Thoughts on the use of gemc for SBS+BB Monte Carlo

Andrew Puckett

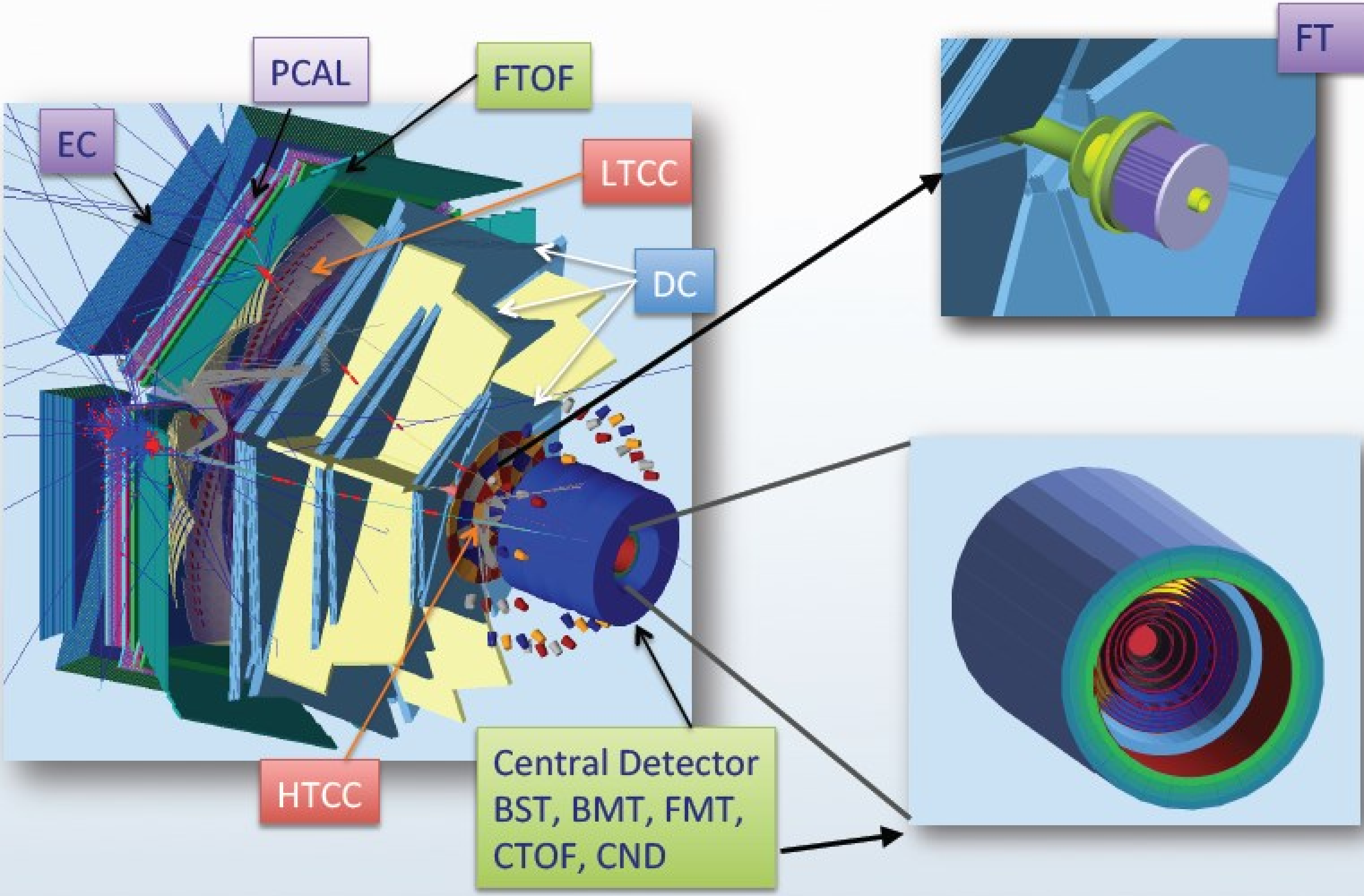
SBS Weekly Meeting

June 19, 2013

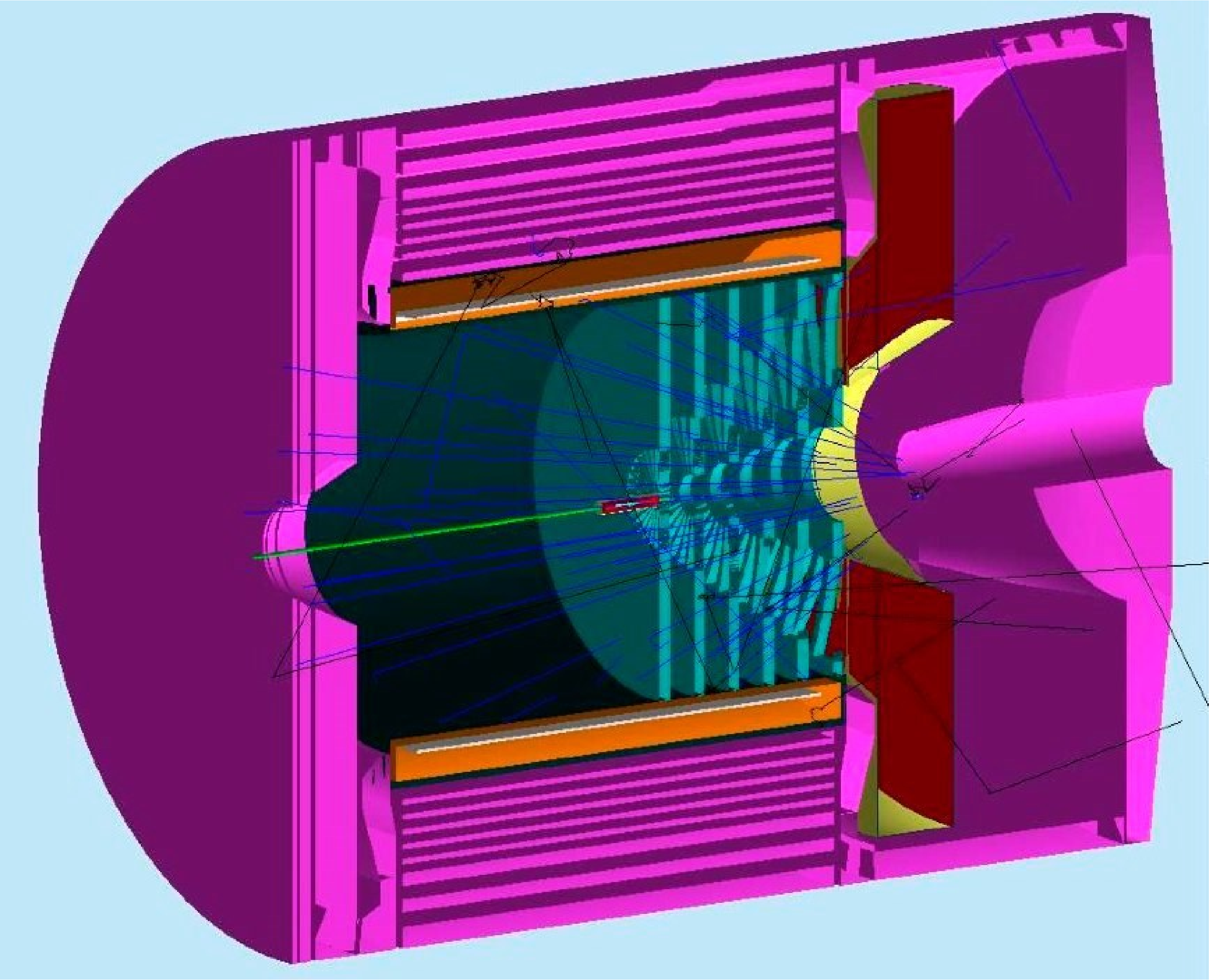
gemc—what is it?

