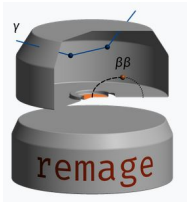
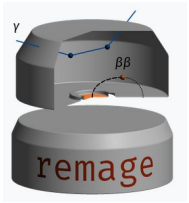


Simulations of underground physics experiments with *remage*

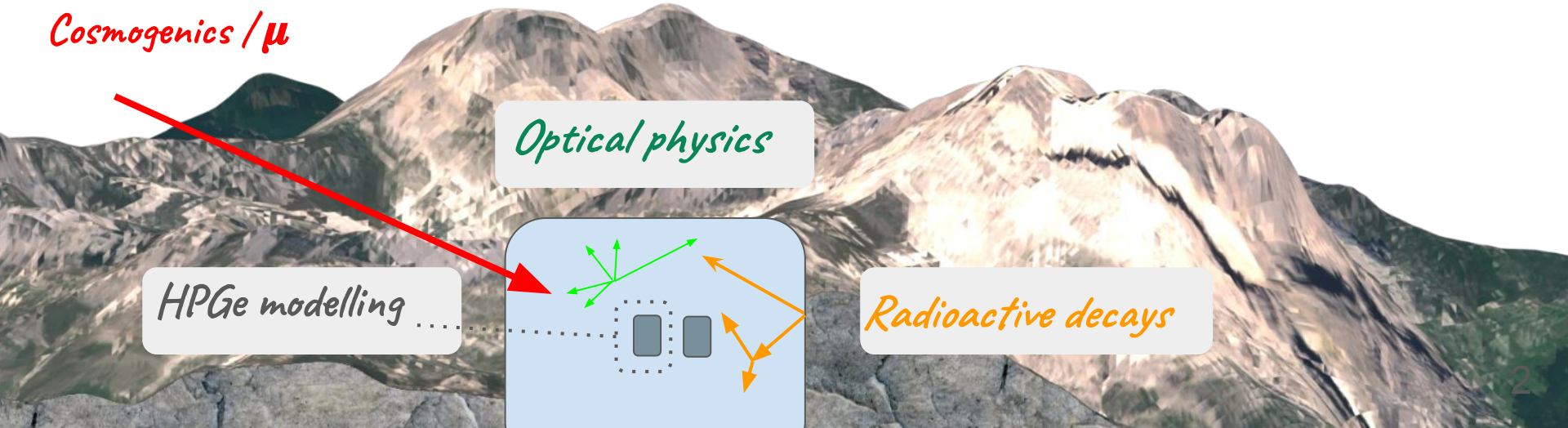
Toby Dixon {toby.dixon.23@ucl.ac.uk}

University College London

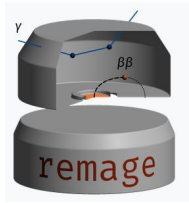
Geant4 simulations of low background experiments



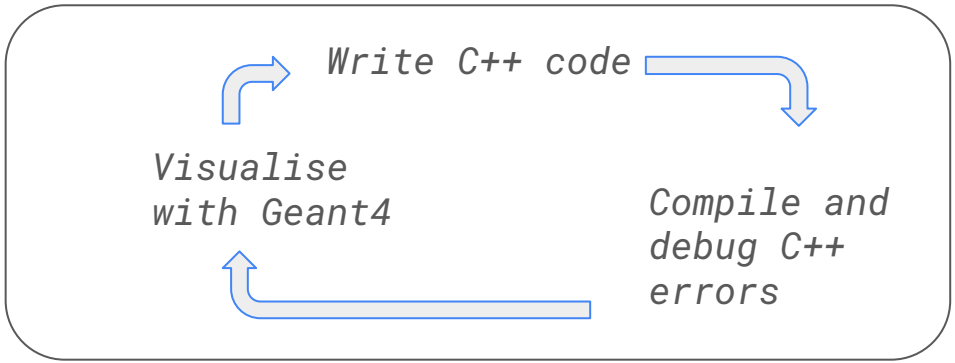
- Geant4 simulations are critical for low background physics experiments to
  - Model backgrounds
  - Predict signal efficiencies
  - But also for **radioactivity screening!**
- Review by Luigi Pertoldi in a past GDR [link]
- I will focus on HPGe and LAr detectors:
  - **But most of the concepts and software can be used for any low background experiment!**



