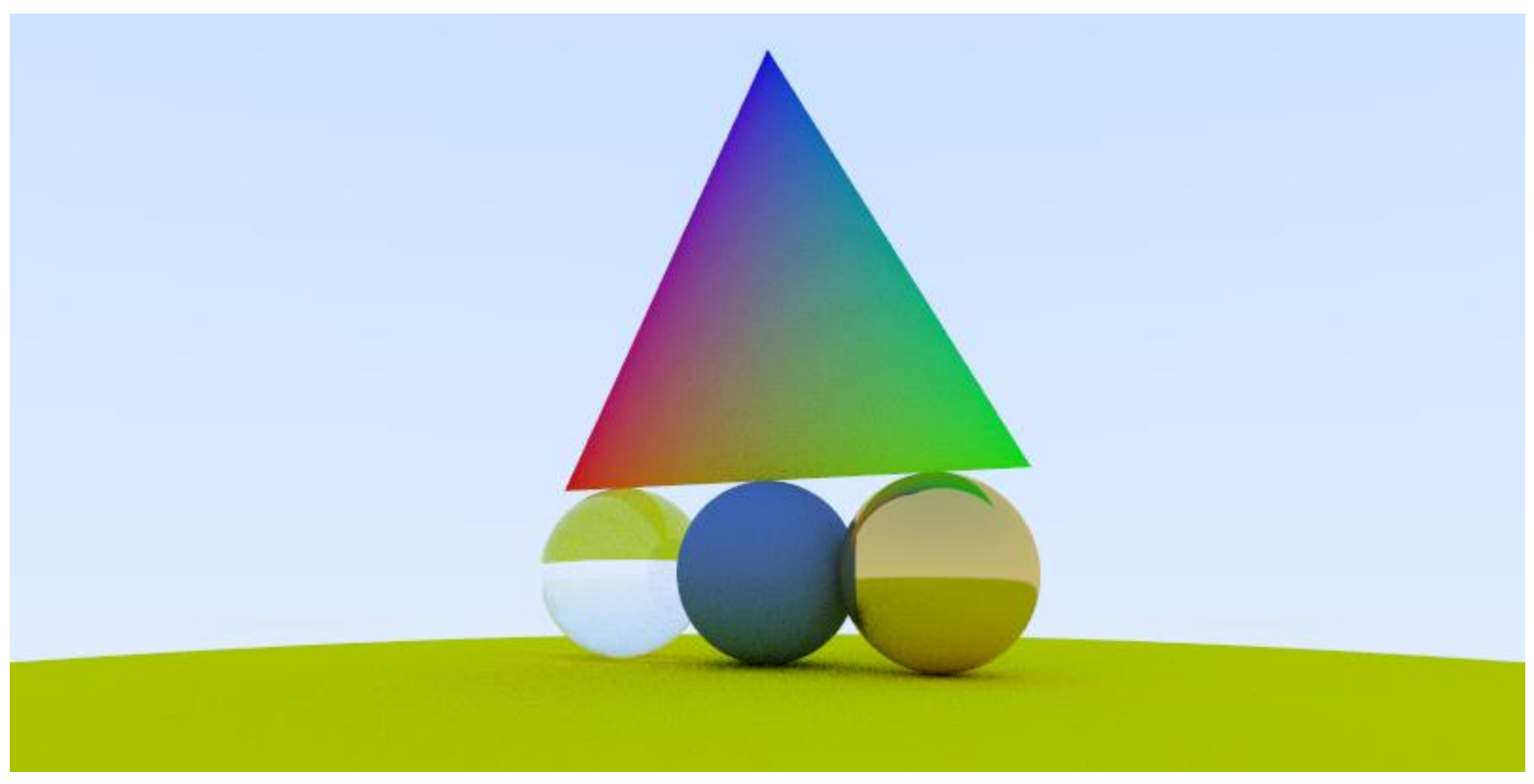
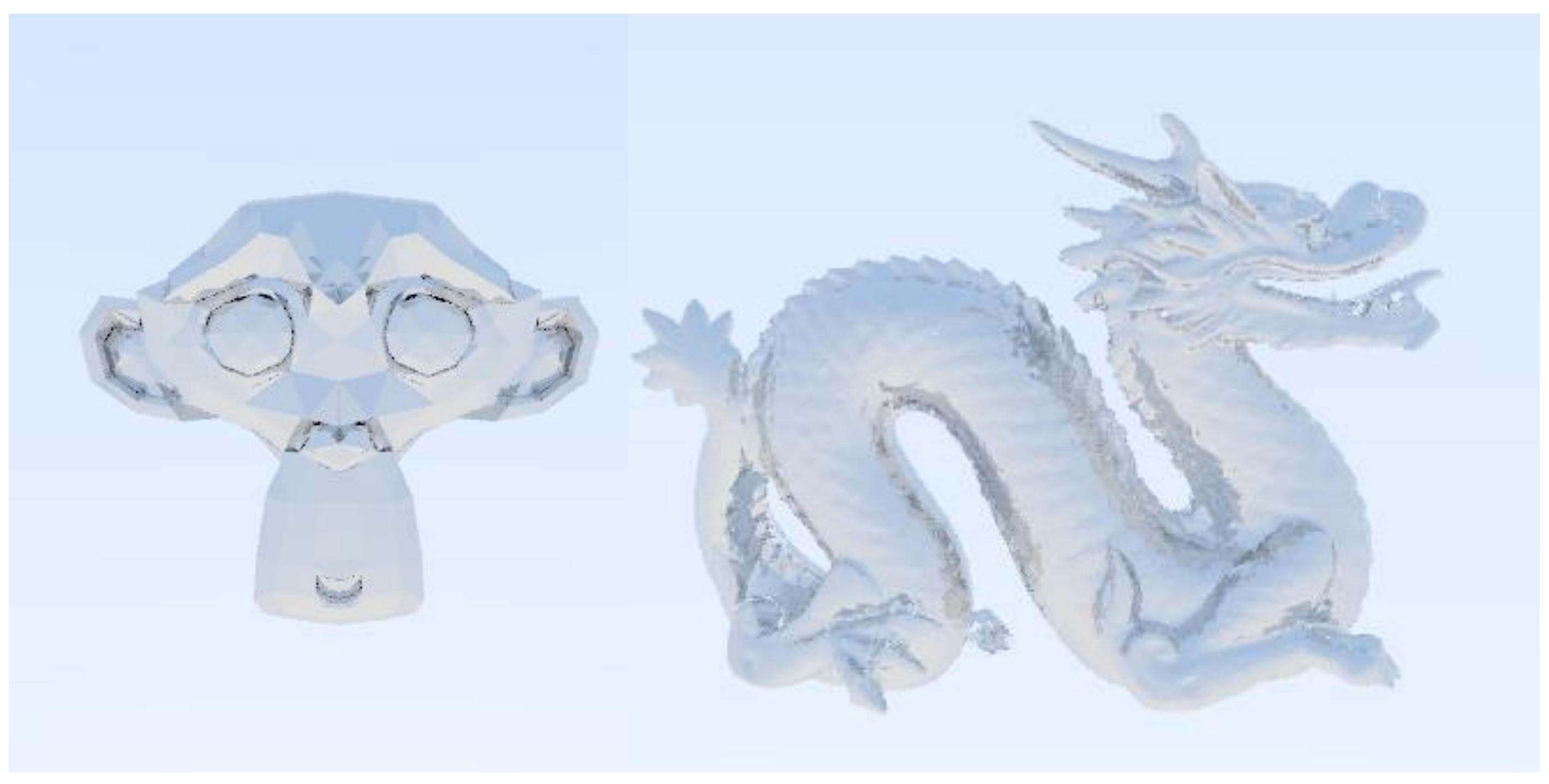
Physically Based Rendering: Ray Tracing

