3D Rendering Engine

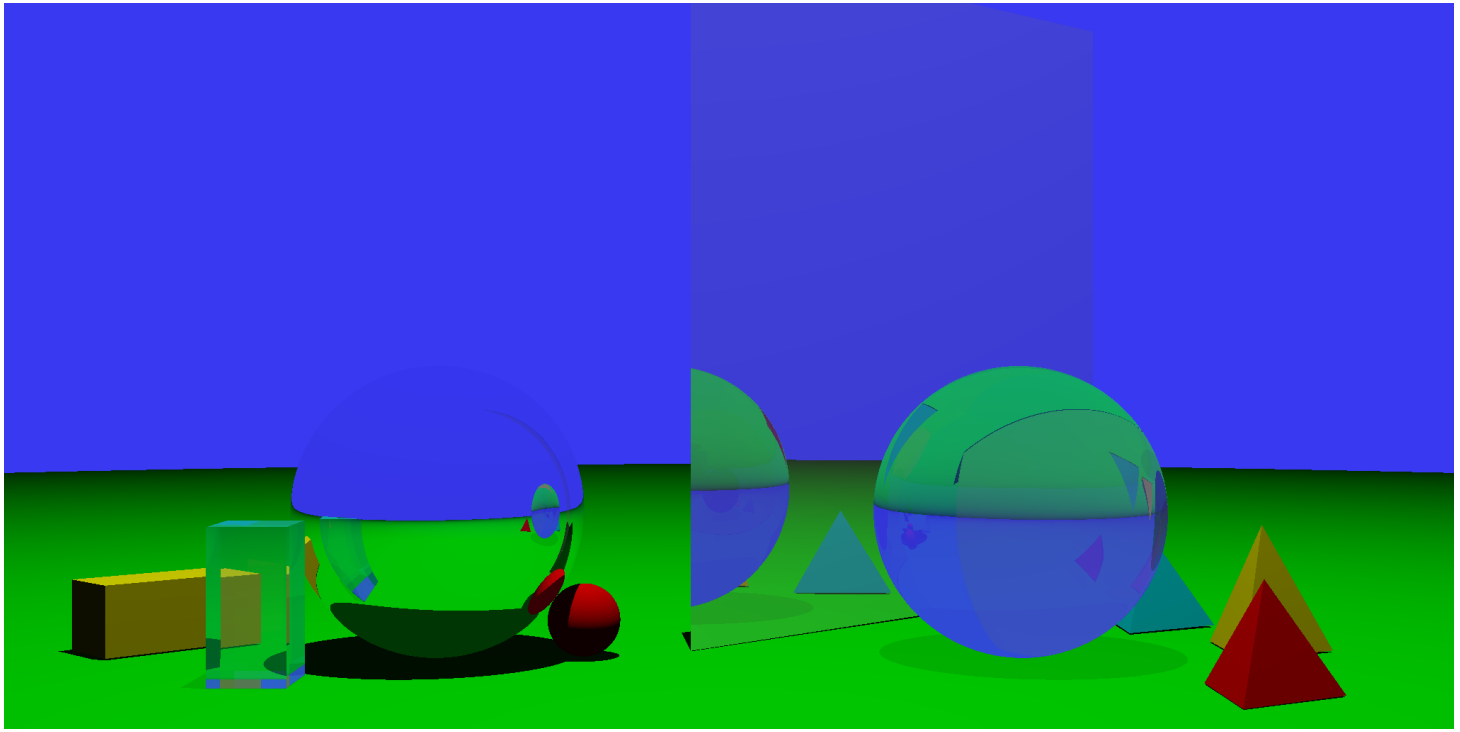
Venu Nathan ME17B081

Problem Statement :

Given a scene containing a number of primitive 3D models (spheres, cuboids, tetrahedrons), with surfaces of different types (diffuse, dielectric), and light sources, generate an image approximating the image captured by a perspective projection camera placed in the scene.

Results :

An image approximating the image captured by a perspective projection camera placed at the scene, can be produced for any scene containing primitive 3D models with surfaces described above.



Too many registers being used :
Due the highly complex functions being executed on the GPU, the corresponding kernel can’t be launched with 1024 threads per thread block as SM’s in most GPUs don’t have enough registers.
Hence the number of threads per thread block had to be restricted to 64, which slightly reduces the performance.

Stack Overflow when determining pixel colours :
The colour of dielectric surfaces is determined by recursively generating and finding intersections of reflected and transmitted rays. Due to heavy computation involved in finding intersection of these rays with the objects in the scene, the size of Call Stack per thread had to be manually increased from 1KB to 8KB to allow a recursive depth upto 8 reflections/transmissions.

Inconvenience of having different primitive models :
Each of the primitive models (spheres, tetrahedrons, cuboids, planes), are objects of a different class. If each of the objects in the scene are stored in seperate arrays, one for each type, then it becomes confusing when adding support for new primitive models, as it changes the definition of the class which stores the objects, and functions which operate on them.
To handle this, objects of all these different types are stored together in a Heterogeneous Container.