Ray Tracing through Viewing Portals

Chris Young
Igor Stolarsky
April 23, 2008

Introduction

This paper presents a method for ray tracing scenes containing “viewing portals” – circular planes that act as windows between two disassociated view points in 3D space. In this regard, we implement them using recursive ray tracing and the casting of shadow rays through portals.

Related Works

Inspiration for this paper was drawn from the Valve Corporation video game *Portal* which implemented a similar idea with a non-ray traced, camera based method.

This paper also employs and builds upon traditional distributed ray tracing methods as described by Cook et al (1984) [1].

Motivation

We wanted to extend the ray tracing method to allow for the inclusion of portals, which would allow scenes that simultaneously depict multiple viewpoints within the scene. In doing this, we endeavored to define how these portals would intuitively work and achieve a faithful representation of that in the rendering. This would enable new possibilities in ray traced scene creation that would that otherwise be unavailable. For example, it would allow for an object to be shown from multiple angles while still retaining spatial relationships, an effect that would not be achievable simply with mirrors. This could then be applied to architectural or artistic modeling, in which such spatial relations could be important.

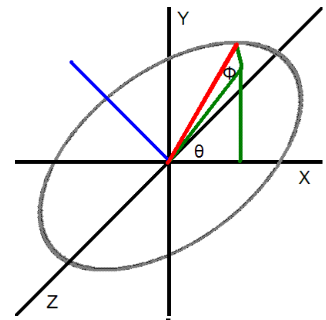
Technique

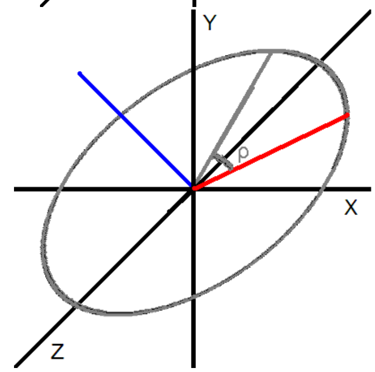
Portal Definition

In our implementation, a portal is a two dimensional circle in space and is linked with another portal. Each portal has a front face that it can be viewed from and a baseline, which define the “up” direction for image seen through it. When looking through a portal, the scene in the portal should appear as though one were looking out of the front face of the linked portal with the view aligned with the “up” direction of the baseline. Portals can thus be seen as a window split into two disassociated panes which each contain a scene from the point of view of the other as though they were still connected.

Representation

A portal is represented in 3D space through an extension of spherical coordinates. As such, it is modeled by six parameters: 3D center position, theta, phi, rho, radius, and a reference to the portal linked to this one, all drawn from a modified .obj format.



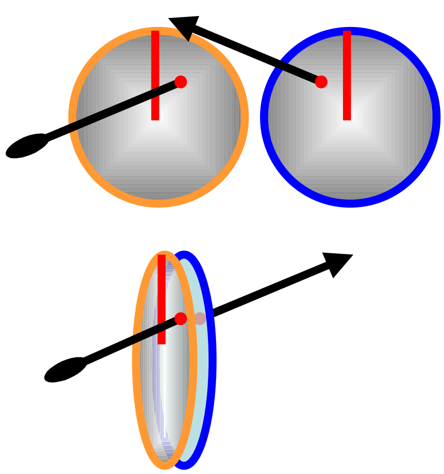


By default, a portal is a circle flat on the x-y plane with a normal along the z-axis and a baseline along the x-axis. The baseline provides an angular reference point for points on the portal. The portal can then be defined by the area swept out by the rotation of the baseline about the normal.

A portal is then oriented by taking the standard definition provided above and rotating the normal and baseline phi radians in x-z plane and theta radians in the x-y plane, in that order. Thus, the swept area of the baseline and normal can be made to face any direction. However, the position of the baseline relative to the normal cannot be set through these rotations alone. For this, the internal angular offset of rho (a rotation about the normal) is used to align the baseline to the proper orientation within the portal.

Ray Tracing

Ray-portal intersection is computed by solving the implicit portal equation and explicit ray equation. To do this, first apply transformations to align the portal in the x-y plane, by rotating the portal according to its parameters. Then apply the same transformation to the ray. Finally, solve the implicit portal (circle) equation and explicit ray equation for the point of intersection relative to the portal in radial coordinates – the angle from the baseline and distance from the center.