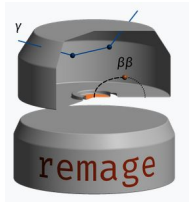
Motivation and background



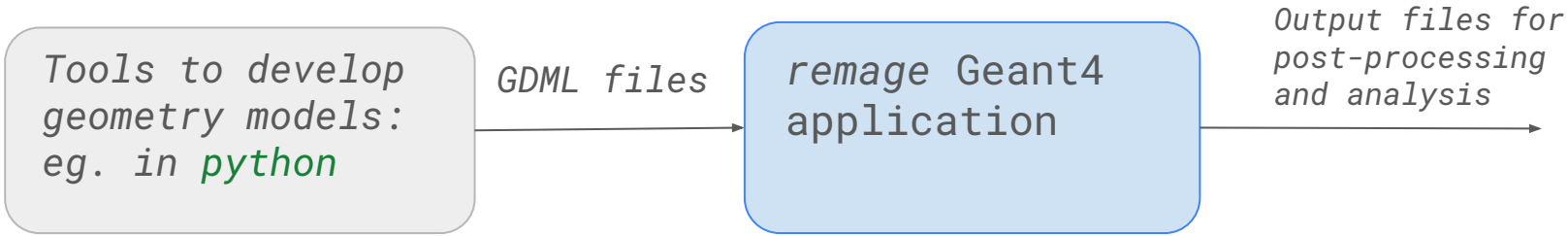
- The “typical” Geant4 application:
  - Full C++ code developed by each user
  - Typically lots of code, mostly to build geometry
  - Difficult to maintain and keep up to date with new Geant4 versions
- Convoluted workflow to develop geometry:



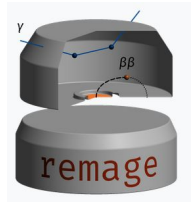
remage



- Developed mainly by [members] of the LEGEND collaboration
- Aims to make a generic framework decoupling geometry from the C++ code
- Allows to easily share the same framework across many experiments!
- Much more:
  - Use modern Geant4 features
  - Extensive documentation
  - Automatically generated validation suite



Developing geometry



- *remage* does not directly handle geometry!
- Interface via:
  a. GDML*: XML based file format to describe fully the geometry (*recommended*)
  b. C++ code linking against *remage*
- pyg4ometry [link] is a nice tool to write GDML in python:
  a. Very easy to use / learn
  b. Fast feedback loop
  c. Lots of [documentation]

Very similar interface to Geant4 C++

.cxx

```
G4Tubs* tubs = new G4Tubs("tubs",
                          0 * mm,
                          50 * mm,
                          50 * mm,
                          0,
                          360 * deg);
```

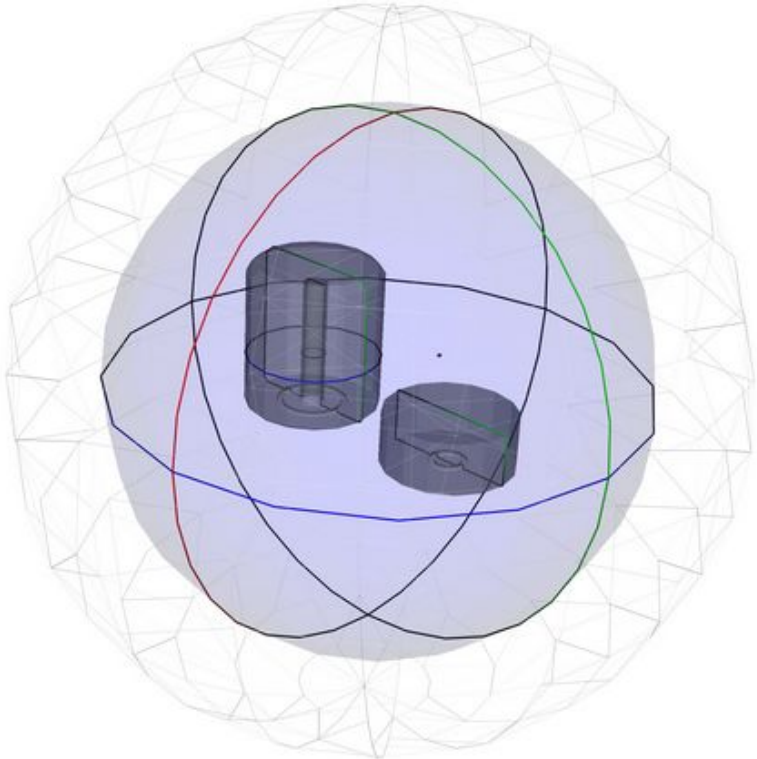
.py

```
tubs = pg4.geant4.solid.Tubs(
                        name = "tubs",
                        pRMin = 0,
                        pRMax = 50,
                        pDz = 50,
                        pSPhi = 0,
                        pDPhi = 2*np.pi,
                        lunit = "mm",
                        registry = reg )
```

