MANCHESTER 1824
The University of Manchester

Pyg4ometry: a python package to manipulate Monte Carlo geometry

Stewart Boogert (University of Manchester)
pyHEP 2023 tutorial

Andrey Abramov (CERN), Laurie Nevay ICERN), William Shields (Royal Holloway), Luigi Pertoldi (TUM), Stuart Walker (DESY)

Pyg4ometry – in a nutshell

- **Pyg4ometry is a python API for GDML with the ability to create 3D surface meshes.**
  - API matches closely to Geant4 C++ API for detector construction (lowers cognitive load on users)
- Primary reason ~5 years ago
  - To avoid users writing Geant4 or FLUKA input *by-hand*
- Significantly evolved from its original mission
- Amazing amount is possible with this simple API

MANCHESTER 1824
The University of Manchester

Introduction - Authors

Stewart Boogert
(University of Manchester)

- Director of the Cockcroft Institute of Accelerator Science
- Accelerator physicist (beam instrumentation, ILC, simulations)
- HEP PhD and post-doc (ZEUS@HERA)

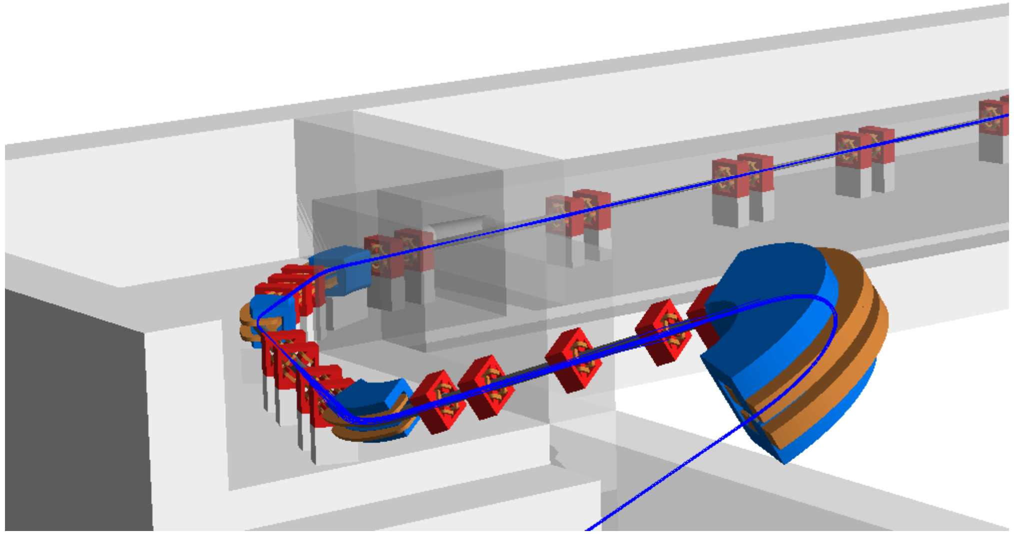
Laurie Nevay
(CERN)

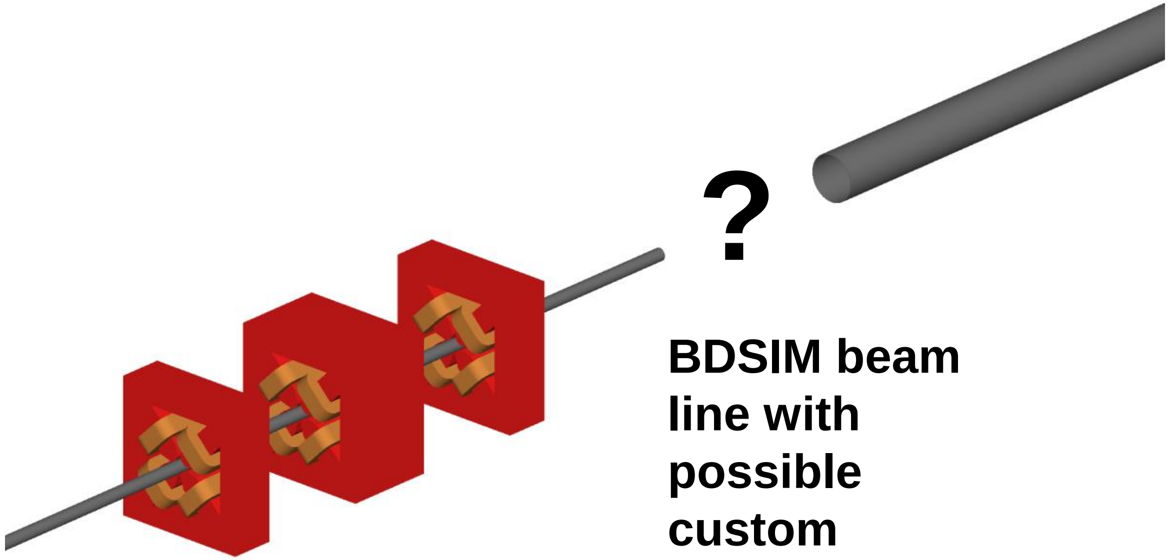
- CERN staff member
- Background in accelerator beam instrumentation, high power fibre lasers
- Lead developer of BDSIM - Geant4 application for accelerator models