- gemc is a “full-featured” GEANT4 application and software framework for the simulation of arbitrary detector geometries
- Philosophy is to minimize interaction of the end user with the source code—detector geometry is built from a database at run-time—facilitates rapid development and deployment, contributions from multiple developers
- (Almost) completely general detector simulations can be built without recompiling the source code—there are several exceptions to this, but most will be eliminated in the near future
- Primary author and lead developer—M. Ungaro, Hall B staff scientist
- Documentation and installation instructions: https://gemc.jlab.org/gemc/Home.html
- Adopted collaboration-wide as the GEANT4 simulation framework for CLAS12 spectrometer in Hall B

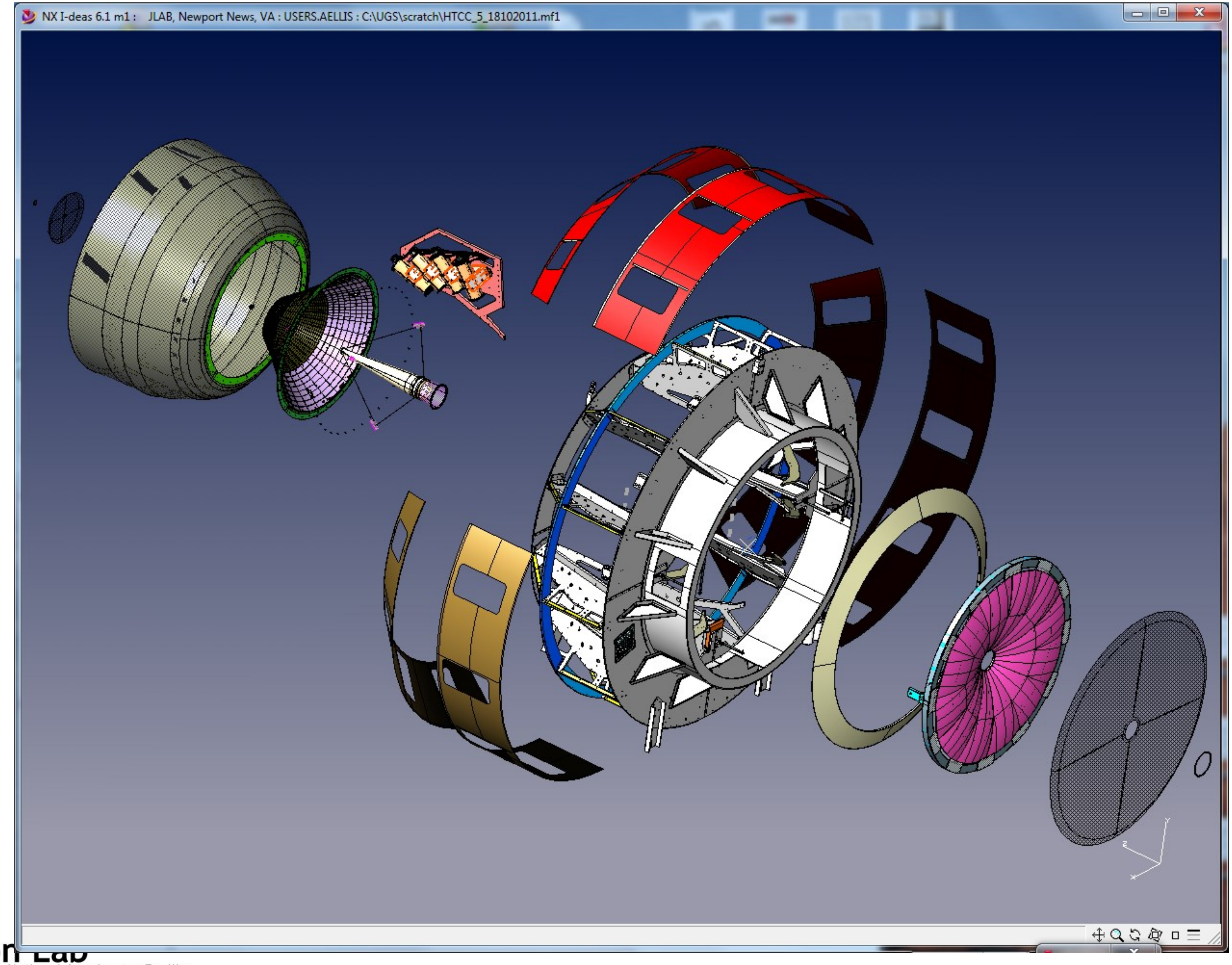
CLAS12 in GEMC



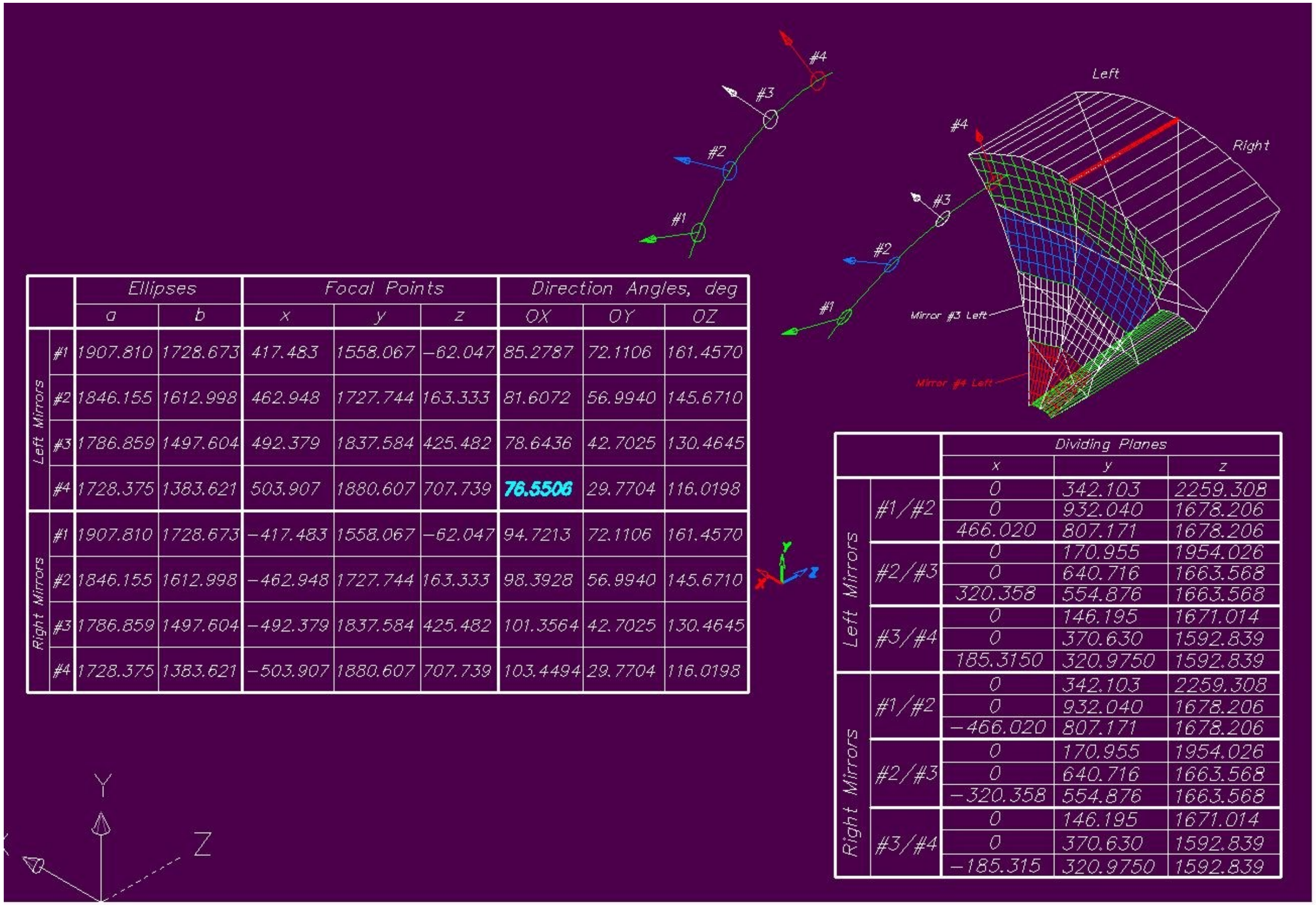
Hall A SoLiD Simulation



Detector construction example: CLAS12 High Threshold Cherenkov Counter



HTCC Geometry: Mirrors

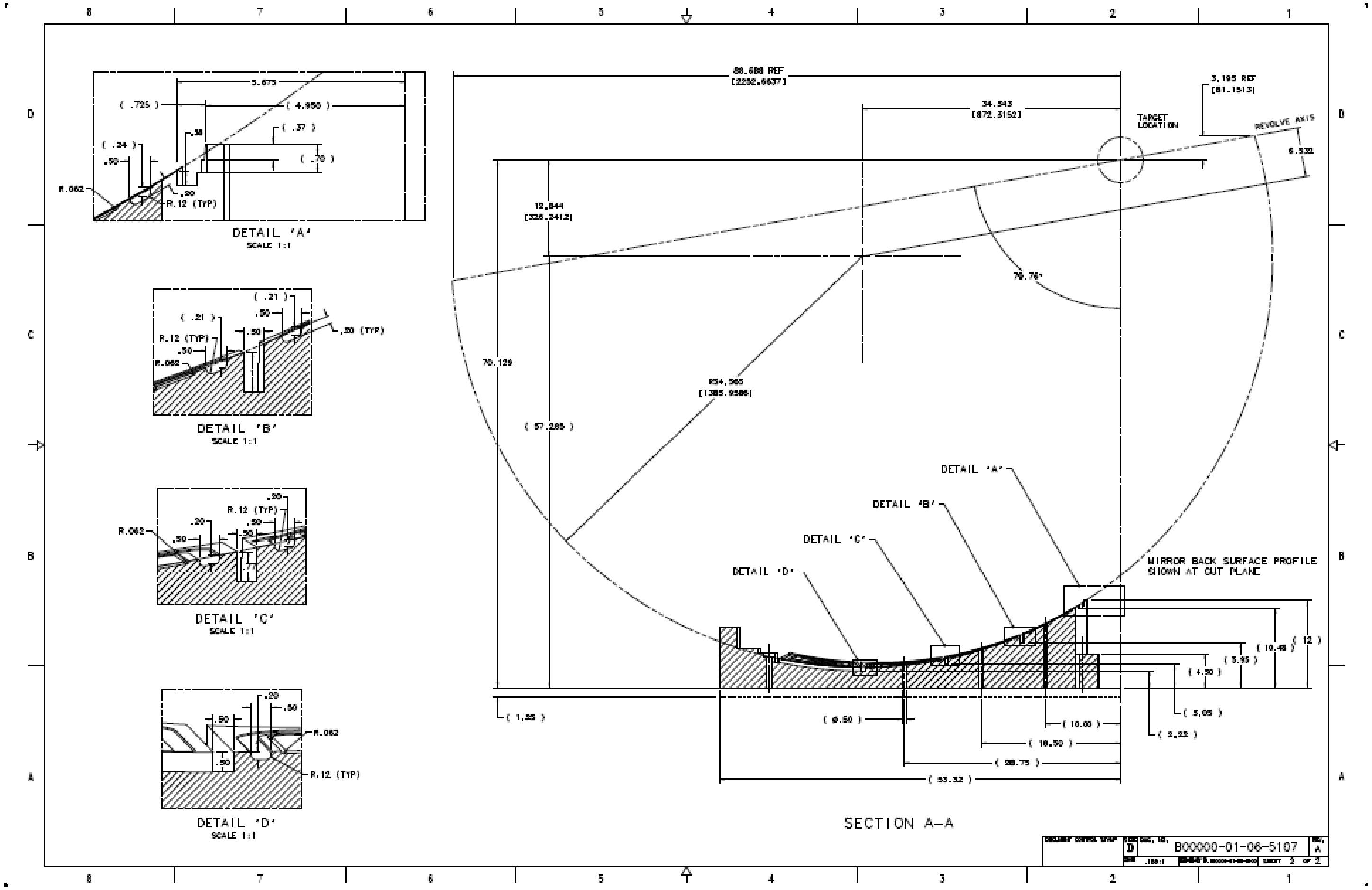


		Ellipses		Focal Points			Direction Angles, deg		
		a	b	x	y	z	OX	OY	OZ
Left Mirrors	#1	1907.810	1728.673	417.483	1558.067	−62.047	85.2787	72.1106	161.4570
	#2	1846.155	1612.998	462.948	1727.744	163.333	81.6072	56.9940	145.6710
