CombLayer

a fast CSG geometry builder

Stuart Ansell
and
Konstantin Batkov

MAX IV Laboratory
Lund University

October 28, 2024
Karlsruhe

Outline

Outline

Introduction

The purpose of CombLayer

To easily and rapidly build complex geometric models
which are fast to run
in Monte Carlo codes
utilising Constructive Solid Geometries.

Introduction

CombLayer allows:

- prepare input files with C++, defining:
  - geometry
  - materials and their mixtures
  - source term
  - estimators
  - physics settings
  - magnetic fields
  - variance reduction
  - *everything — no need to post-edit the generated input files*
- export models into different formats:
  - FLUKA
  - MCNP
  - PHITS $\rightarrow$ STEP
  - POV-Ray $\rightarrow$ STEP
  - VTK $\rightarrow$ ROOT (see my mc-tools talk)

Introduction

CombLayer helps you build
easily manageable
fully parametric and fast Monte Carlo models.

Introduction

- Main author: Stuart Ansell
- since 2011
  - KB: since 2015
- $\sim$ 500 k lines of code
- GPL 3 licence
- Daily used at MAX IV, ESS, ISIS
  - \+ some projects at SNS, Delft and PIK
- Very stable code
  - is used to build detailed models of facilities

https://github.com/SAnsell/CombLayer

Outline

Geometry

- Geometry is described in object-oriented approach
  - whole model is built from objects of C++ classes
- Individual components are assembled together much like the LEGO bricks
- Each component is described in its own coordinate system
  - can be placed anywhere with arbitrary orientation
  - can be flipped with respect to arbitrary plane
- Each component is described in its own surface and region space
  - no need to care about overlaps in surface/region naming with other components
- Each component can be reused multiple times
  - copies may differ from each other
- Depending on needs, only specific parts of the whole geometry can be built
  - significantly improves tracking speed
- Whole model can be arbitrary rotated and translated

Geometry

- C++ allows a user to benefit from its object-oriented programming approach:
  - reuse the code through inheritance and polymorphism
- This enables rapid geometry construction, which is easy to modify
  - all variables (e.g. geometry dimensions and materials) are parametric and can be changed runtime (as command line arguments or .xml file entries)
    - optimisation calculations
    - modelling various scenarios
- Big warehouse of already-built components and a library of pre-defined materials
  - accelerator, reactor, neutron guides, x-ray beamlines
  - shared among users

Geometry optimisation

- CombLayer performs a two factor Boolean optimisation to minimise the number of literals in the region description
  - removal of duplicated implicants
  - partial disjunctive normal form optimisation
- In intersection it removes unnecessary surfaces and allows the regions to be split or merged as needed minimising Monte Carlo run-time cost
  - e.g. complex-shaped walls can be split by layers for importance biasing

- Intersection of components is done just by specifying which component goes into which one
  - all the low-level math is handled by the code, allowing users to spend their time more efficiently

```
Spoon->InsertTo(Cup);
```

Geometry
Tracking system

CombLayer has a built-in geometry tracking system, that allows

- internal geometry debugging
- efficient intersection of components
- removal of zero-volume regions
- weight-window generation (see next slide)

Biasing

CombLayer has a built-in deterministic mesh-based weight window generator for FLUKA, MCNP and PHITS

- 3D mesh of importances superimposed with geometry
- handy for deep penetration calculations
- not as powerful as e.g. ADVANTAG, but very quick to setup and run
- nested meshes are possible

Outline

Export

- As indicated above, one can define whole input file with CombLayer and export it to one of the listed Monte Carlo codes (MCNP, FLUKA, PHITS), and POV-Ray and VTK
- However, due to difference between Monte Carlo codes, it is sometimes not possible to export exactly the same model to different codes
  - some estimators and surfaces exist in some codes, but not in the others
  - source term, physics and biasing settings don't completely overlap between the codes

