1. Went through reactor module tutorial and built meshes that represent full reactor cores. Played around with input files in order to figure out what options do what.
2. Worked through learning C++
3. Looked at creating a new mesh generator that would go at the end of creating the mesh in MOOSE called CSGGenerator. This didn’t work because we were only able to receive a mesh object and not the mesh generators that had previously operated on it. Pivoted to building CSG method into each MeshGenerator (current implementation). We later learned that it should be possible to retrieve a history of the mesh generators that operated on the mesh
4. Moved on to looking at how OpenMC creates the geometry. Specifically, we made different geometries in OpenMC using the python api/jupyter notebooks and looked at the format of the file called “geometry.xml”. This file serves as an input to the OpenMC executable.
   1. The format is detailed in Figure 6 of the conference poster
5. Started looking at pugixml. Hardcoded some parameters (like radius, material id…) and created the xml tree as created in “geometry.xml”. We started with a single sphere, worked up to multiple concentric spheres, and then to a pincell. Checked that the pugixml tree output matches the “geometry.xml” from the OpenMC python api. Implemented the whole pincell 3d CSG in this test project.
6. Implemented the xml tree creation into MOOSE. This adds generateCSG to the MeshGenerator class that each mesh generator object must then override. I think that the default behavior (which is supposed to make a nullptr) is broken or the call to use the default behavior (such as in the incomplete implementation in AdvancedExtruder) is broken, so any mesh generator that does not override generateCSG might throw an error.
   1. Mostly implemented the 2d pincell generation into PolygonConcentricCircle. The boundary condition and material ids are hardcoded and should be added as additional input parameters in the validParams() method. The implementation currently only supports squares – if the polygon has not 4 sides then the code might break due to a faulty call to the default behavior of generateCSG (see above)
   2. Tried to implement AdvancedExtruder but didn’t really get anywhere. This requires passing along the csg from one mesh generator object to the other and that requires some serious framework modification. I commented out the additions I made attempting to start this but that didn’t really work (such as getCSG).
   3. The biggest problem meshgenerators are the ones that modify an existing mesh. Not only does this require passing along a CSG from one to the next, but also requires modifying the xml tree