	#3	1786.859	1497.604	492.379	1837.584	425.482	78.6436	42.7025	130.4645
	#4	1728.375	1383.621	503.907	1880.607	707.739	76.5506	29.7704	116.0198
Right Mirrors	#1	1907.810	1728.673	−417.483	1558.067	−62.047	94.7213	72.1106	161.4570
	#2	1846.155	1612.998	−462.948	1727.744	163.333	98.3928	56.9940	145.6710
	#3	1786.859	1497.604	−492.379	1837.584	425.482	101.3564	42.7025	130.4645
	#4	1728.375	1383.621	−503.907	1880.607	707.739	103.4494	29.7704	116.0198

		Dividing Planes		
		x	y	z
Left Mirrors	#1/#2	0	342.103	2259.308
		0	932.040	1678.206
		466.020	807.171	1678.206
	#2/#3	0	170.955	1954.026
		0	640.716	1663.568
		320.358	554.876	1663.568
	#3/#4	0	146.195	1671.014
		0	370.630	1592.839
		185.3150	320.9750	1592.839
Right Mirrors	#1/#2	0	342.103	2259.308
		0	932.040	1678.206
		−466.020	807.171	1678.206
	#2/#3	0	170.955	1954.026
		0	640.716	1663.568
		−320.358	554.876	1663.568
	#3/#4	0	146.195	1671.014
		0	370.630	1592.839
		−185.315	320.9750	1592.839

Mirror is divided in 12 φ and 4 θ segments—ellipsoids intersect in planes allowing full solid-angle coverage with no gaps/shadowing

HTCC Geometry: Mirror back surface

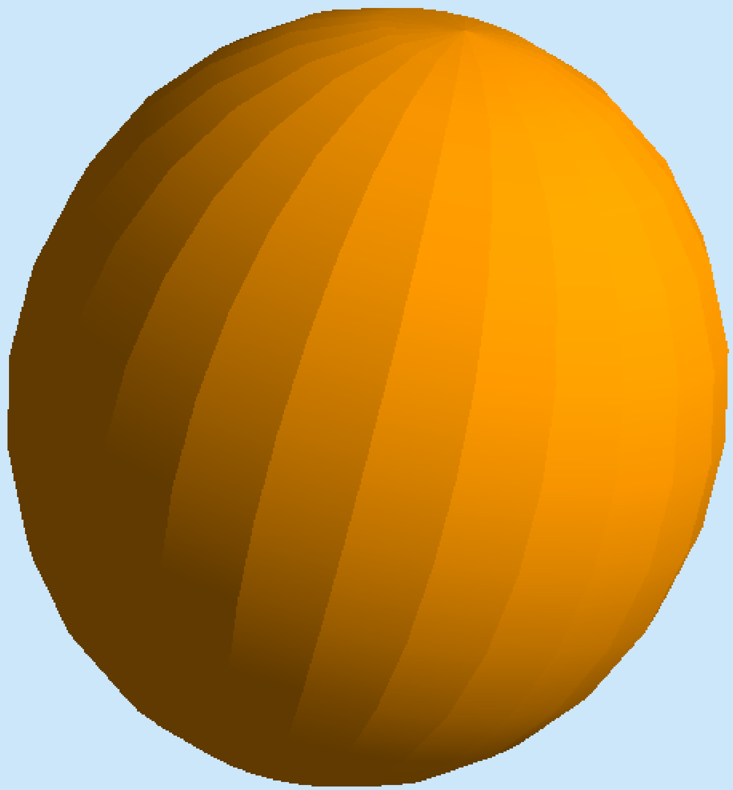


To simplify construction/assembly, mirror back surfaces have a common "barrel" geometry—obtained by revolving a circle about a chord that is not a diameter

