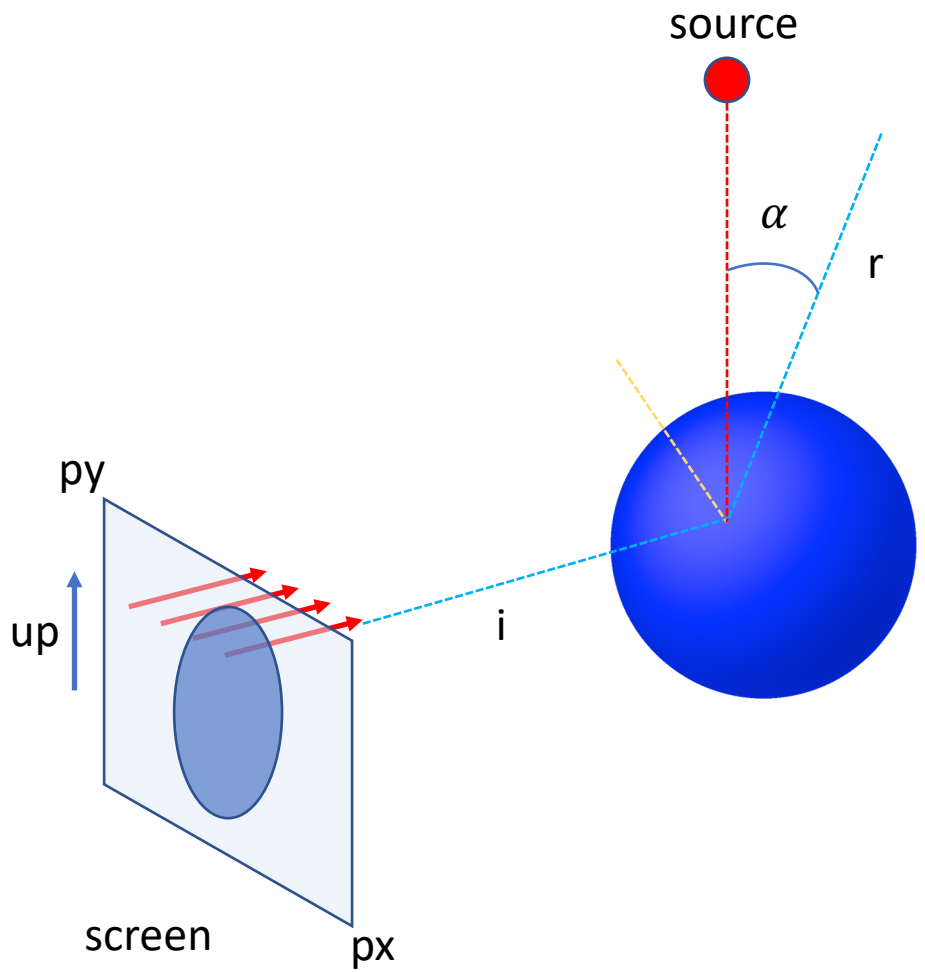
RT prototype preliminary

The model

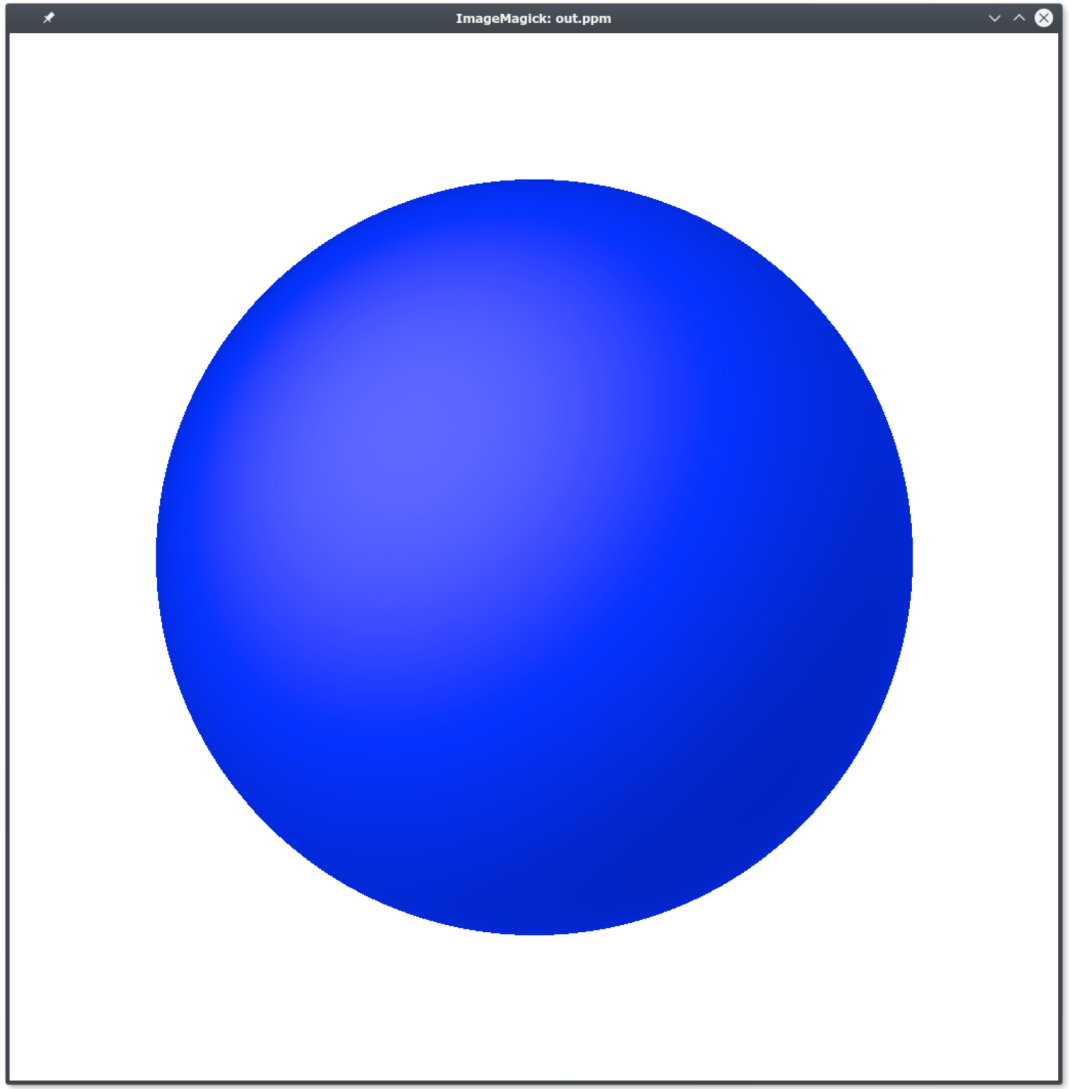
- Input: screen position/size, up vector, object/light colors, RT model (specular/transparent)
- Shoot rays from each “pixel”
- Traverse geometry model & compute pixel color depending on the model
  - Specular: stop ray when reaching a given depth
  - Transparent: traverse full geometry
  - To do: reflection + refraction: allows generating new rays