** Geometry Description Markup Language*

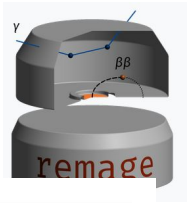
Developing geometry cont.

- pyg4ometry has in built high quality visualisation software (VTK)
- Interactive feedback while generating geometry
  - No need to compile!
  - Easy to run on your laptop even when Geant4 isn't available!
- Full example in the *remage* [tutorial]!

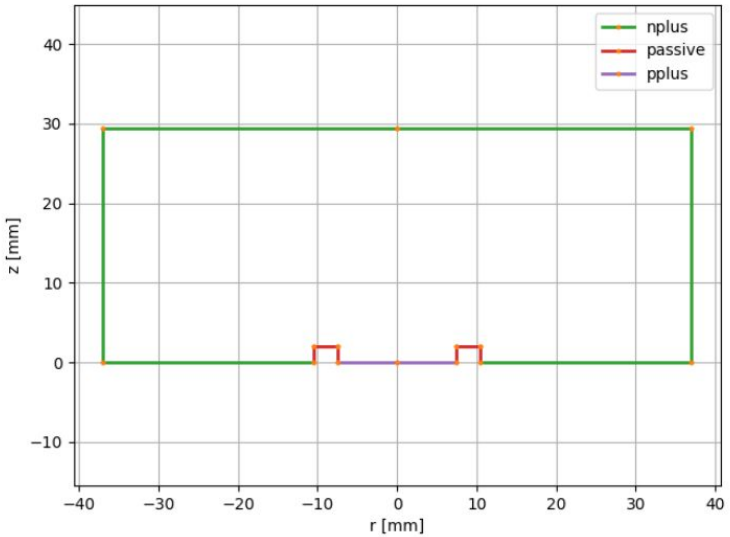


Fast learning curve for new users / students!

Tools to help with creating geometry

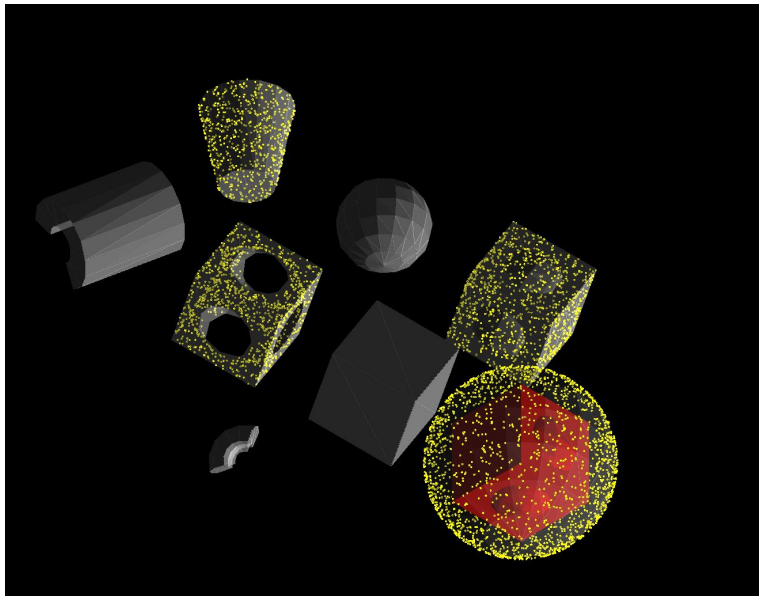


- Lots of tools created to help handle creating geometry!
- For example legend-pygeom-hpges [link] is a package to handle HPGe detector geometry
  a. Starts from a standard **configuration file** (eg. height, radius detector type etc.)
  b. Constructs the G4 **logical volume**
  c. Has a series of useful methods eg:
     - Mass / surface area of the detector
     - Functions to plot the profile
     - Distance of a point from the surface



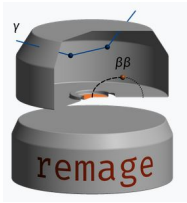
Features of the *remage* application

- Full functionality for radiogenic and optical simulations
- Includes some “tricky” parts of code that often take time to develop:
  - Generic sampling on the surface of any solid [link], very important for simulating radon daughters!
  - Handling of decay chains
- Output file as either ROOT or HDF5 files based on the LEGEND [specification], including physical units!