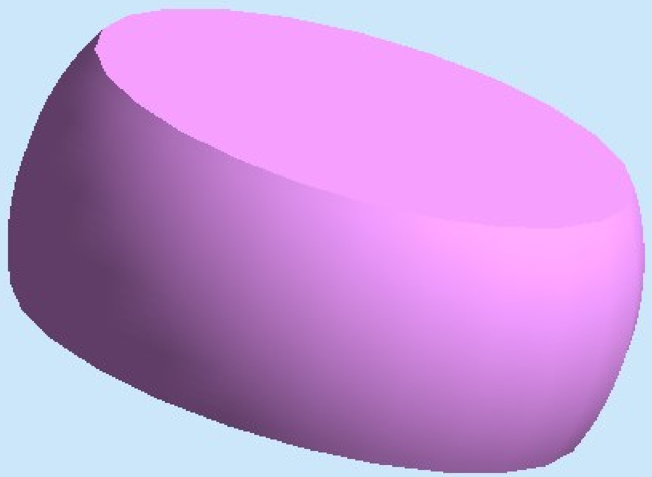
Building the HTCC Mirrors in gemc

- Complicated geometry cannot be built from standard GEANT4 solids alone: requires standard solids AND Boolean operations:
  – Solids: Tube, Cons, Polycone, Ellipsoid, Box
  – Operations: Subtractions and Intersections
  – Mirror back surface (surface of revolution of a circle about a chord that is not a diameter) cannot be described with standard GEANT4 solids; requires approximate description using "polycone"
  – Mirror segments are divided by planes, necessitating additional Boolean operations.
  – Procedure for Mirror #1 outlined below (mirrors 2/3/4 similar)

