DAGMC

UK Atomic Energy Authority

- Direct Accelerated Geometry Monte Carlo (DAGMC)
- Move away from Constructed Solid Geometry (CSG) for MC simulations and directly use CAD models
- Integrated within a number of MC codes MCNP5, MCNP6, Geant4, FLUKA, Tripoli4, OpenMC and Shift
- Faceted geometry with Oriented Bounding Boxes BVH structure
  - Smaller boxes
  - Finer granularity on boxes
- Mesh based geometry of entity sets: Volumes -> Surfaces -> Curves

Dependencies

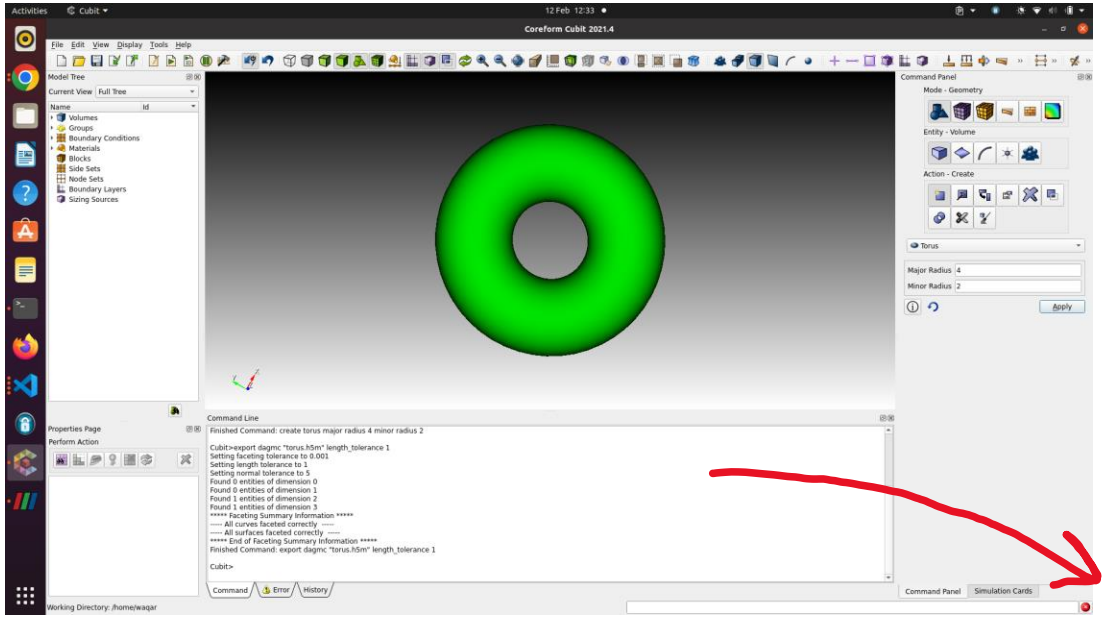
UK Atomic Energy Authority

- MOAB --> Core Geometry engine
- Embree --> Intel's ray tracer
- double-down --> double precision interface to Embree

- DAGMC provides an easy interface to use CAD Geometry (MOAB) and perform ray-geometry intersections via ray tracer (Embree)

- The use of Embree is optional (MOAB has a built in ray tracer) but I don't know why you wouldn't use it