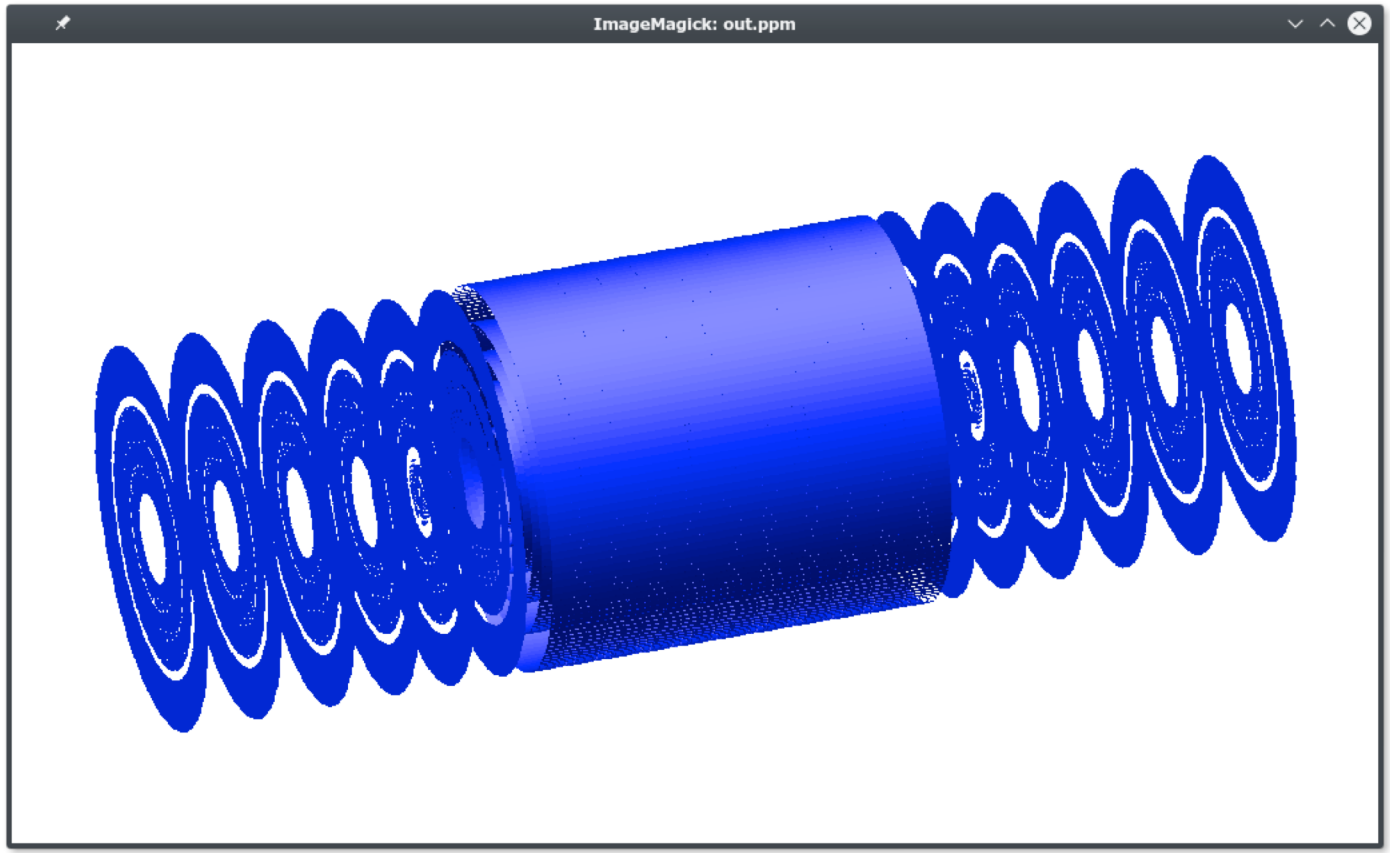
The prototype

- Reading geometry from gdml file
- Computing scene parameters to fit the view window
- Pre-allocate px*py *Ray* objects
  - Position, direction, current/next navigation states, number of crossed boundaries, “done” flag, final pixel color
- Propagation kernels
  - One step per ray: invoking a shader with all RT algorithms, deciding if the ray continues or not
  - Block stepping: one step for all rays for which *done = false*
  - Full propagation: repeat block stepping until all rays are done
- Image saved as 24-bit color map as text format (.ppm)