Introduction - 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
- 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

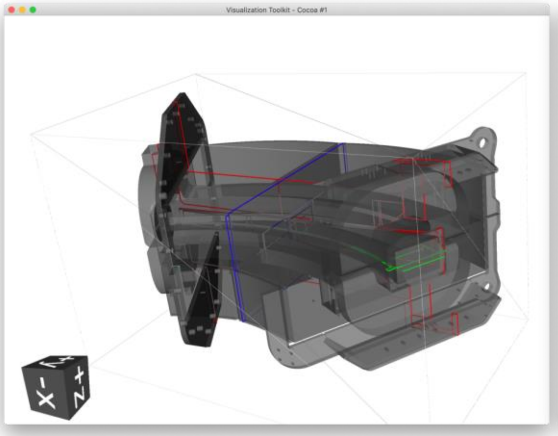


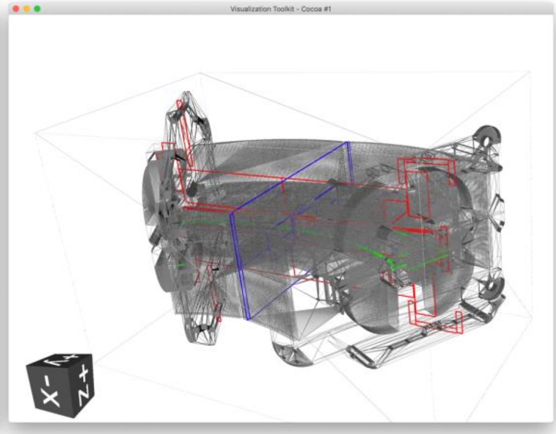


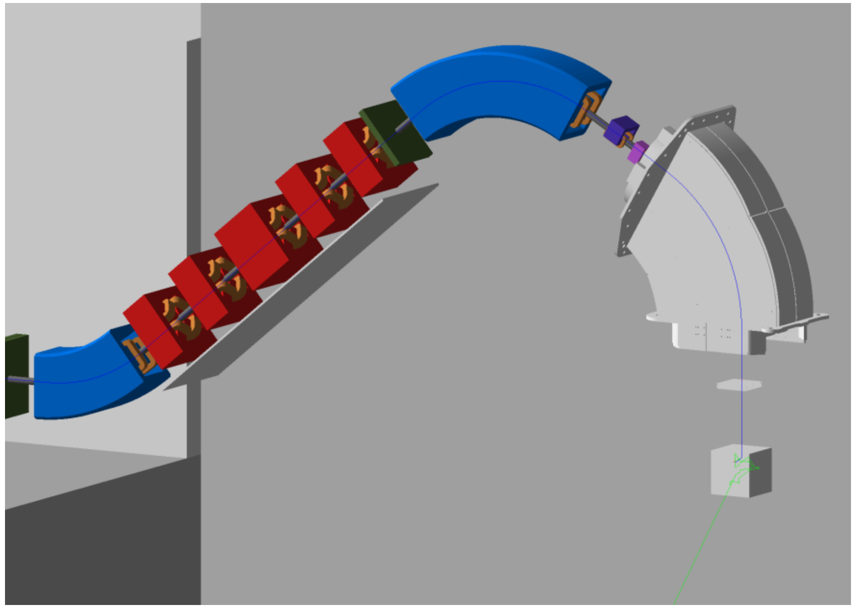
BDSIM beam line with possible custom component

Introduction BDSIM - 2

- Load STEP file using OpenCascade
- Still need to simplify CAD file
- **Parts** and **assemblies** map well to LV and PVs respectively. Convert bodies to triangulated mesh and place
- Need to account for material
  - Not used in consistent way in CAD







Introduction – pyg4ometry

Accelerator physics also user of tools like Geant4

- Others too MCNP, FLUKA, PHITS etc.

Other tools like DD4Hep or root TGeo do not work well in the context of accelerators.

- e.g. calibration and reconstruction not important
- Diverse users are not at same skill level as HEP community

The University of Manchester

Why?

