called x x = pyg4ometry.gdm.Constant("x", 10, reg)`. Below this, it states that the normal set of mathematical operations in python can be performed and evaluated, with an example: `y = 2*x + 10 y.eval()`.

Potential projects and future directions



▪ **Pyg4ometry developments**

- C++ output for compiled code
- Add "loop" GDML tags
- GUI for controlling geometry creation
- Output mode for Unity/Unreal 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?)

▪ **Geant4 developments**

- Create VTK based visualizer for Geant4 (might help with solids which cannot be displayed in OpenGL visualization)
- Contribute to Fluka/Geant4 interoperability projects
- 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?