Example from the validation report

Running *remage*



Geant4 macros:

- Users interact with via a GDML file and a Geant4 macro
- **Common issue:** “what are the available macro commands?”
- Created an online documentation page with a full description of every macro command

Running remage:

- There is no need to install *remage* or Geant4 from scratch!
- Available in a container with all dependencies
- Very fast to set up on Linux systems [link to instructions]

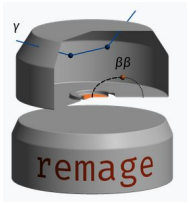
/RMG/Manager/Interactive

Enable interactive mode

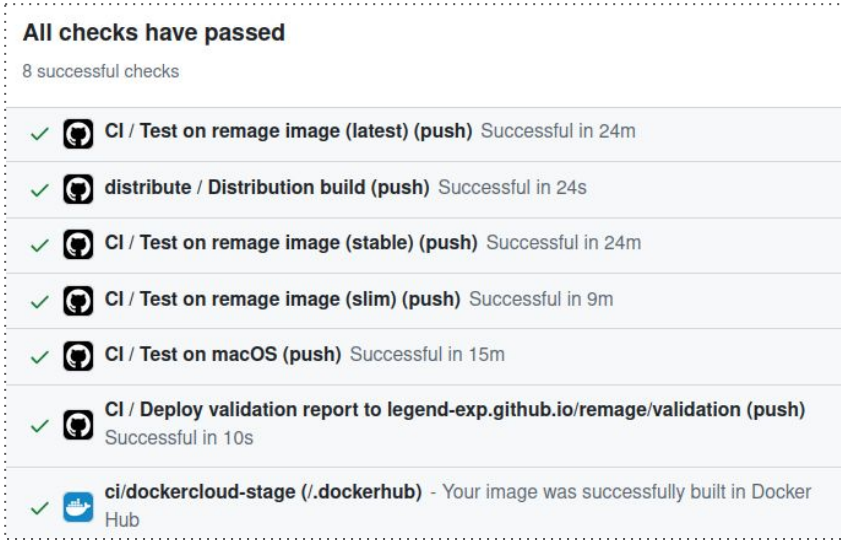
- **Parameter** – `boolean`
  - **Parameter type** – `b`
  - **Omittable** – `True`
  - **Default value** – `true`
- **Allowed states** – `PreInit Idle`

Example macro documentation

Modern Geant4 features

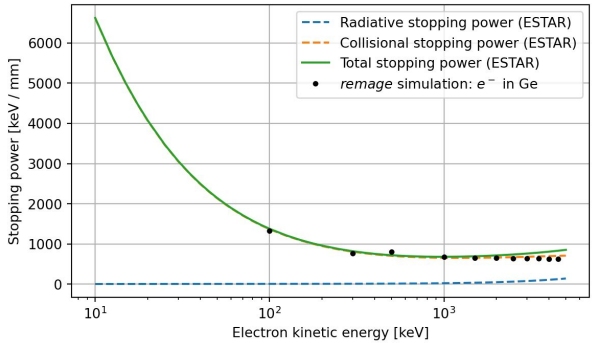


- Geant4 is constantly improving:
  - Better and faster physics models
  - New features
  - Some very useful features like multithreading
- *But for most users it is tricky to keep up with Geant4 changes...*
- *remage* uses automated testing run as github actions to keep the code updated with new Geant4 versions
- **Also checks there are no changes having an effect on the functionality!**