Parametric geometry interface allows

- **algorithmic** generation of geometry
- Stable **complex** manipulation of geometry
- **conversion** of geometry between formats
- **sustainability** of pre-existing geometry information
- **feature extraction** from legacy geometry
- **new** applications (XR)

Potential users

- HEP detector physicists
  – E.g. those making small prototypes
- HEP simulation developers
  – E.g. those making interfaces with optical ray tracing codes)
- Radiological protection
- Space weather
- Medical physics
- Nuclear physicists

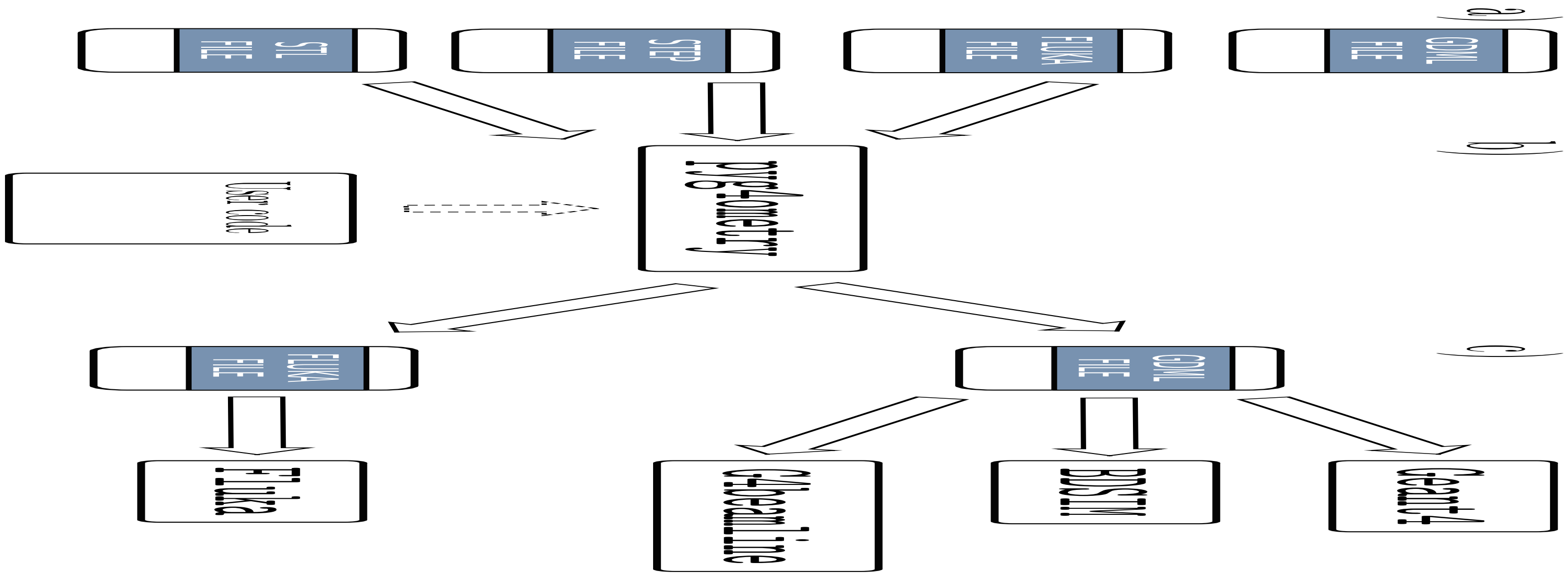
The University of Manchester

Requirements

- Load (and convert) GDML, STL, STEP, FLUKA, ROOT 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 intern student)
- Simple to contribute to (think of a PhD student)
- Reasonable performance