Export same settings to different Monte Carlo codes

- Geometry
  - geometry of any complexity can be exported to all codes provided that all required surfaces are supported by the given code
    - i.e. single cones exist in MCNP but not in FLUKA
- Materials
  - material composition can be exported to all codes
  - temperature and $S(\alpha, \beta)$: only MCNP and PHITS
- Estimators and Physics settings
  - typically to be individually defined for each code
- Magnetic fields
  - currently, only FLUKA
- Biasing
  - to be individually defined for each code
  - \+ built-in weight window generation for all supported codes

POV-Ray
Persistence of Vision Raytracer

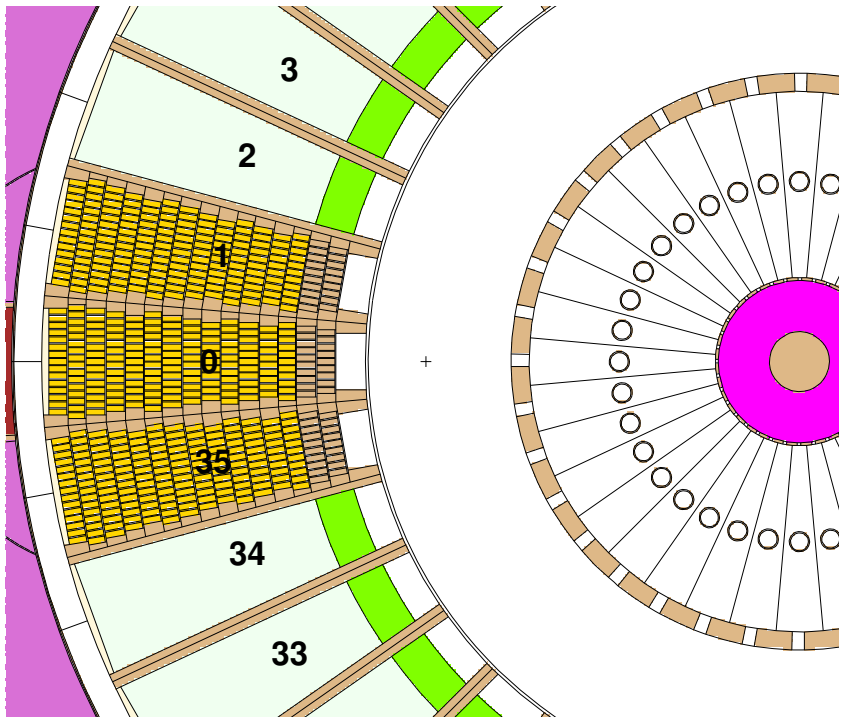
- POV-Ray is a CSG-based raytracer that produces realistic 3D images

- CombLayer only exports geometry and materials as simple textures, so all scene setup must be prepared manually
- http://povray.org