HTCC mirror geometry in gemc

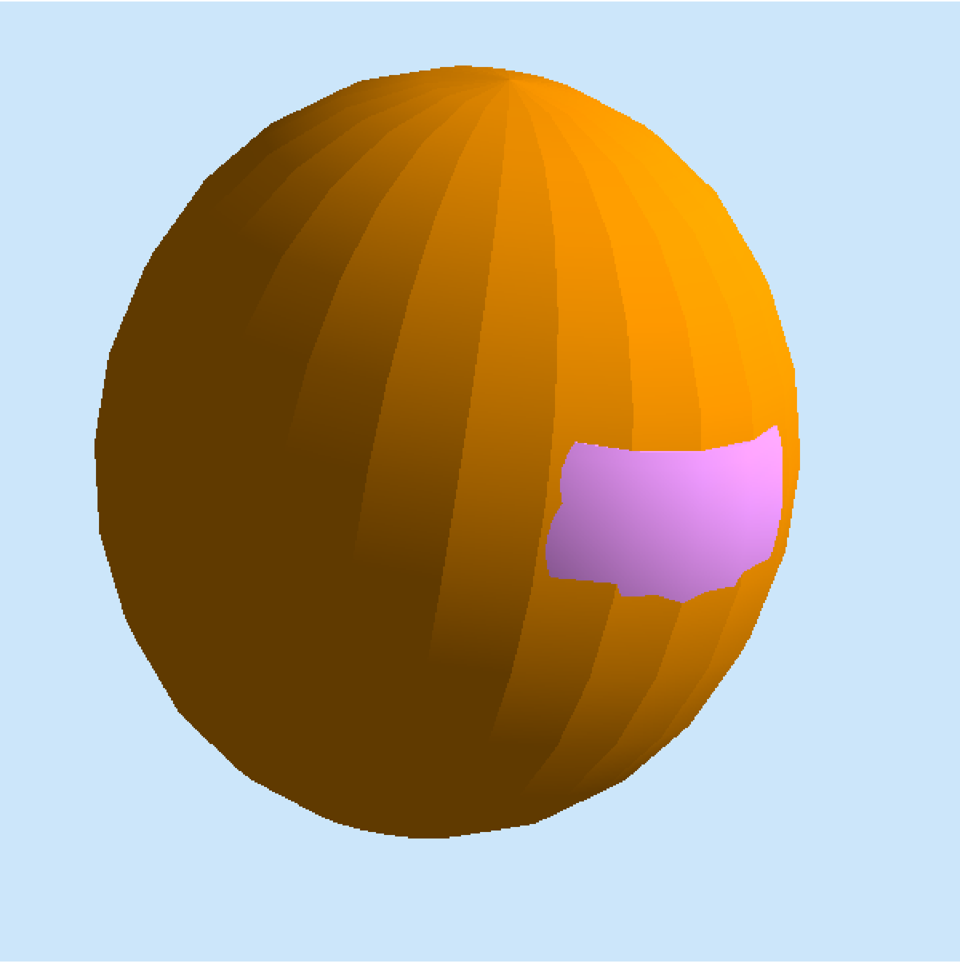


Step 1: Ellipsoid w/placement

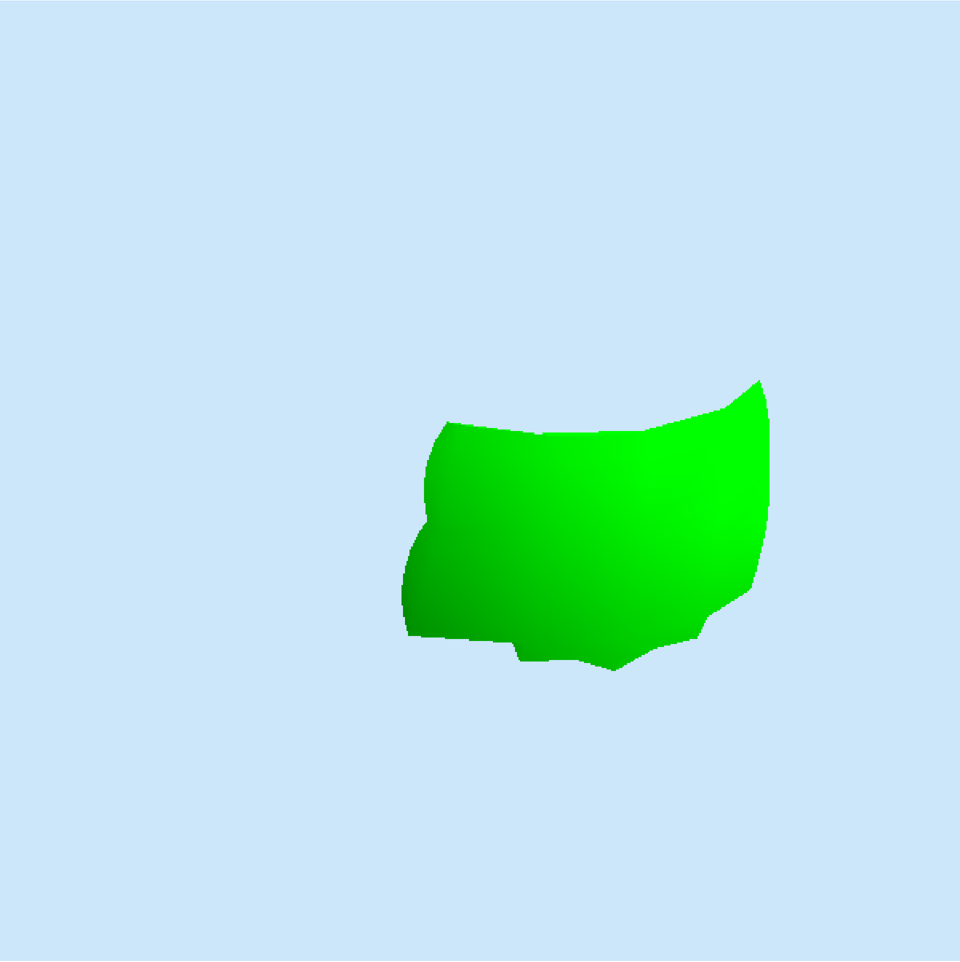


Step 2: Barrel w/placement

HTCC mirror geometry in gemc

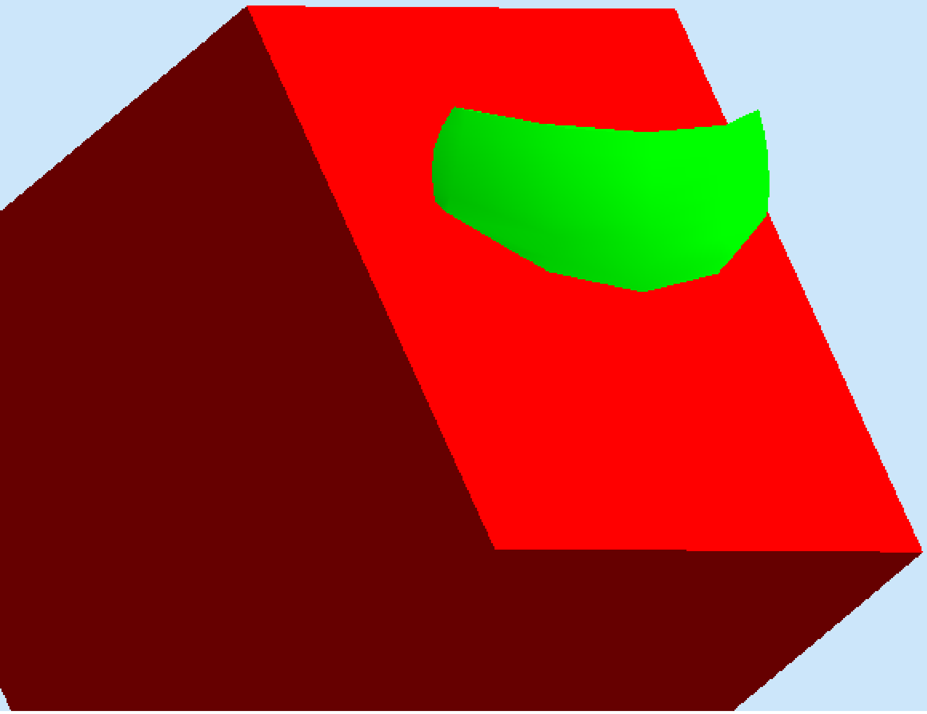


Step 3: Barrel and Ellipsoid w/placement

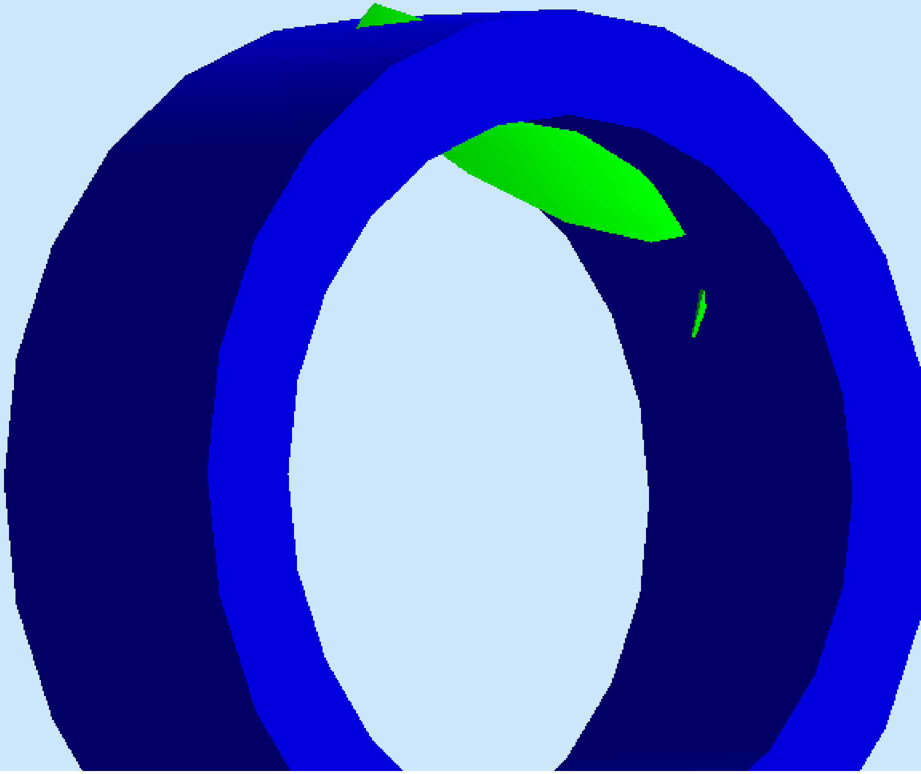


Step 4: Subtraction of Ellipsoid from Barrel

HTCC mirror geometry in gemc



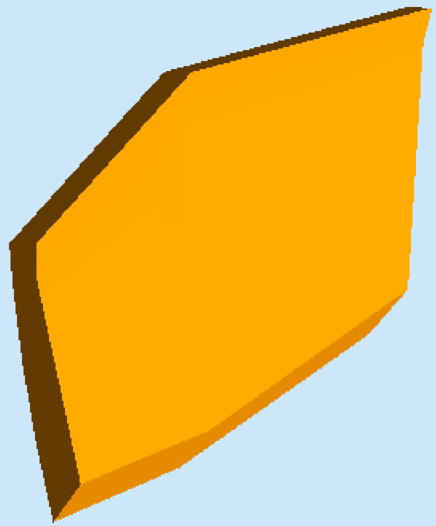
Step 5: (Barrel – Ellipsoid) - Box (Mirror #1/#2 dividing plane)



Step 6: (Barrel – Ellipsoid – Box – Cylinder) (outer edge at $\theta=35^\circ$)