Once the relative point of intersection has been found on the portal hit by the ray, the respective point on the link portal is found by applying the relative coordinates of the intersection to the global parameters of the linked portal. Note that because the portals are connected “back to back”, this means that the relative position and direction of the ray in the linked portal will be reversed across the plane of the baseline and the normal.

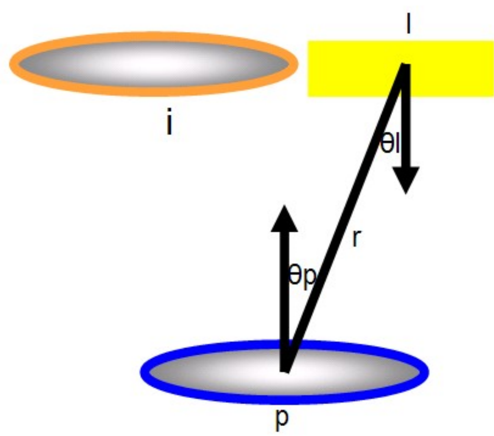
A new ray is then created at that point with the direction of the original, intersection similarly transformed in the same manner as the intersection location and traced through the scene. The answer is then recursively passed back to the original ray.

Lighting

Intuitively, light coming through a portal would result in a cone of light being emitted through the linked portal, like light coming into a room through a window. To attempt to create this effect, we applied several methods of lighting through portals.

The first method was a very naïve treatment of each portal as an area light source. To determine the amount of light emitted by a portal, the amount of light reaching its linked portal from light sources in the scene was calculated by the following equation:

$$i_{emit} = \sum_{l} l_{emit} * \frac{\cos\theta_l \cos\theta_p}{r^2} V_{lp} * \frac{A_i}{A_p}$$



Note that this is to some extent derived from the radiosity equation for computing form-factors. Some liberty was taken, however, in order to achieve acceptable results.

Each point would then, in addition to shooting shadow rays towards all light sources in the scene, shoot shadow rays towards all portals in the scene and process the shadow in the same way. The results of this method, described in the following section, were unsatisfactory, as they did not capture the desired conic property – any point to which the portal was visible was lit.

The second method was to recursively cast shadow ray towards the center of all portals in the scene to determine if they hit a light source. The shadow rays were transformed in the same method as visible rays going through portals and recursively reported any emitted light that they struck. This nearly captures the conic effect predicted above, but will only create lit areas in the shape of the area light. To achieve a smoother, more accurate visual at the cost of time, stochastic sampling was applied, shooting multiple rays randomly towards each portal. The stochastic method resulted in marked improvement over the other methods, as delineated below.

Results

To demonstrate our implementation, a number of scenes were created to show some of the different possibilities with this technique. All scenes used two reflective spheres, the smaller of which was tinted red, in a large white plane with a blue background as subject matter.

Basic

A portal facing the viewer was placed behind the sphere and a second portal, also facing the viewer was placed above the spheres. The spheres can be seen in the second portal as visible from the first.

Infinite Hallway

The two portals are placed facing each other with the spheres in between, creating an “infinite hallway” effect similar to standing in between two mirrors.

Infinite Spiral

Similar to Infinite Hallway, but one of the portals’ baselines is offset slightly, leading each successive image to be rotated by that offset from the last.

Aerial

A portal is placed above the spheres, facing down and another, larger portal is placed in the air facing the viewer. The spheres appear to be larger in the portal facing the viewer. The orientation of the portals allows for complex ray paths to be traced.

Light (with Area Light)

One portal is placed in the air, directly beneath and facing a light source and the other is placed on the ground near the spheres. When traced with the naïve approach to lighting, light is even across the entire visible range of the portal. Tracing a single shadow through portals results in a lit area in the shape of the scene’s light source because the only pixels getting light are the ones positioned to hit the light in a one to one ratio. Stochastic sampling yields the best looking result, giving a gradient of light that reflects more accurately which points would actually be lit through the portal, but needs a very high sampling rate in order to look good.

Light (without Area Light)

The same as Light (with Area Light) but the area light source blocked from reaching the scene, leaving the land bound portal as the only source of light. This accentuates the individual characteristics of each lighting model. The naïve method is completely non-directional, a single shadow ray isn’t very smooth, and stochastic sampling yields the best, though most expensive result.