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

The University of Manchester

All dependencies are all open source and well maintained

Python

pybind11

ANTLR

Visualisation Toolkit

OpenCASCADE

Computational Geometry Algorithms Library

SymPy

Parsing GDML Mathematical expression

Viewing geomery

Loading CAD geometry

Algorithms e.g. booleans

The University of Manchester

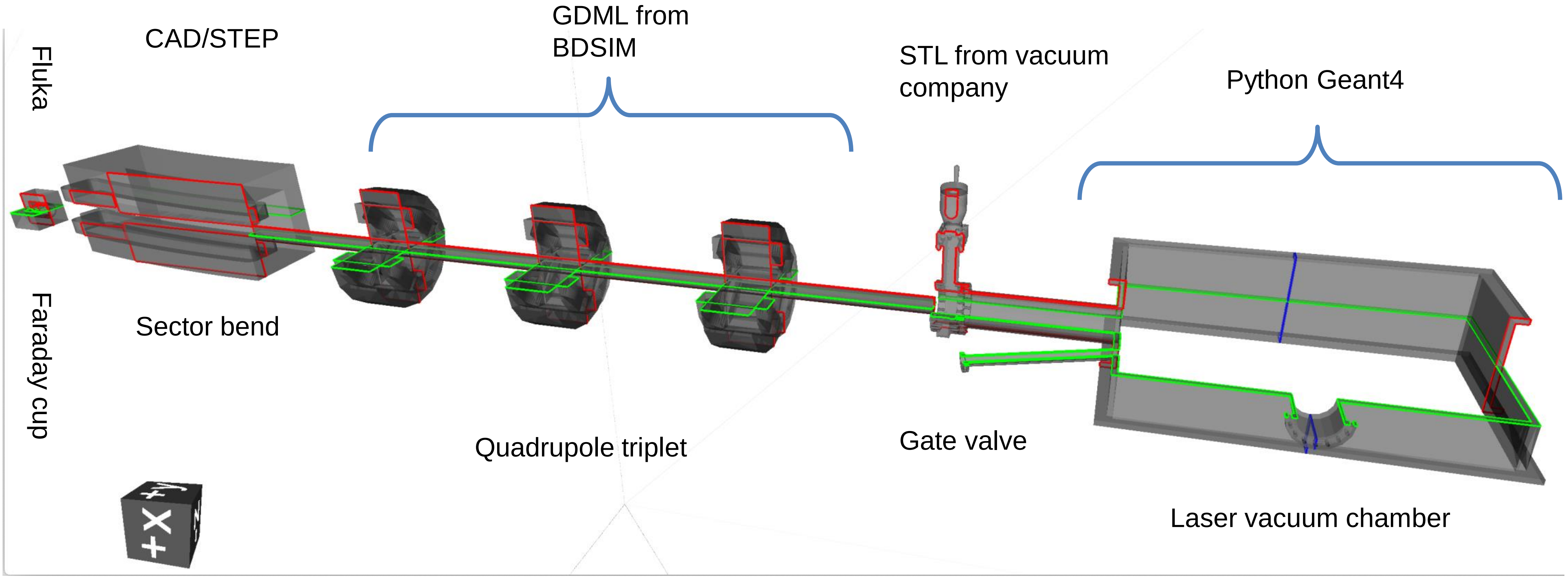
Legitimate questions

- **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

Juypter notebook

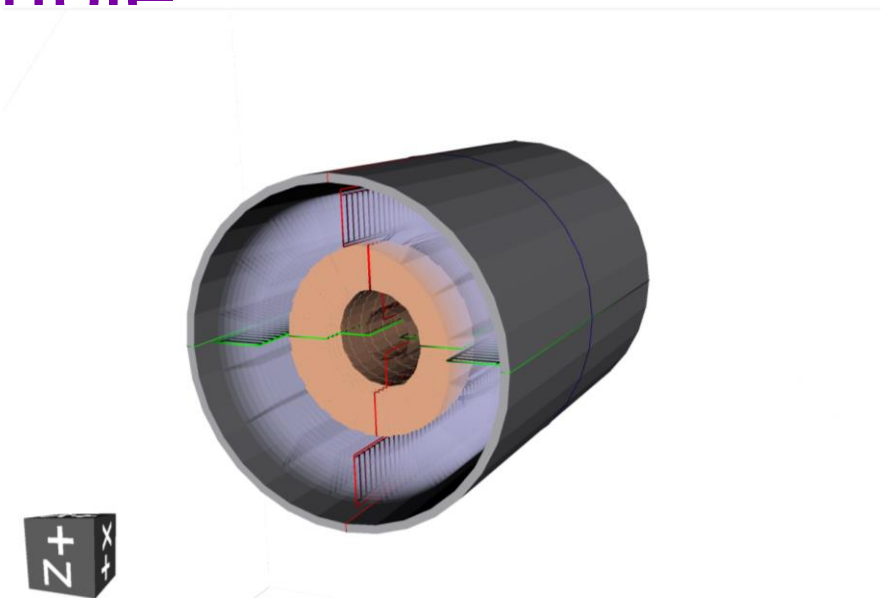
1. Creating simple geometry
2. Parametric geometry
3. Modifying geometry
4. Loading CAD and other formats
5. Compositing geometry
6. Converting geometry