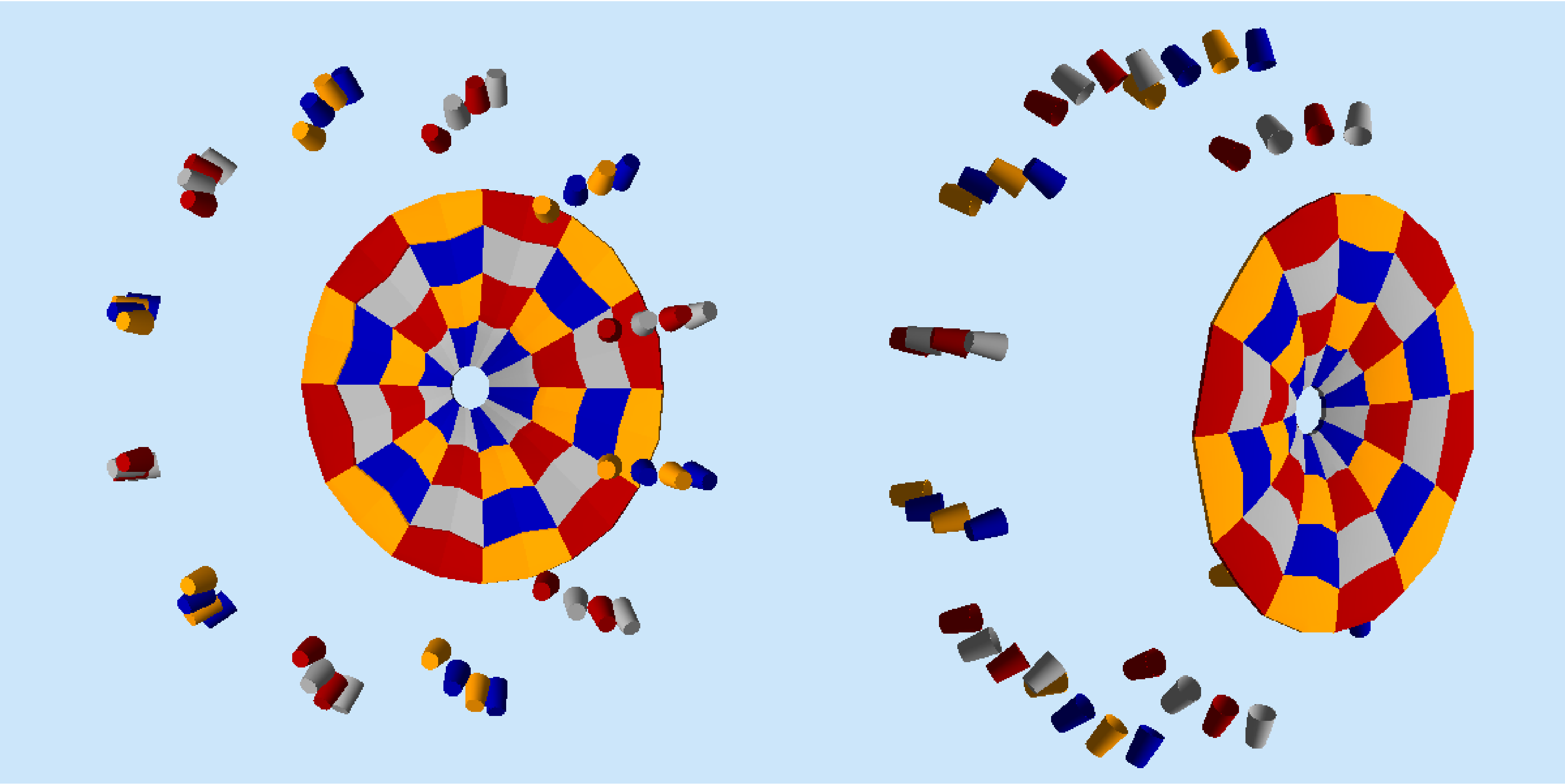
HTCC mirror geometry in gemc

Step 7: Intersection w/ cylindrical wedge (phi section)



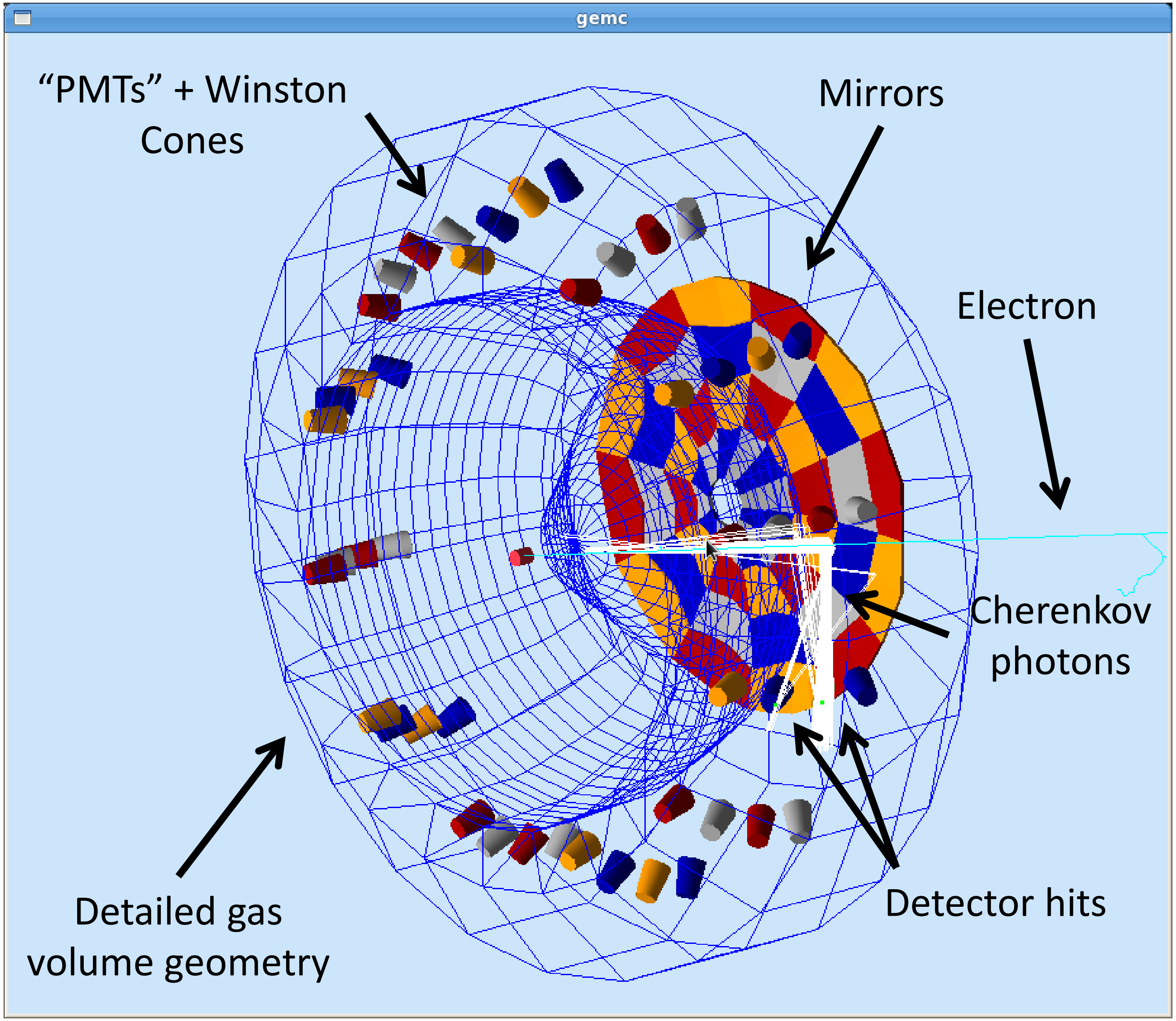
Final Mirror #1

HTCC geometry in gemc



All Mirrors shown with PMTs (only windows are included) and Winston Cones (paraboloids), gas volume not shown