Conclusion

While relatively undeveloped, this technique shows much promise. Portals were rendered to pleasing effect and basic lighting was implemented without tremendous effort in a relatively short period of time and without excessively high computation time. Slight short comings were found in the lighting implementation in the sense that a large quantity of shadow rays must be employed to yield a good result, greatly slowing down the rendering.

The method described in this paper leaves a lot of possibility for extension. Some possibilities for future work include: adding portal attributes such as clouding, scattering of intersecting rays, transparency, etc; developing a more effective lighting model, one that better represents light coming from the source through a portal such as photon mapping; adding interaction between portals and objects; radiosity through portals for ambient light.

Work Breakdown

Most of the project was a combined effort with both contributors working simultaneously. The following is the breakdown of individual tasks.

Igor

- .obj Modification and data loading
- OpenGL portal representation
- Lighting via shadow rays through portals

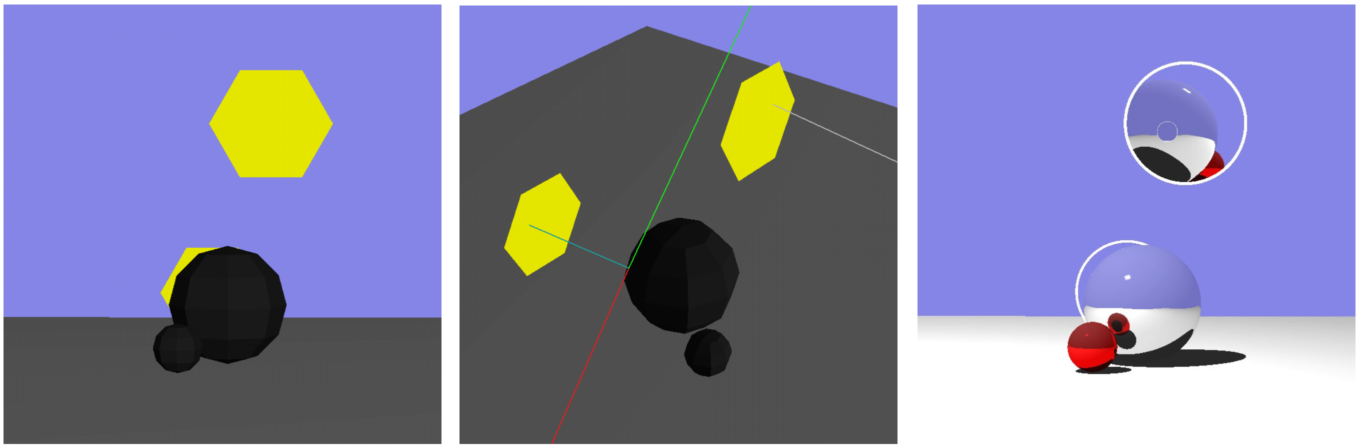
Chris

- Portal object
- Ray tracing
- Naïve lighting

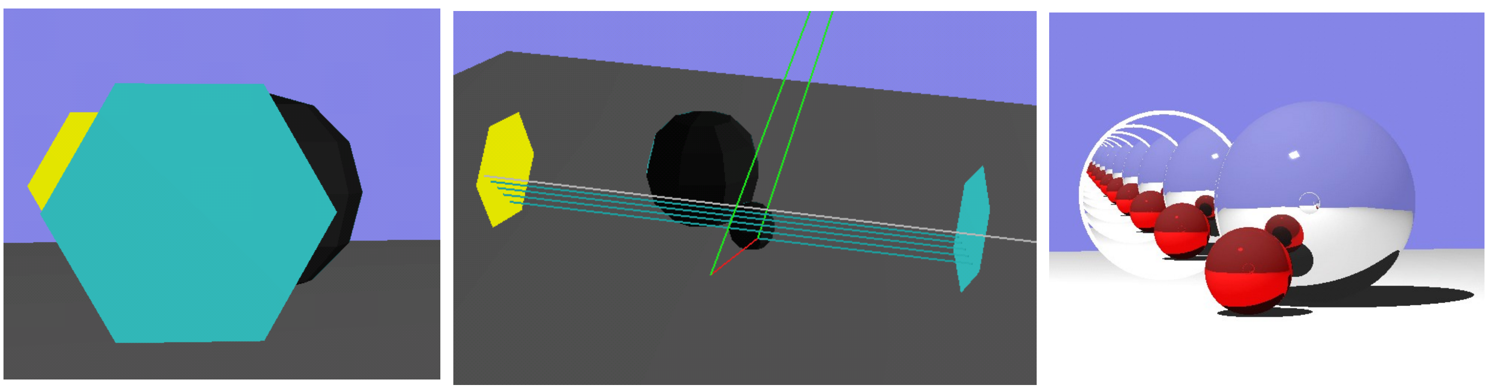
References

[1] Cook R., Porter T., Carpenter L. "Distributed Ray Tracing" July, 1984. Computer Graphics (Proceeding of SIGGRAPH 84) 18, 3, 137-144.