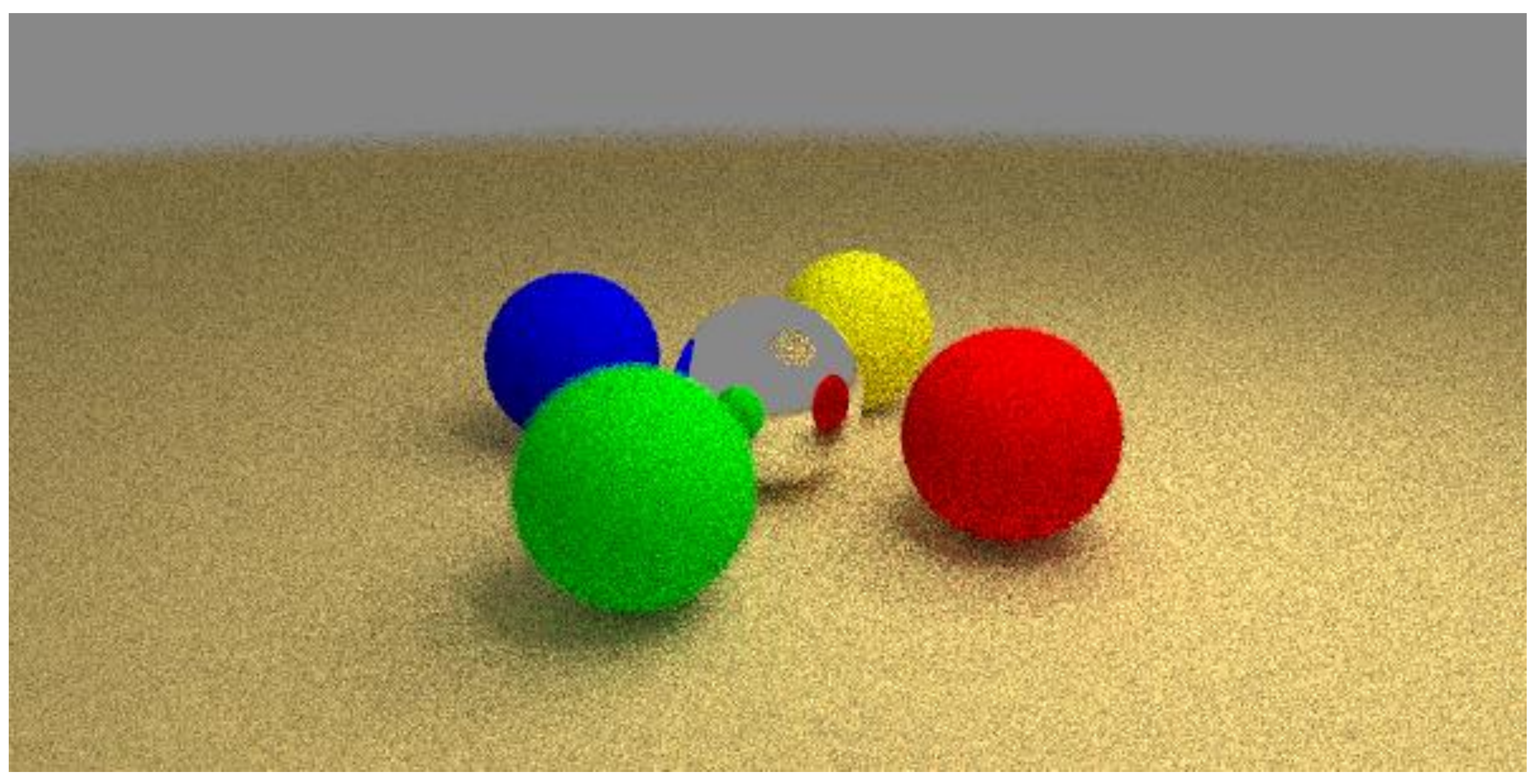
Raymond Liu
Computer Graphics - Oregon State University