Setting up a DAGMC instance



- Create faceted .h5m in cubit
- Read in .h5m
- Initialize OBB tree
- Initialize storage for Vols and Surfs
- Use MOAB instance for more

```
// ---------------------- Geometry Preparation ----------------------
std::unique_ptr<moab::DagMC> DAG;
DAG = std::make_unique<moab::DagMC>();
moab::Range Surfs, Vols, Facets, Facet_vertices;
DAG->load_file("torus.h5m"); // open test dag file
DAG->init_OBBTree(); // initialise OBBTree
DAG->setup_geometry(Surfs, Vols); // store surface and volume entity sets

// return moab instance for more flexible geometry query
DAG->moab_instance()->get_entities_by_type(0, moab::MBTRI, Facets);

std::cout << "----------------------------------------" << std::endl;
std::cout << "Number of Volumes = " << Vols.size() <<  std::endl;
std::cout << "Number of Surfaces = " << Surfs.size() << std::endl;
std::cout << "Number of Triangles = " << Facets.size() << std::endl;
std::cout << "----------------------------------------" << std::endl;
```

```
Cubit>export dagmc "torus.h5m" length_tolerance 1
Setting faceting tolerance to 0.001
Setting length tolerance to 1
Setting normal tolerance to 5
Found 0 entities of dimension 0
Found 0 entities of dimension 1
Found 1 entities of dimension 2
Found 1 entities of dimension 3
***** Faceting Summary Information *****
----- All curves faceted correctly -----
----- All surfaces faceted correctly -----
***** End of Faceting Summary Information *****
Finished Command: export dagmc "torus.h5m" length_tolerance 1
```

```
Using the DOUBLE-DOWN interface to Embree.
Loading file inres1_shad.h5m
Initializing the GeomQueryTool...
Using faceting tolerance: 0.001
Building acceleration data structures...
----------------------------------------
Number of Volumes = 3
Number of Surfaces = 54
Number of Triangles = 75610
----------------------------------------
```

[Link to MOAB Interface docs]