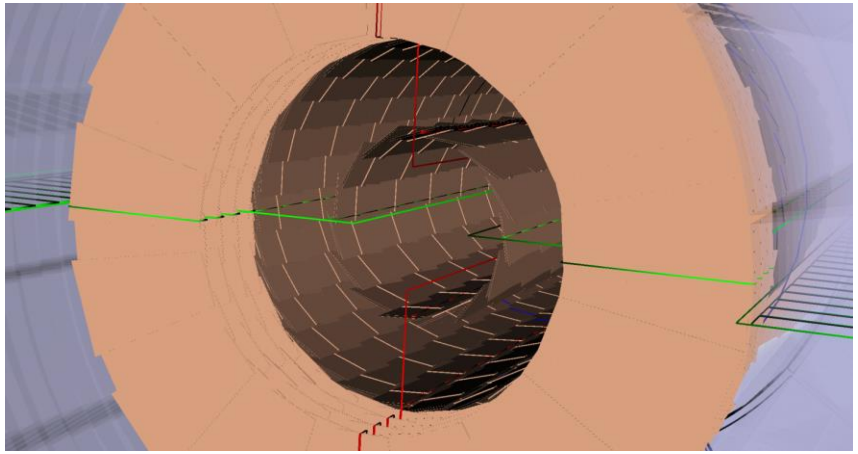
Compositing detailed example



Parametric design example

- 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

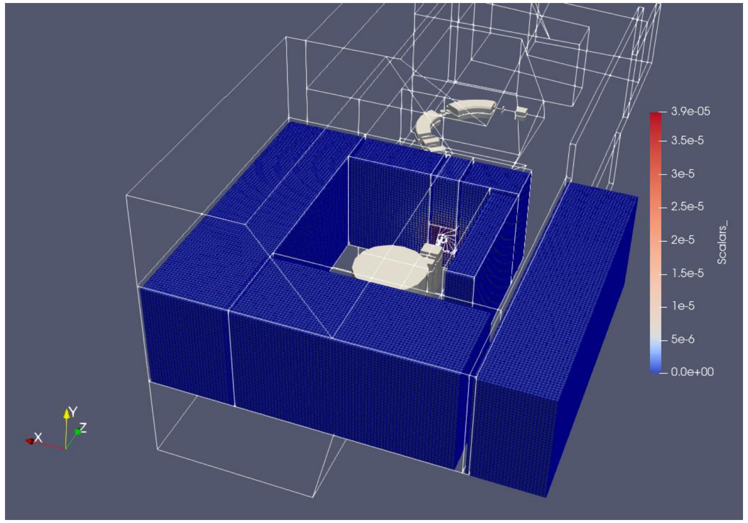


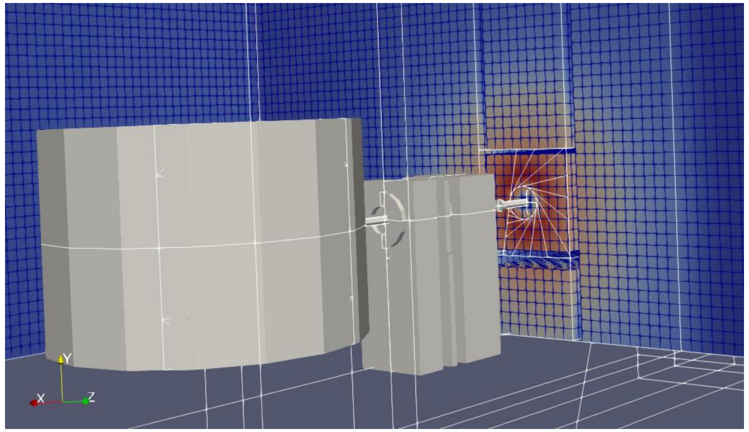


MANCHESTER 1824
The University of Manchester

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

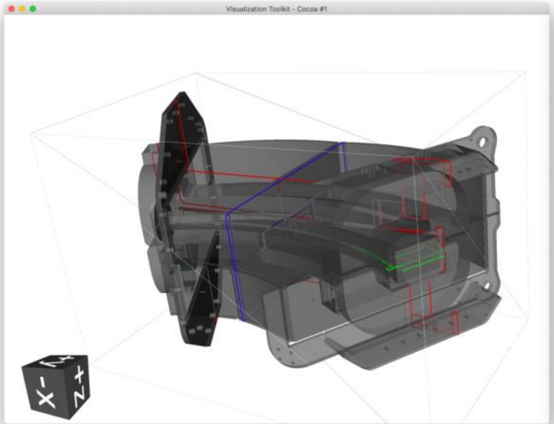


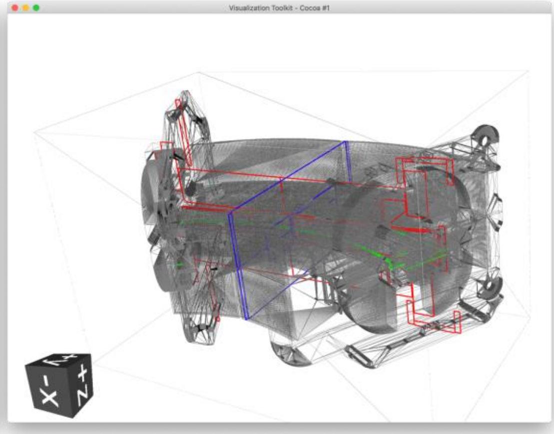


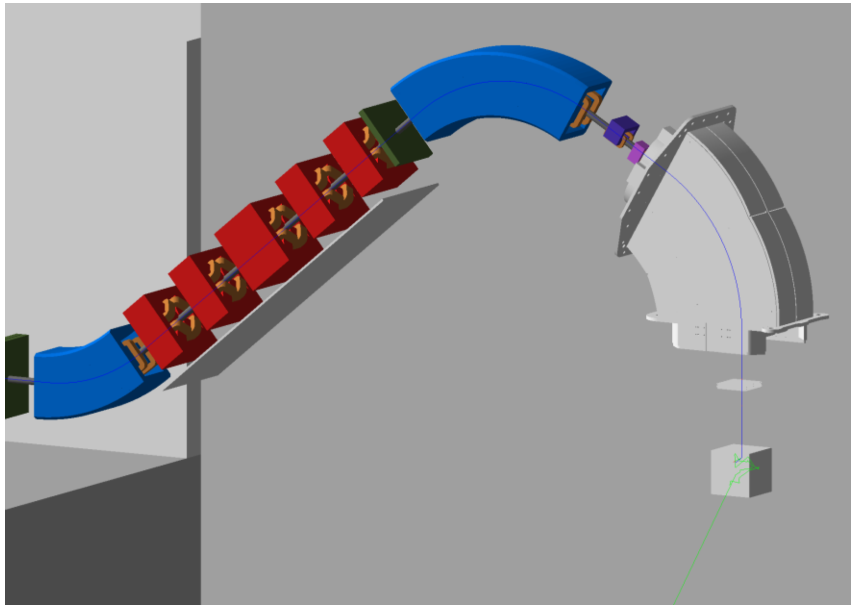
MANCHESTER
1824
The University of Manchester

CAD example

- Load STEP file using OpenCascade
- Still need to simplify CAD file
- **Parts** and **assemblies** map well to LV and PVs respectively. Convert bodies to triangulated mesh and place
- Need to account for material
  - Not used in consistent way in CAD