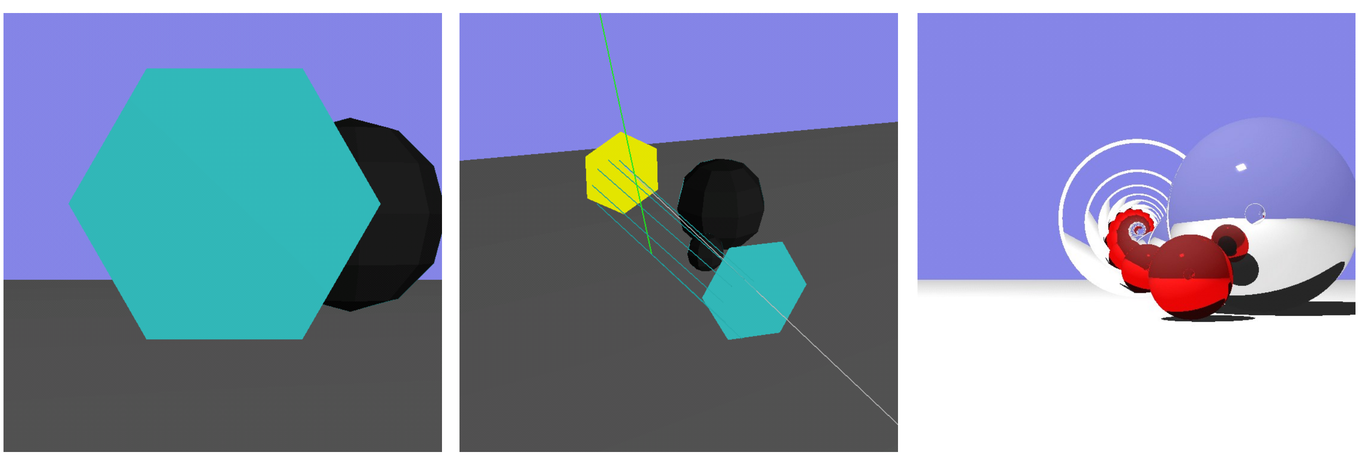
Screenshots



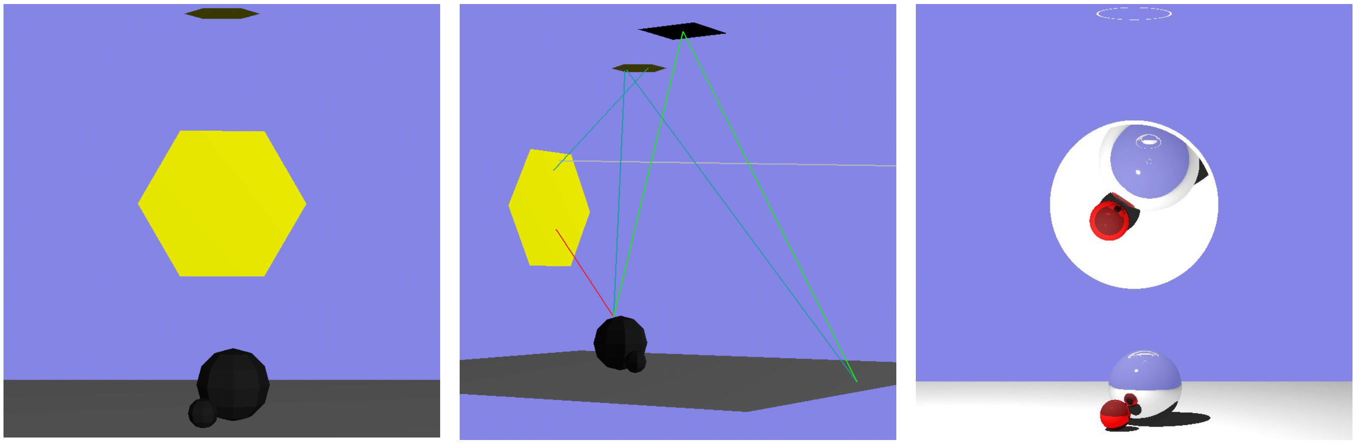
Basic portal scene