An electron in HTCC



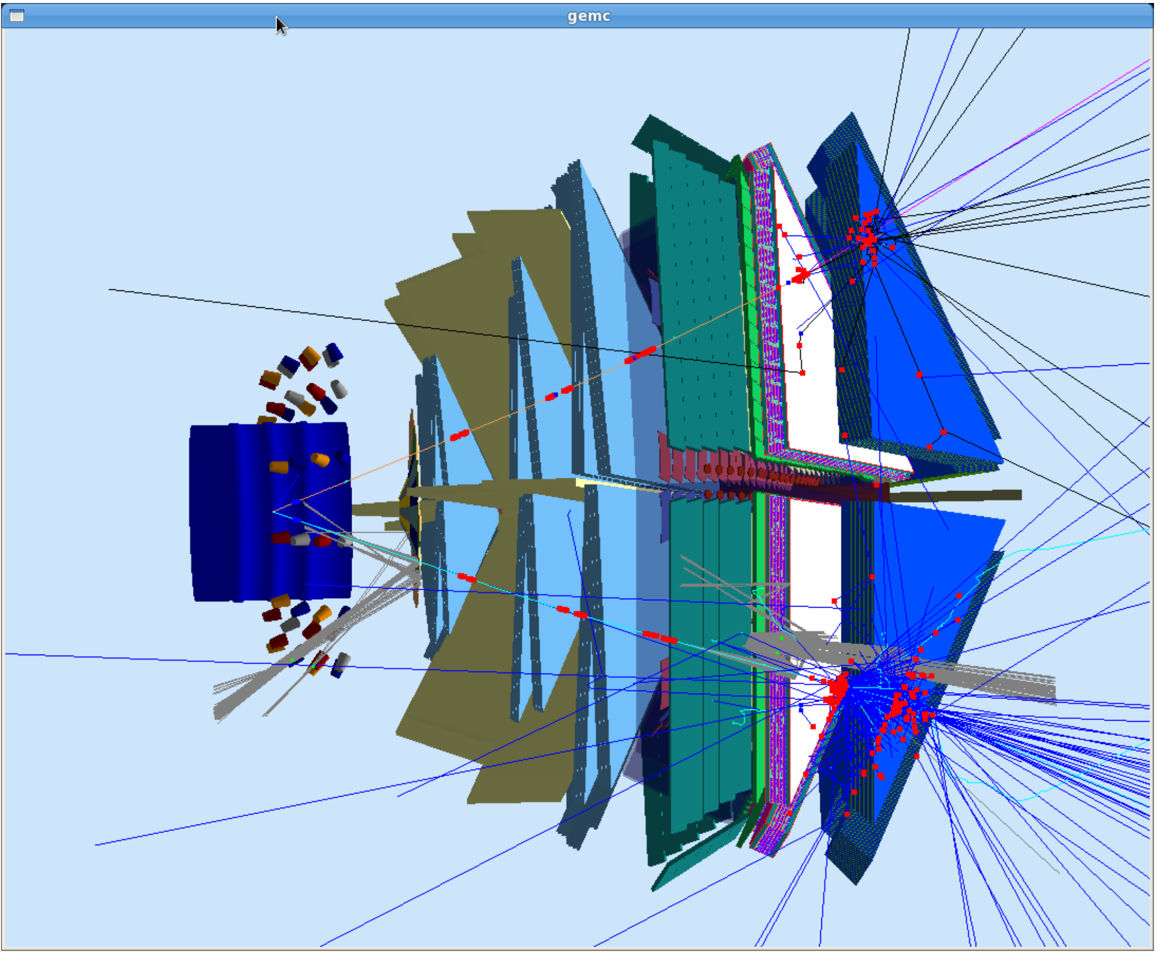
What is defined via the database

- All geometry:
  - solids, logical volumes, positioning, rotations
- Type of detector sensitivity, type of detector hit (this determines what hit process routine is performed at run-time), time window to integrate hit information, etc.
- Magnetic field
- EVIO or text output bank structure

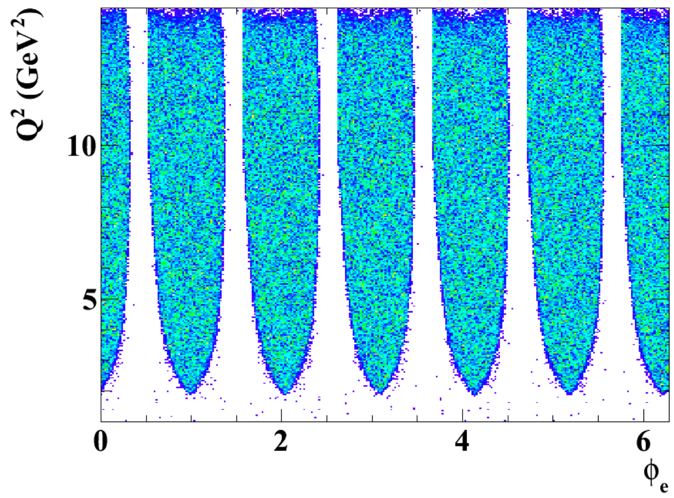
What is still hard-coded

- Presently, all material definitions/properties are still mostly hard-coded in gemc
  - Building material definitions from the database coming soon in the release of gemc 2.0
- “Hit Process”/digitization routines: algorithms for generating hit/signal information for sensitive detectors need to be coded; many general-purpose examples exist already
  - Which “hit” routine is called for any given sensitive detector is defined/chosen via the database.
  - Existing routines can be used/re-used, or new can be developed (but this requires re-compilation of the source code)
  - This is likely to remain hard-coded—very difficult to make this sufficiently general outside the source code

Full CLAS12 simulation

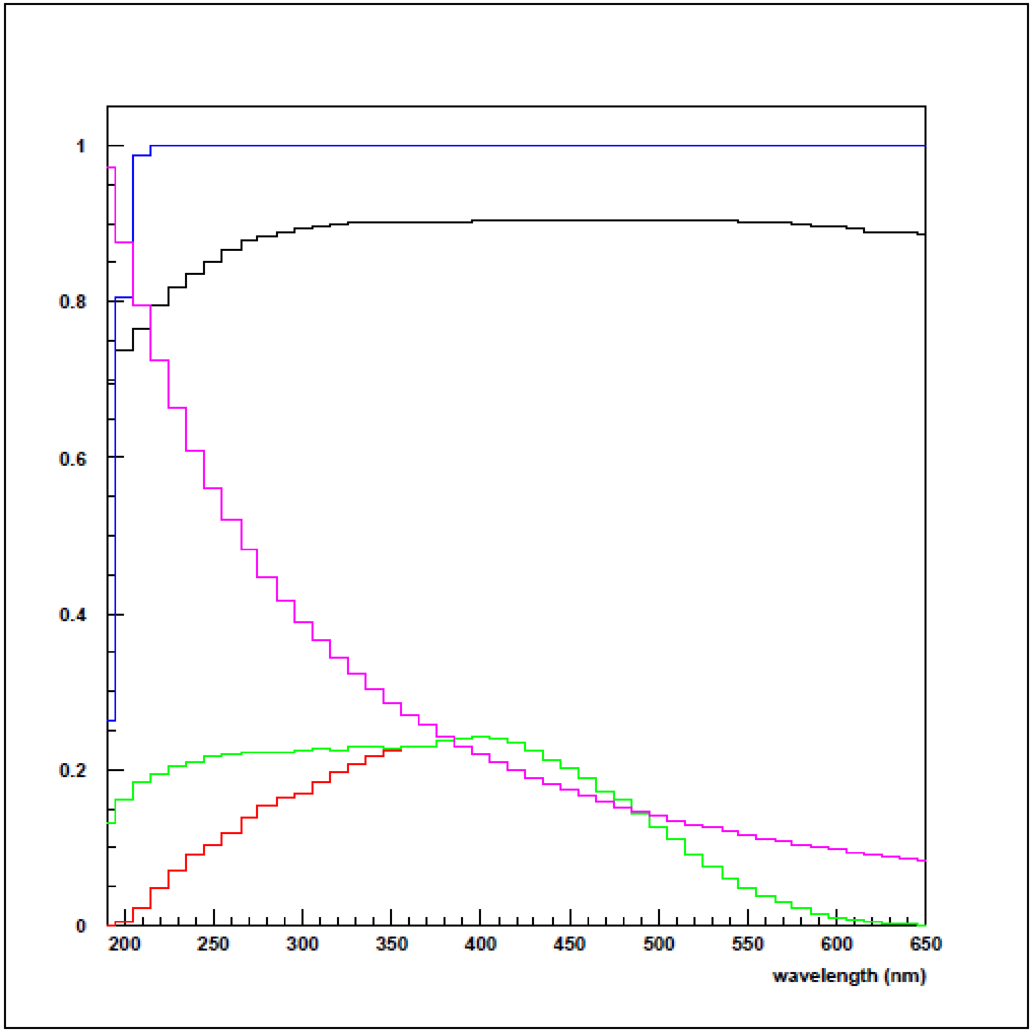


A high-Q^2 ep→ep event in CLAS12 with full detector package/magnetic field, etc.