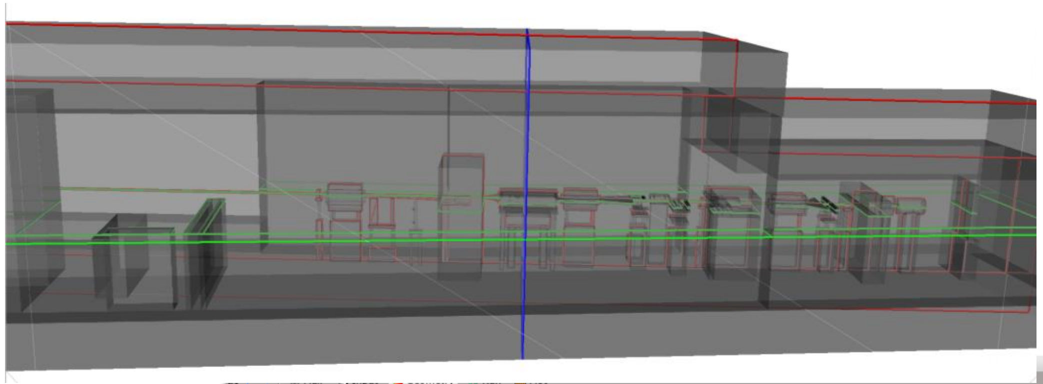


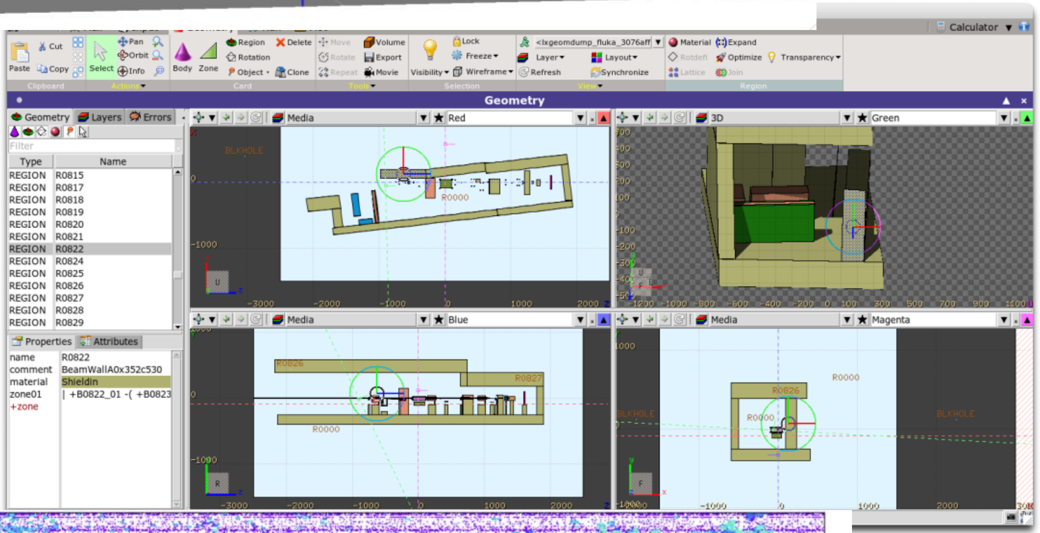


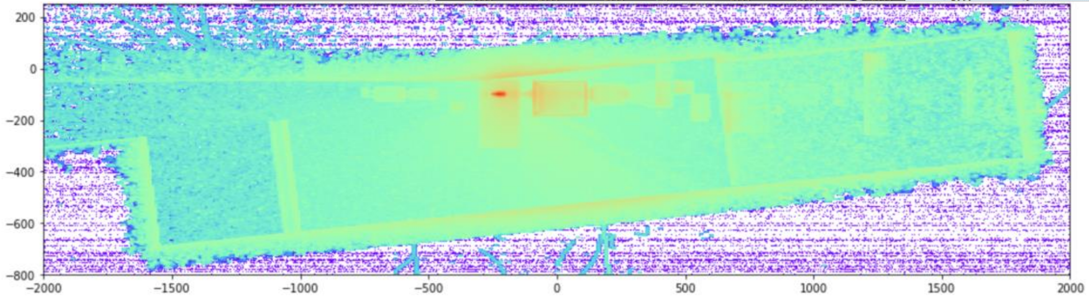
Full experiment FLUKA conversion (LUXE)

1. Experiment Geant4 instance
2. Export to GDML
3. Load GDML in pyg4ometry
4. Change/simplify etc
5. Export to FLUKA
6. Check geometry in FLAIR
7. Run FLUKA

Process takes minutes







MANCHESTER 1824
The University of Manchester

Jupyter notebook (things not shown)