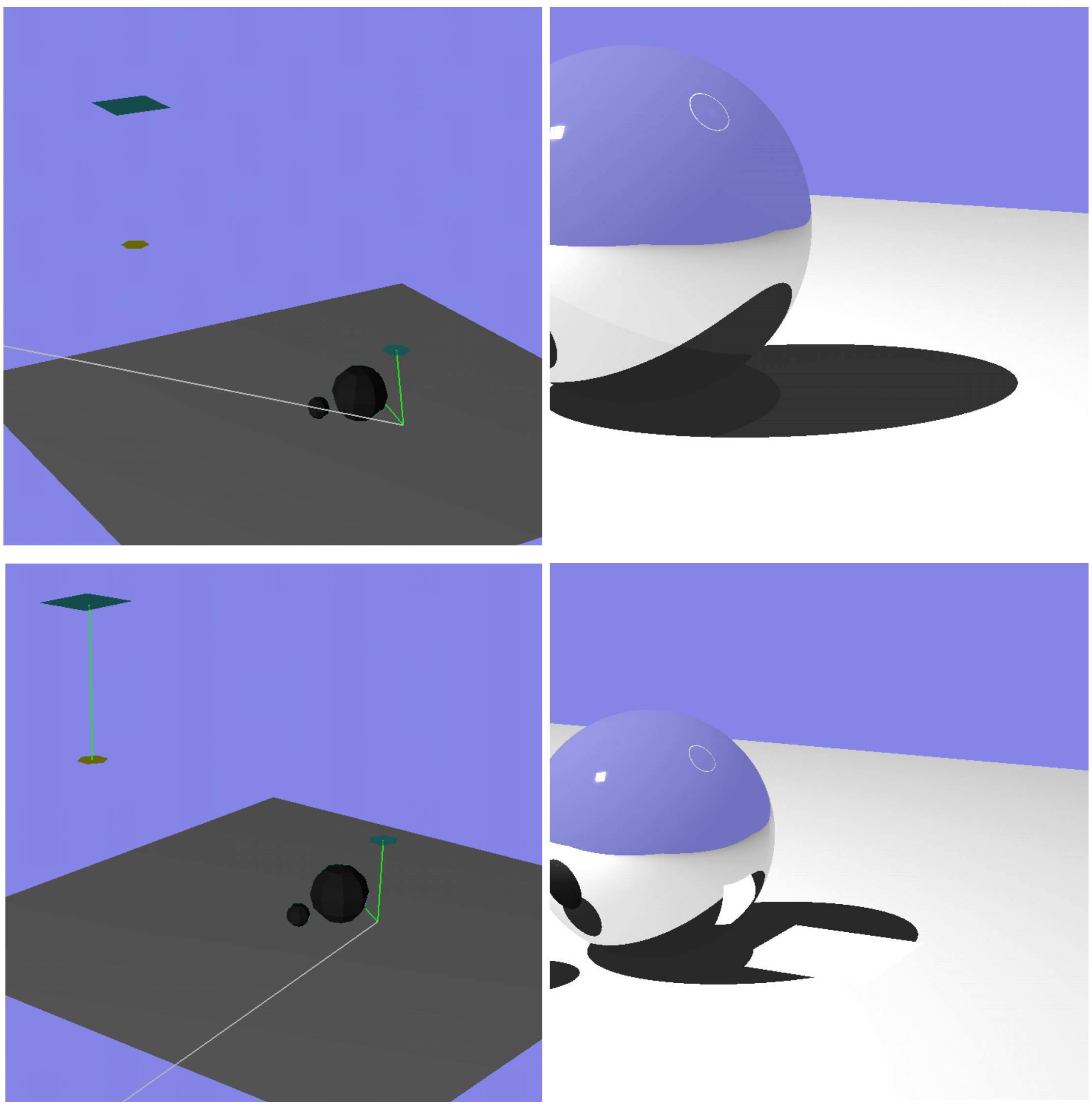
“Infinite hallway” scene

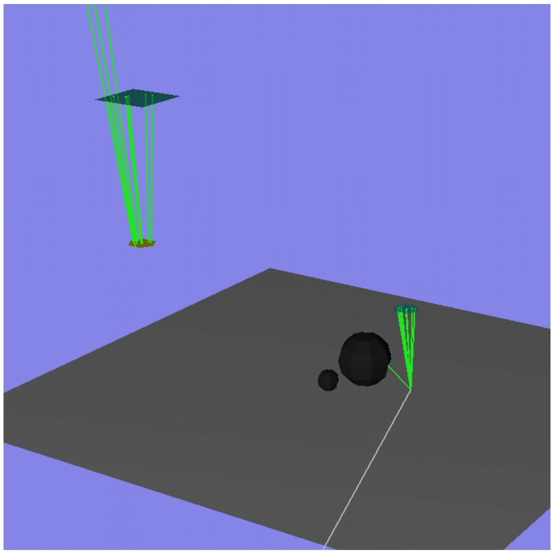
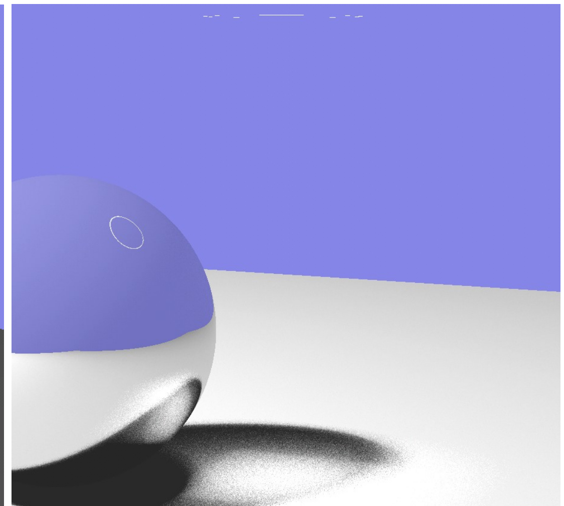