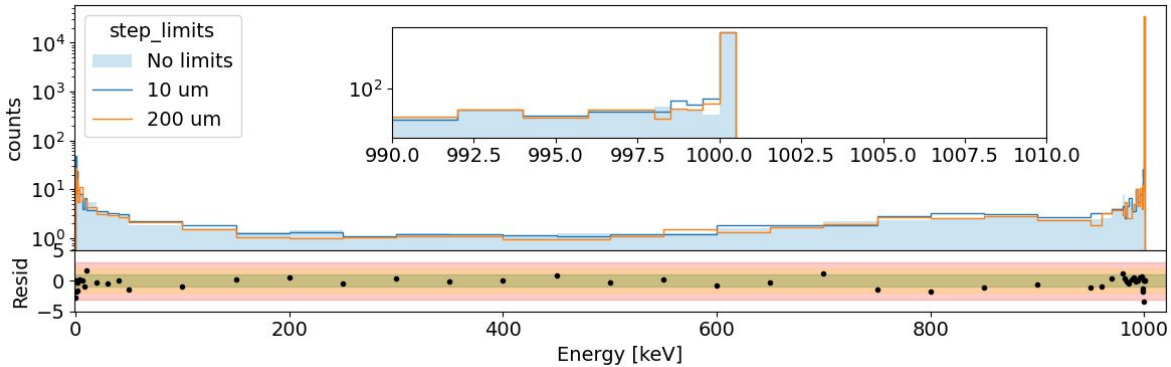


Validation

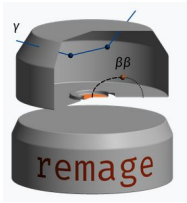
- As part of this we have an **automatically generated validation suite** and [online document] with the results
- Runs both a series of **tests** and some basic simulations
- Still under active development but includes:
  a. Checks on the generation of primary vertices
  b. Dependence of basic observables on user choices
  c. Comparison to external databases (ESTAR)



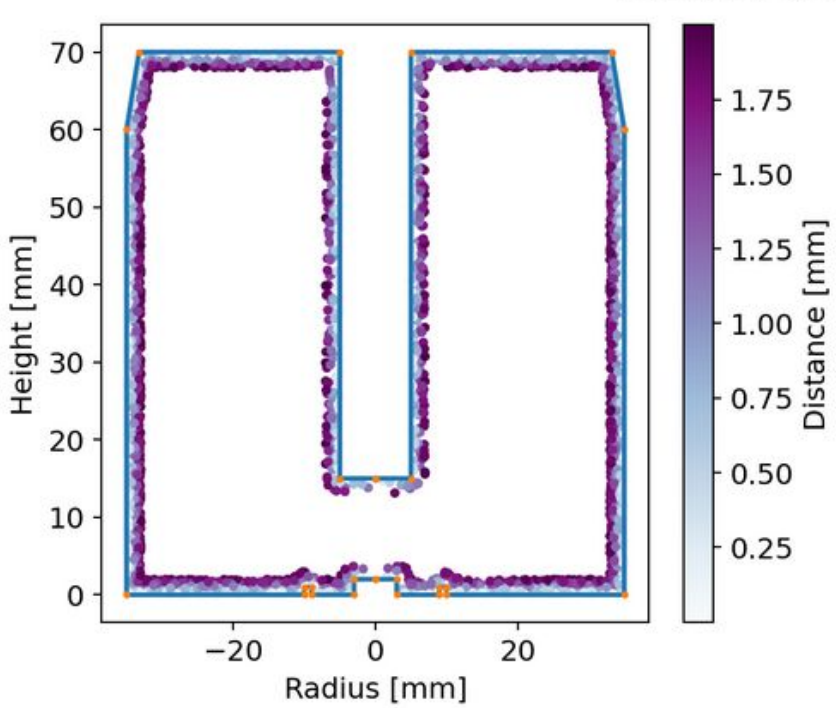
Some examples



Post processing and detector response



- But HPGe detectors are complex objects!
  - **Surface** response
  - Internal **fields** and pulse shape discrimination
- Much of the detector response can be handled as a “post-processing” step of the simulation
  - Avoids the need to rerun the simulation
- *python* package *reboost* contains tools to apply detector response
- Tutorial and documentation [here]