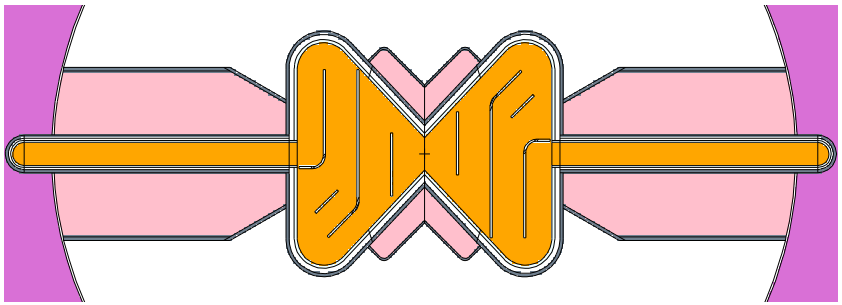
Outline

Examples
ESS target wheel

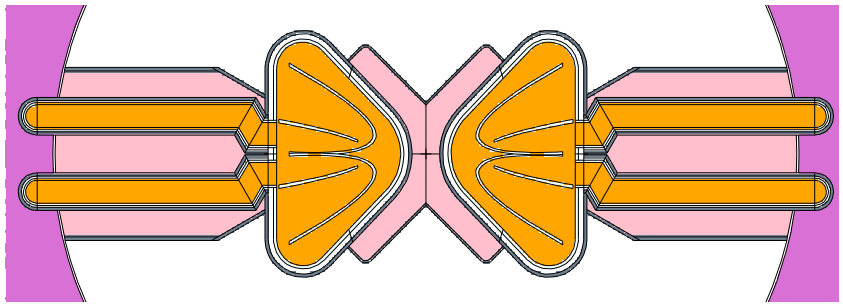
- The target wheel consists of 36 sectors (numbered)
- Each sector is a C++ class instance
  - $\Rightarrow$ the wheel is made of 36 copies of the same object
- All sectors are made of Tungsten bricks (see sectors 0, 1, 35)
- The brick surfaces and regions are defined with the `for` loops
- To speed-up calculations, other sectors are made of homogenised Tungsten
  - the switch is implemented just by setting the `BricksActive` variable to `false` in the command line



Examples
ESS moderators



Moderator 1



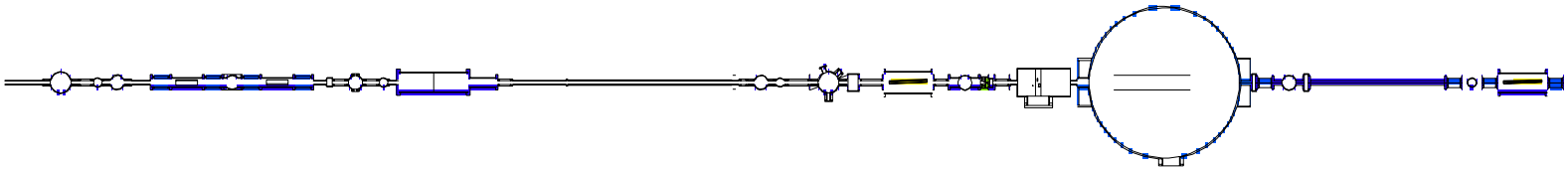
Moderator 2

- To maximise neutron production, the ESS target-moderator-reflector assembly have been optimised
- Different geometries of neutron moderators were studied
  - two of them are shown above
- The moderator geometry is selected by setting a single command line variable, i.e.

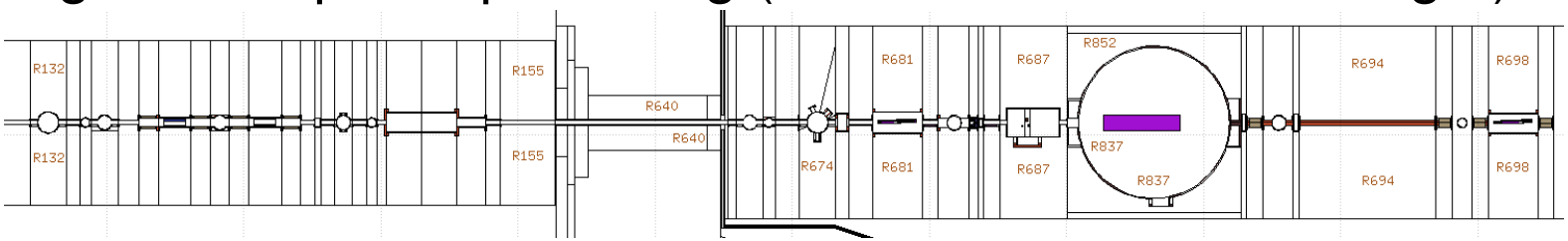
```
-v ModeratorType BF1
```

Examples
Accelerator beamline

- Part of an accelerator beamline with complex outer region:



- The same geometry with outer region split into simpler regions to speedup tracking (and reduce the DNF length):



- CombLayer does it semi-automatically, almost without user intervention
  - only need to indicate the bounding box dimensions
  - and the start/end surfaces of each component
    - this is anyway needed to attach other components

Conclusion

- CombLayer is a powerful and user-friendly tool for building fully parameterised and fast CSG Monte Carlo models
- Source code and documentation: https://github.com/SAnsell/CombLayer