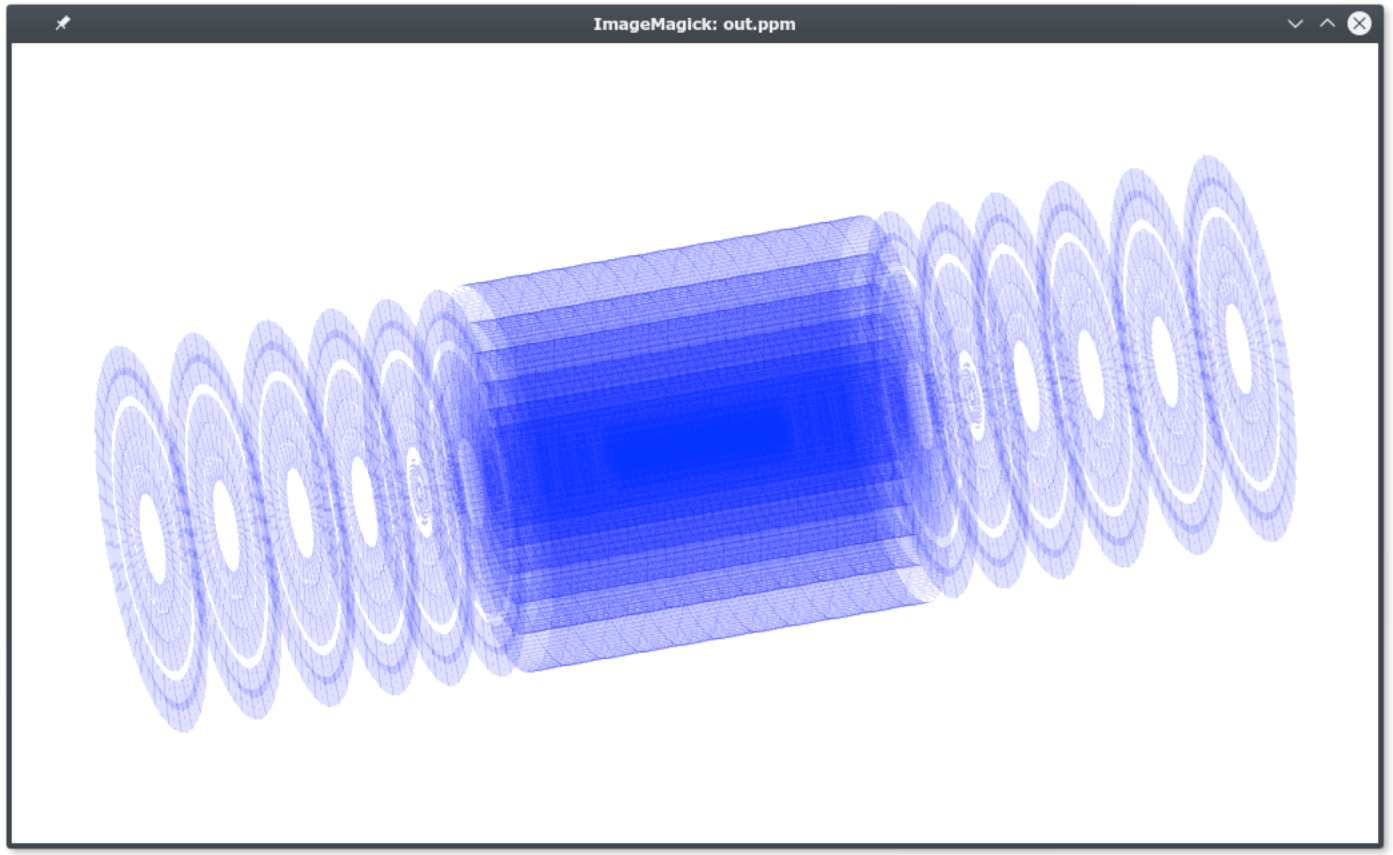
Simple shapes



TrackML geometry, specular view



TrackML geometry, transparent view



Next steps

- Moving the model to GPU
  - Instrumenting the Raytracer class with host/device macros -> cuda::Raytracer
  - Building geometry on host, moving on device
  - Initializing rays on device directly
- Handling parallelism on CPU
  - Probably using OMP
- Writing the kernel for GPU (reentrant)
- Evolving the RT model to handle reflection + refraction (ray generation)
- Investigate conversion using Alpaka on the RT code