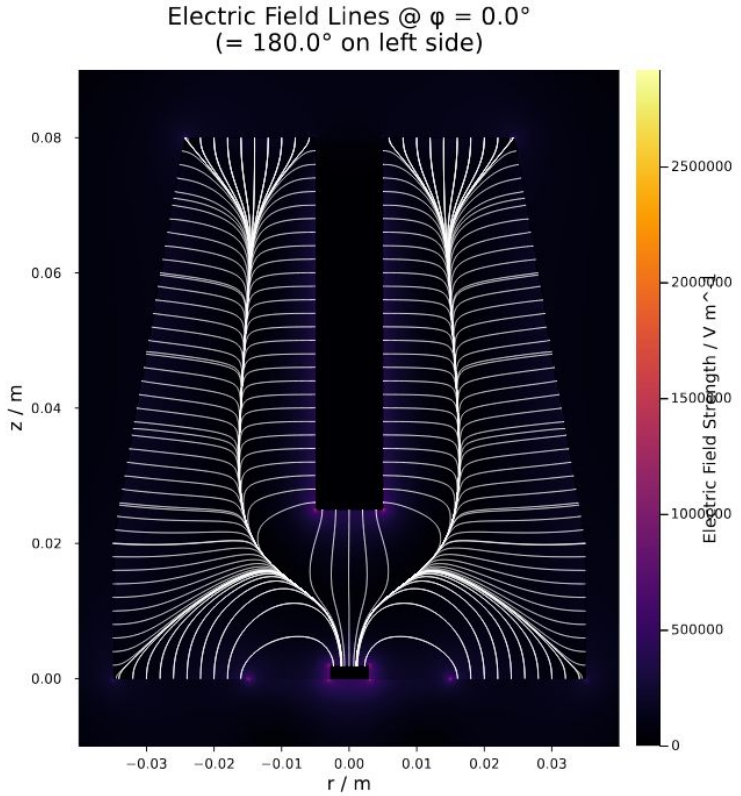
Simulating HPGe response

Lots of packages to simulate the HPGe response:

- Agata Detector Library (EPJA 52, (2016) 700.)
- *siggen* [link]
- SolidStateDetectors.jl [link]

Another fully open source package!

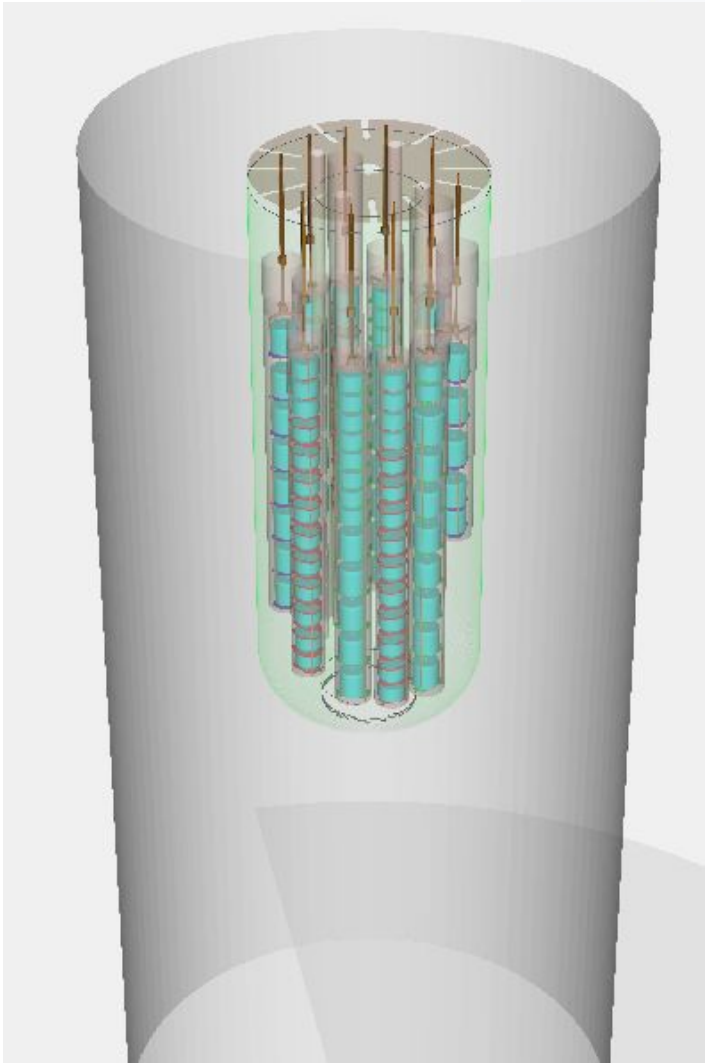
- Easy to read *remage* files into julia!
- Can be used to generate waveforms for events of interest