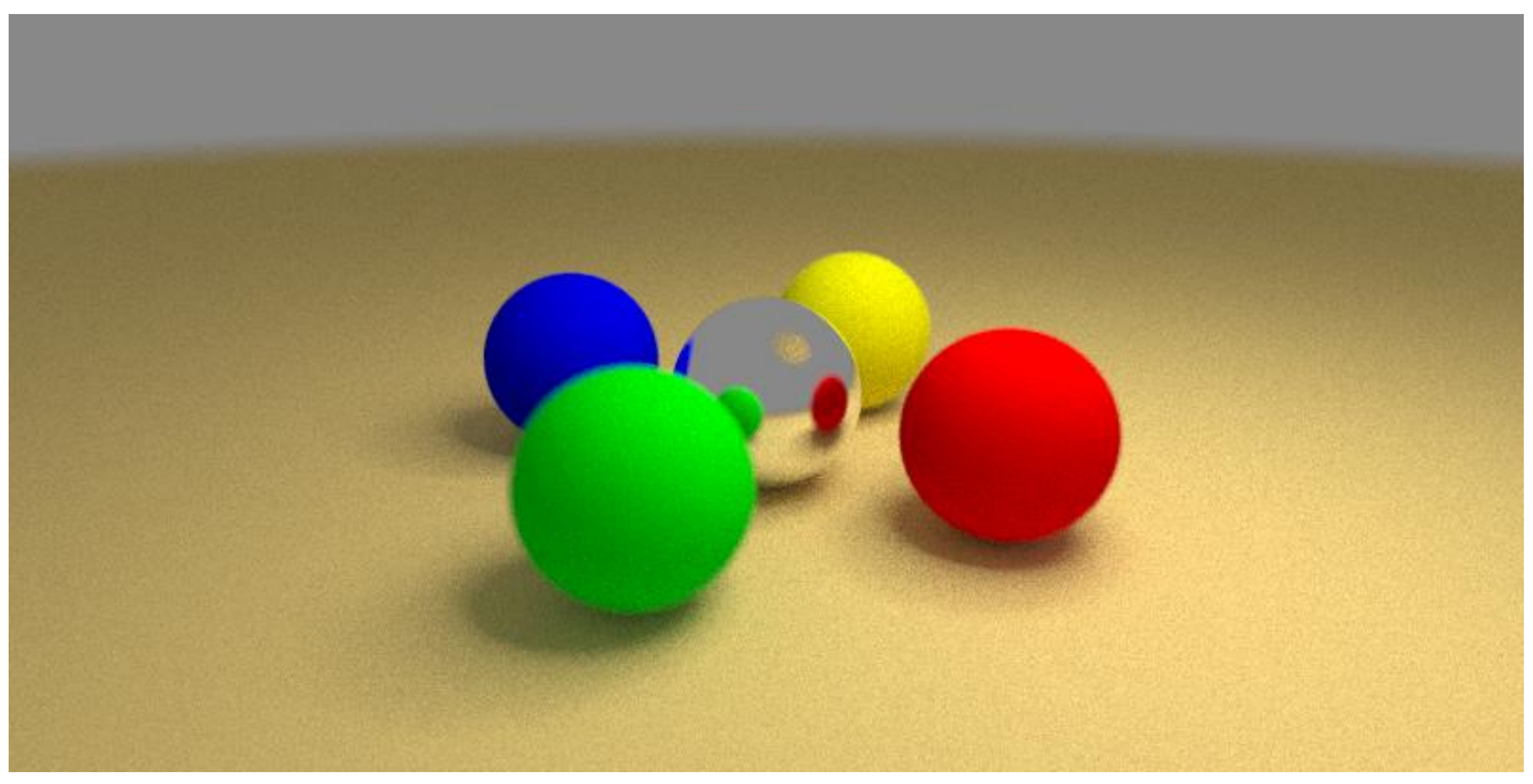
Methodology

- Ray tracing is guided by the concept that all the objects we see are illuminated by light, which comes from various light sources. Each beam of light that we see goes from a light source, to various objects. We trace light rays backwards from our eyes to various objects to the light's source. At each intersection between a ray and an object, we simulate the interaction.
- A bounding volume hierarchy and Octree was used to accelerate the rendering process.
- Monte Carlo path tracing and bidirectional path tracing used to approximate the render equation.


Example materials (left to right): glass, diffuse, metal. Light interacts with each of these materials in different ways. We can place objects (and the camera) wherever we want.


If we put a bunch of small triangles together we can make meshes such as a monkey head or a dragon.


We can solve the rendering equation using Monte Carlo integration. This allows us to calculate how much light is emitted from any ray-object intersection point. Render time: 86 seconds.


Bidirectional path tracing uses ray paths that start from the eye as well as paths that start from the light. This takes longer but produces higher quality images. Render time: 184 seconds.

$$L_o(x,w) = L_e(x,w) + \int_{\Omega} f_r(x,w',w)L_i(x,w')(-w'\cdot n)dw'$$

The rendering equation (Kajiya1986): Solving this allows us to figure out how much light an object reflects given the incoming light and the object's emittance.

Challenges

- Compared to rasterization, which is a computer graphics technique commonly used in video games, ray tracing is exceptionally slow – things like real-time ray tracing are still considered “the future”. Thus, it is important to figure out how to accelerate the ray tracing process.
- The process behind how lighting is simulated can be confusing at times. For example, I found the application of Monte Carlo Integration to solve the rendering equation to be complicated.

Results

- The finished ray tracer is capable of loading objects and meshes into a scene, directing a camera, simulating interactions between light and various materials, and accelerating the rendering speed, also Monte Carlo path tracing and bidirectional path tracing.

GitHub: https://github.com/liura/ray-tracing

References

- Physically Based Rendering: From Theory to Implementation. Matt Pharr, Wenzel Jakob, Greg Humphreys.
- Realistic Ray Tracing – 2nd Edition. Peter Shirley, R. Keith Morley.

Acknowledgments

- Prof. Eugene Zhang, Prof. Yue Zhang, Jinta Zheng, Fariba Khan and ASE Saturday Academy.