Infinite Spiral scene

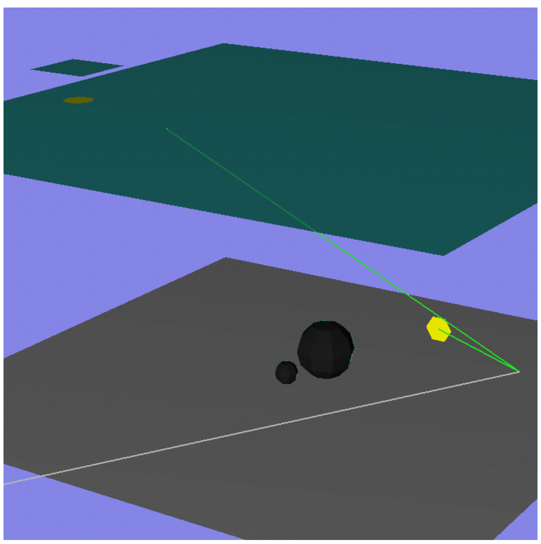
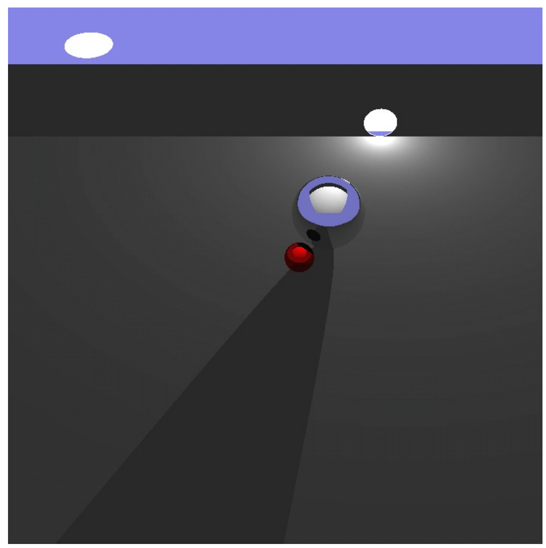
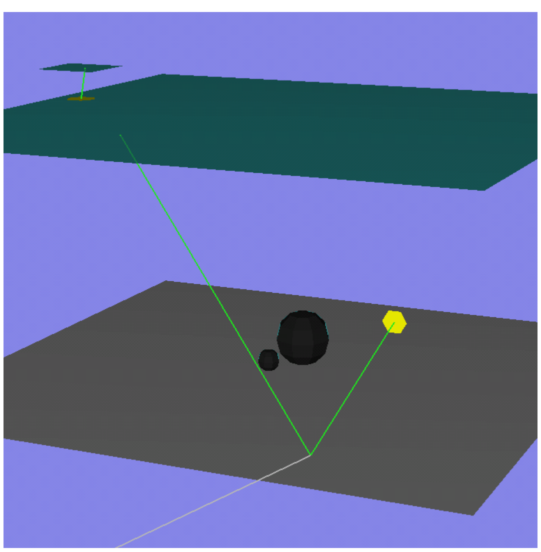