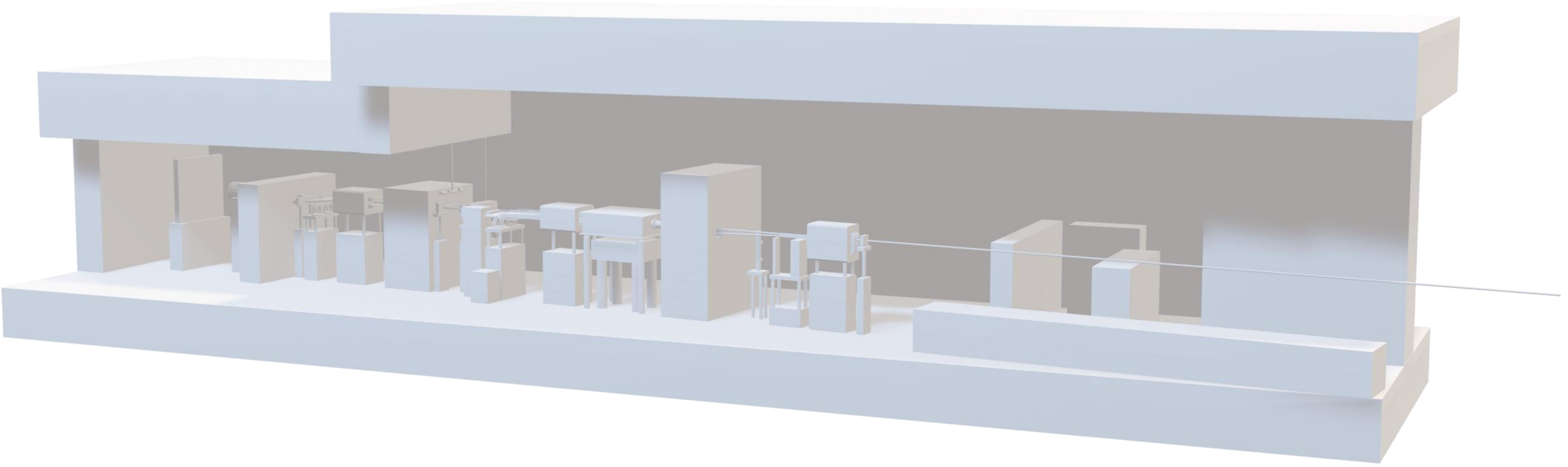
- Extracting features from geometry
- Cutting and clipping geometry
- Advanced visualization
- ROOT geometry loading
- Optical surfaces
- Pybind11 interfaces to OpenCASCADE and CGAL
  - Granular binding to functionality of libraries. So pyg4ometry offers many possibilities in python to create work-flows
- **Please look at tests if interested pyg4ometry/tests/**