Basic ray trace

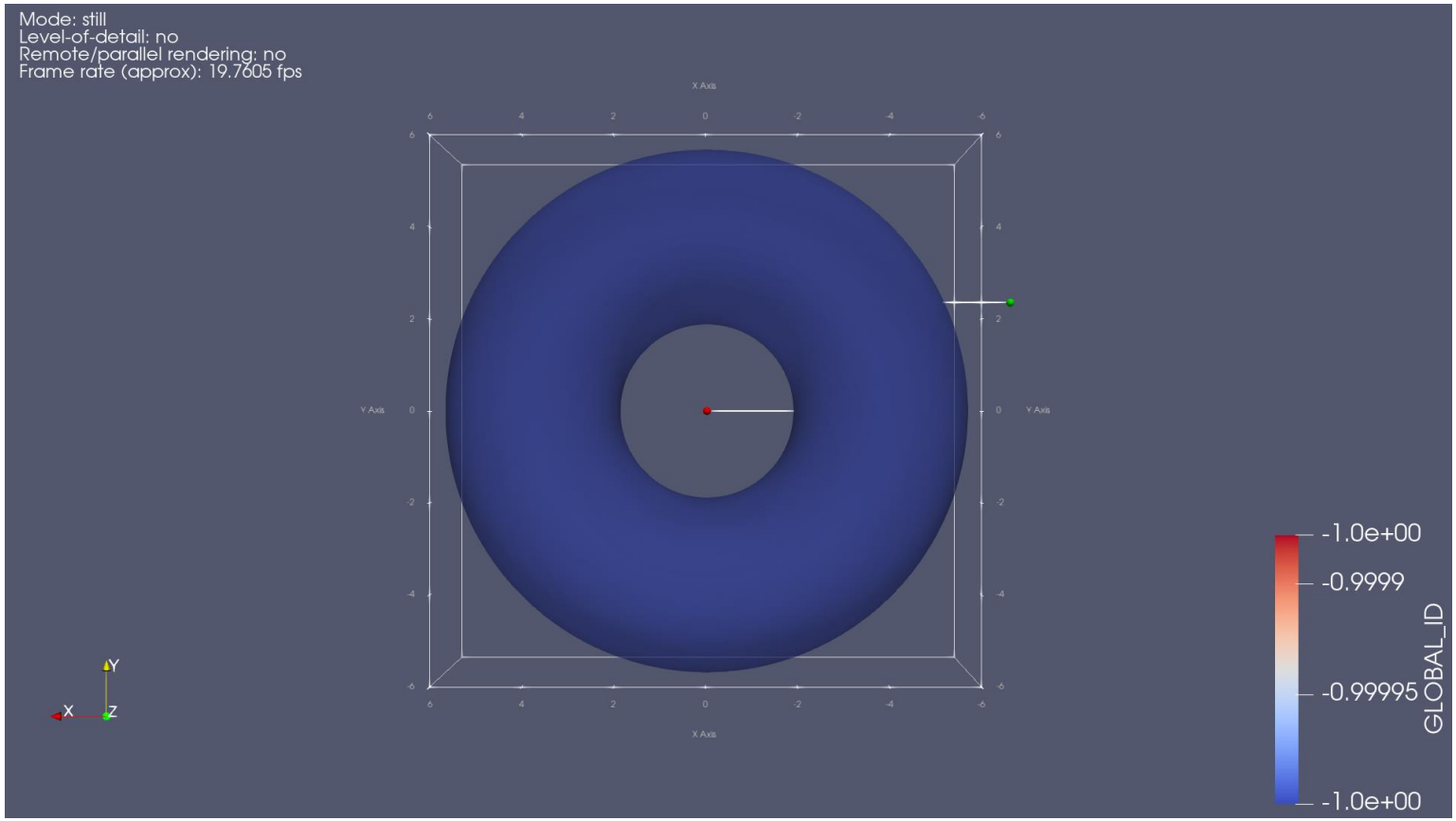
UK Atomic Energy Authority

```
moab::EntityHandle VolID = DAG->entity_by_index(3,1); // volume to fire ray at
double origin[3] = {0,0,0}; // origin of ray
double direction[3] = {-1,0,0}; // direction of ray
moab::EntityHandle nextSurfaceHandle; // EntityHandle of surface hit
moab::EntityHandle facetIntersected; // EntityHandle of triangular facet hit
moab::DagMC::RayHistory rayHistory;
double nextSurfaceDistance; // distance travelled before surface hit
double rayDistanceLimit = 10.0; // optional distance limit of of ray
int rayOrientation = -1; // If provided determines intersections along the normal provided,
                         // e.g. if -1 allows intersections back along the the ray direction,
                         // Default is 1, i.e. exit intersections

std::ofstream rayPoints("ray_pts.txt");
double ray_pt[3];
double finalPosition[3];

DAG->ray_fire(VolID, origin, direction, nextSurfaceHandle, nextSurfaceDistance, &rayHistory, rayDistanceLimit, r

// ---------------------- Second ray trace ----------------------
if (nextSurfaceHandle != 0) // if surface hit
{
  rayHistory.get_last_intersection(facetIntersected);
  std::cout << "Surface hit! EntityHandle of surface - " << nextSurfaceHandle << std::endl;
  std::cout << "EntityHandle of Facet hit - " << facetIntersected << std::endl;
  std::cout << "----------------------------------------" << std::endl;
```

```
----------------------------------------
Surface hit! EntityHandle of surface - 12682136550675316739
EntityHandle of Facet hit - 2305843009213693953
----------------------------------------
Surface hit! EntityHandle of surface - 12682136550675316739
EntityHandle of Facet hit - 2305843009213727384
----------------------------------------
```