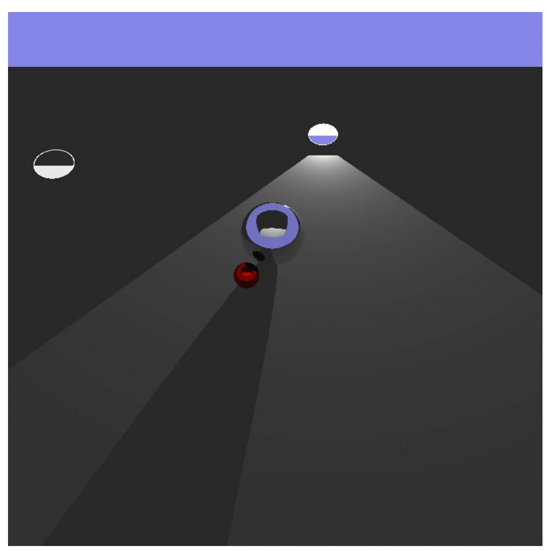


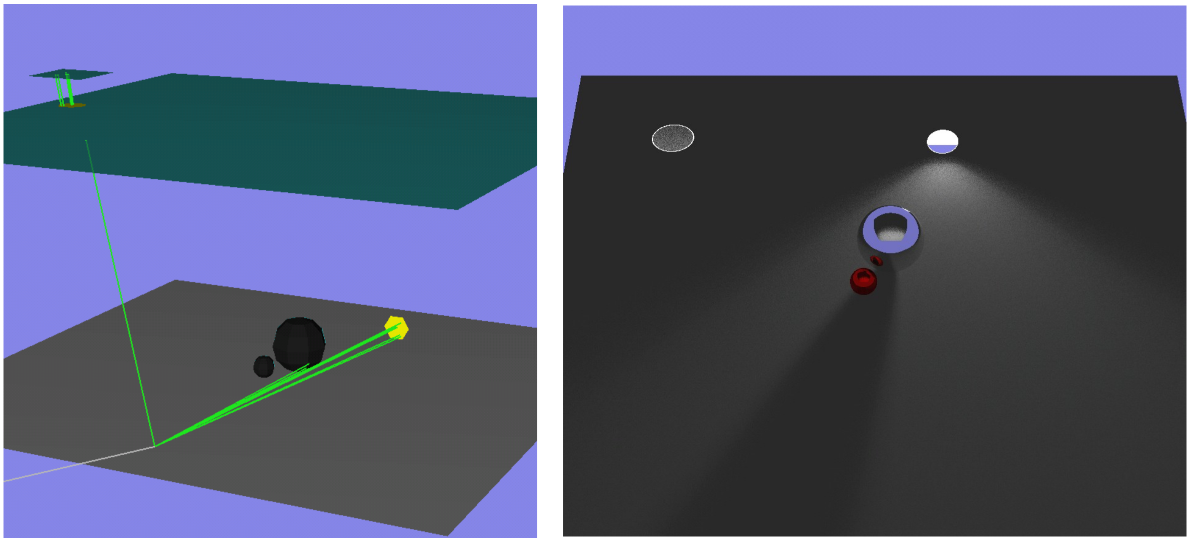
Aerial portal scene



Plain light scene



Blocked light scene