- CLAS12 acceptance simulation from gemc for electrons; Q^2 vs. phi for ep→ep

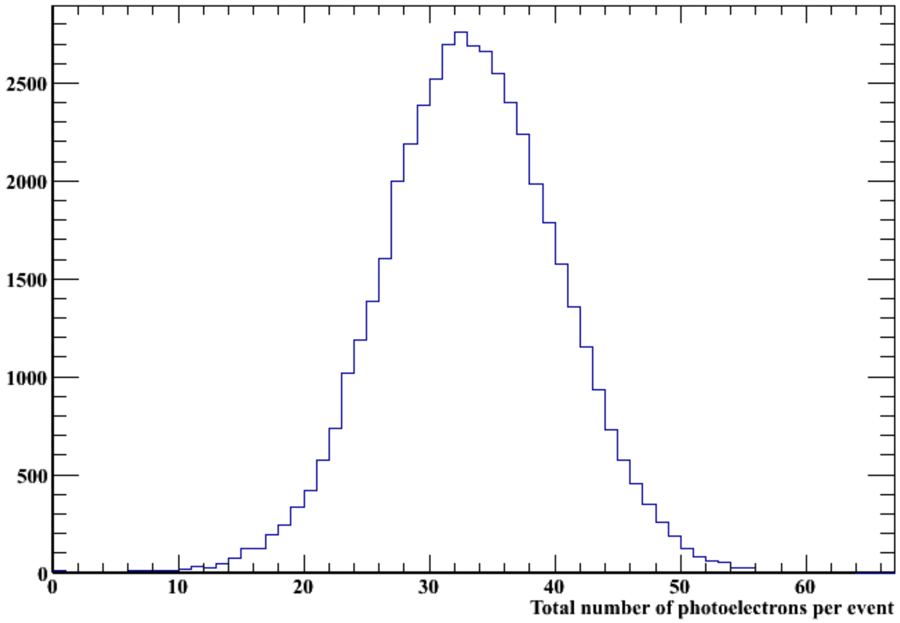
HTCC materials/optical properties in gemc

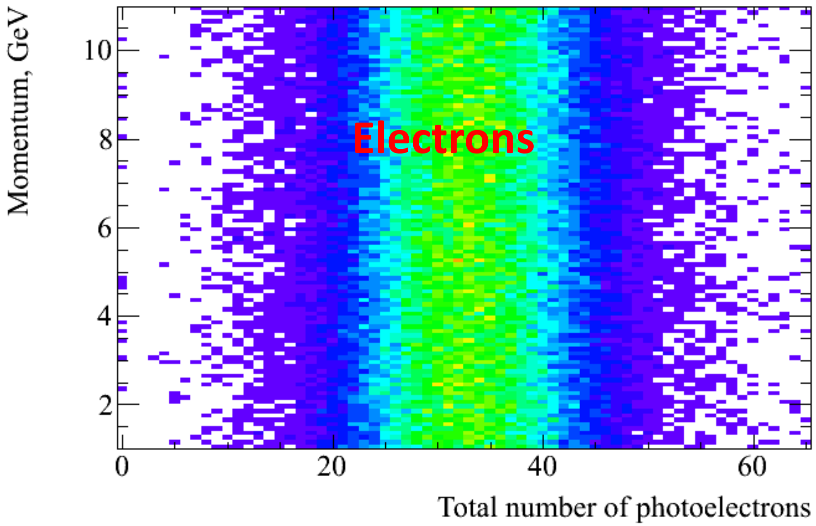


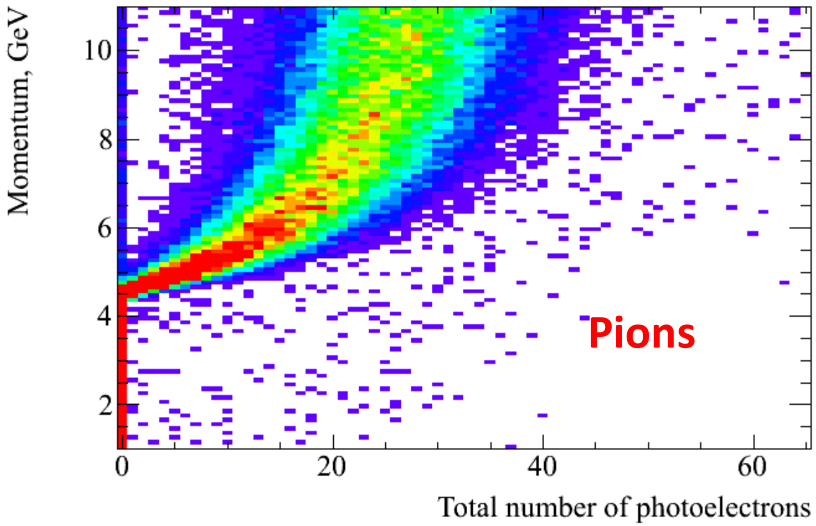
- HTCC optical properties:

1. Transparency of CO_2
2. Mirror reflectivity with $AlMgF_2$
3. Cherenkov spectrum ($dN/d\lambda \sim 1/\lambda^2$)
4. PMT Quantum efficiency with quartz window
5. PMT Quantum efficiency with UV-glass window

GEMC HTCC Simulation Results







- Top left: *Total* number of photoelectrons per event in gemc
- Right: Momentum dependence of ph.e. yield for electrons (top) and pions (bottom)
- evio output converted to ROOT trees for further analysis

Use of gemc for SBS+BB

- Advantages:
  - Easy to develop even complicated detector geometries via easy-to-learn, easy-to-use Perl/MySQL database interface—quick learning curve for students/post-docs
  - Comprehensive and general physics lists available
  - Many materials already defined, common "hit" algorithms exist
  - Don't need in-depth knowledge of C++/GEANT4 libraries (but of course it helps)—reduce overhead of coding the entire application environment
  - Avoid reinvention of wheel with multiple codes/application environments floating around with various parts of detectors
  - Long-term code management/development/support by Hall B staff
  - Multiple users/designers can contribute simultaneously, build up the geometry of all detectors/magnets/beamline in a single, central database
  - Use of common framework with Hall B and others—develop, share and benefit from expertise and experience of others; support ease of use in post-upgrade JLab computing environment
  - Very good documentation exists w/lots of tutorials/examples

Use of gemc for SBS+BB

- Disadvantages:
  - Managing different versions of detector geometry is currently not straightforward; a single database, often overwritten w/changes; hard to know what configuration was actually used when the geometry is changing often—what event generator, what parameters, etc...
  - Database structure is not necessarily conducive to "two-arm" experiments typical of Hall A; with multiple spectrometer angles/positions in a single experiment.
  - Could become problematic w/ customizable, "modular" spectrometers as in SBS/BB
- My view—these pitfalls are relatively minor, solutions are already being developed, and the benefits far outweigh the disadvantages

Summary and Conclusion

- gemc framework developed for GEANT4 simulations of CLAS12 has proven very powerful and flexible, so much so that several other large experiments/collaborations have adopted it
- This could be exploited to develop the Monte Carlo for BigBite and SuperBigBite
- Given what already exists, is it necessary?
- Monte-Carlo coordinator needed—should be long-term collaborator
  - Organize efforts of collaborators/students/postdocs in collecting geometry info
  - Interface with designers/engineers
- If we adopt gemc as simulation framework; need to think about how to interface with reconstruction code—Hall A analyzer.
  - Output is EVIO—same format as Hall A DAQ
  - Bank structure may be different
    - Possible to write gemc "digitized" output in same format as Hall A raw data format?