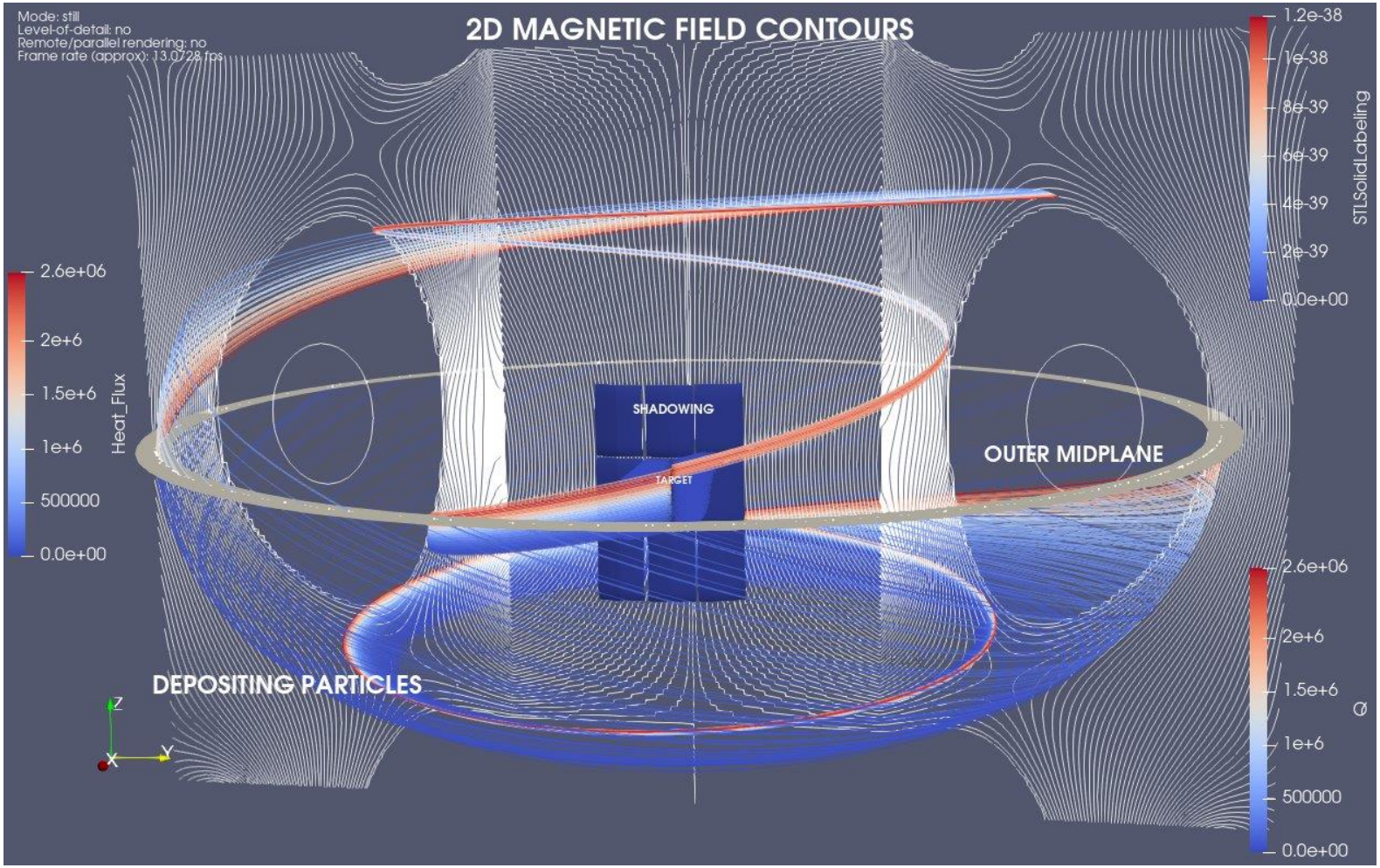
- Get volume of interest
- Set ray origin/direction
- Fire ray
- Return EntityHandles of surface and/or facet

Basic ray trace



- Reduced opacity of torus – Can see that rays terminate when they intersect surface
- Two ray traces (origins red and green points)

A more interesting use case



Useful Links

UK Atomic Energy Authority

- DAGMC Install Instructions and DAGMC developer theory guide
- Coreform Cubit Meshing software and preparing geometry for DAGMC
- MOAB repository and MOAB docs
- Double-down repository and double-down docs
- Some further slides about DAGMC by Andy
- MOAB interface methods (from MOAB docs)

- Docker image available with dependencies pre-built here for quick start/play around:
  - https://github.com/Waqar-ukaea/dagmc-basics