Cool applications

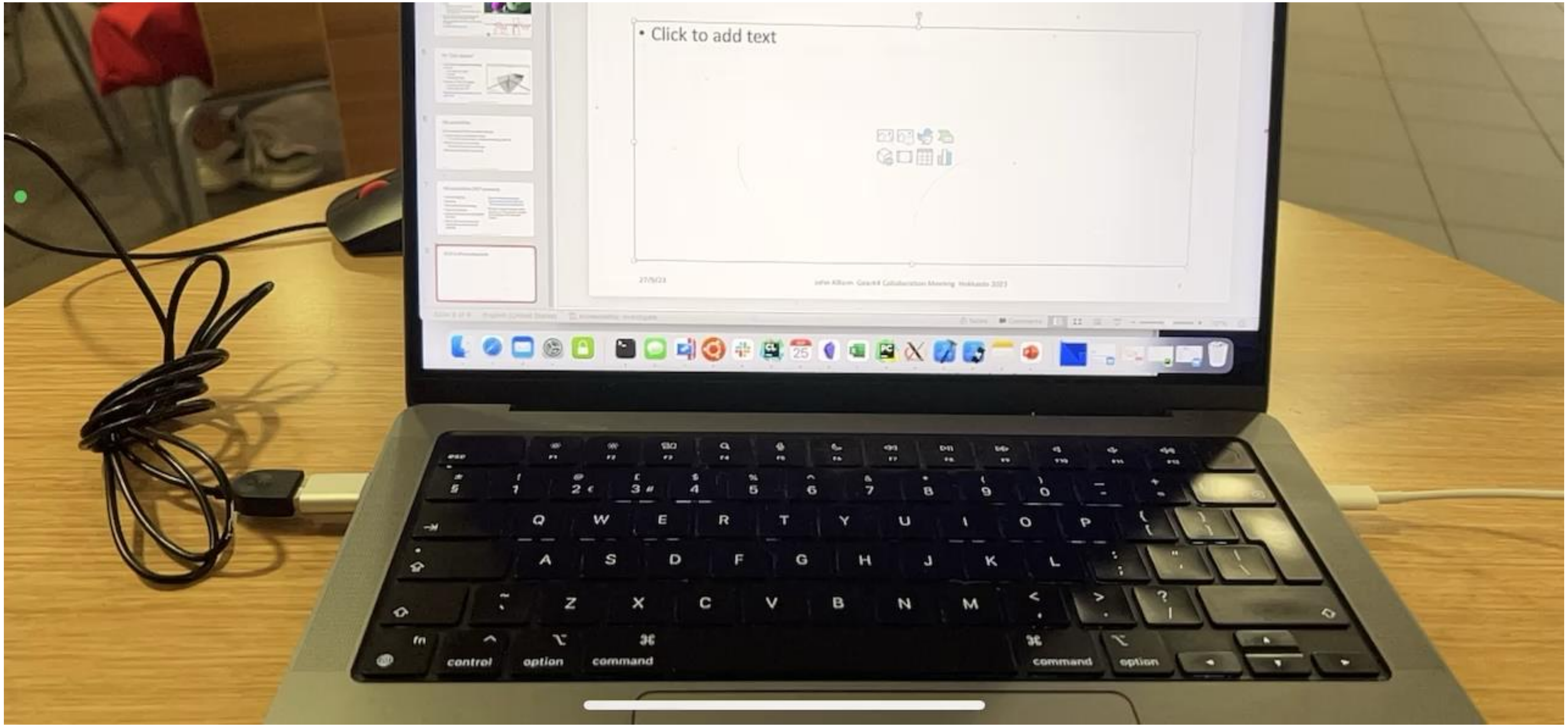
- Modern interface to HEP geometry has lots of applications
  - Simple interface to high end rendering (ospray, pbr)
  - Simple interface to multi-physics codes and visualization (paraview, visit, comsol, ansys etc)
  - Easy interface for testing new techniques (Optiks, Mitsuba etc)
  - Closer connection between engineering and HEP
  - Ability to load and write STEP files

3D Model directly in power point

Model directly in power point (LUXE experiment)



XR model export



Potential improvements

- Toolkit developed (responding to need) over 6 years. Interface could be made more uniform
  - G4 has many constructors for geometry
- More convivence methods for modifying geometry
- “advanced python” dynamic remeshing upon parameter updates (cyclic dependency graph)
- More *pythonic* or at least *syntactically sweet* interface to parameters
- More informative __str__ and __repr__ methods

The University of Manchester

Links and information

- Paper
  - https://doi.org/10.1016/j.cpc.2021.108228
- Online manual
  - https://pyg4ometry.readthedocs.io
- Code repo
  - https://github.com/g4edge/pyg4ometry

Conclusions/summary

- You can do so much with a stable API and small number of algorithms
  - Pyg4ometry of utility to many people using Monte Carlo particle transportation
  - Collaboration always welcoming new people. **Luigi** developed CI/CD and advanced sk-build implementation, pip deployment
  - Work like this is informing MC tracking and visualisation work