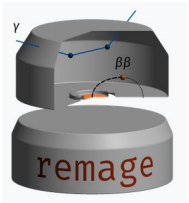
Taken from [link]

Example: LEGEND -200/1000 geometry

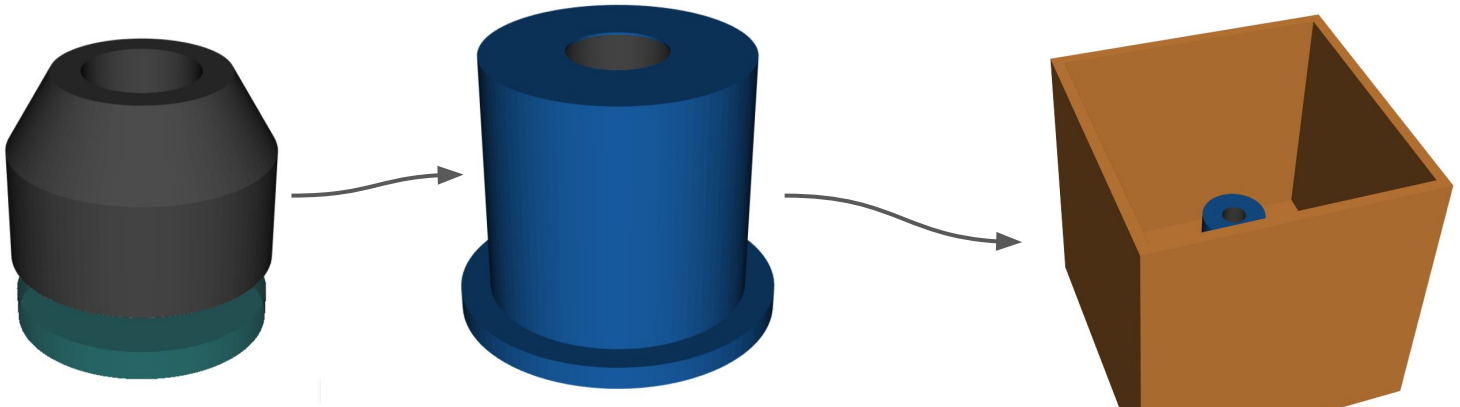
- Full implementation of both LEGEND-200 and LEGEND-1000 in **open-source** packages [link-l200,link-l1000]
- Private information located in other packages
- Can be run with a public geometry (no private HPGe information)
- All optical properties attached to the output GDML files



remage for HPGe screening



- The package is very well suited to gamma assay
- Easy to generate a typical screening station geometry [example]
- Also easy to generate new geometry:
  - eg. the sources themselves without changing Geant4 code
- Users can profit from a joint code-base, no need for everyone to reinvent the wheel!

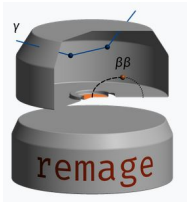


HPGe and lead shield

Al shielding

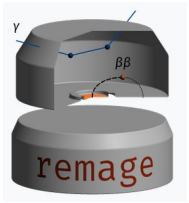
Cu shield

Optical simulations



- Challenging → very CPU intensive
- There are also lots of unknown parameters → need to rerun simulations multiple times
- Two strategies to speed up the simulation:
  a. Only propagate photons when energy deposited in Ge, or other “interesting” events [macro-link]
  b. Generate optical map [reboost-link]
- Could be interesting for other experiments using Scintillators as a veto!

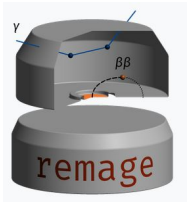
Some lessons learned



A few things we found while developing *remage* that might be useful!

1. File size is important
    - The total run time can be dominated by writing data to disk
    - [Clustering] of Geant4 steps implemented to reduce this
2. Geant4 step lengths are important
    - Default step sizes can lead to a bias in total reconstructed energy in HPGe
3. Coding in python is much faster
    - Better to move everything possible outside of the Geant4 app (also event generators etc.)

Summary



- We developed a new open source simulation package for low background experiments
- Designed for HPGe's in LAr but could work well for a variety of experiments
- Particularly promising approach for HPGe screening (constant need to write new geometry)

All software is open source and so we are happy to have new users or contributors!