Raytracing Through Portals

Max Sichel Sol Toder

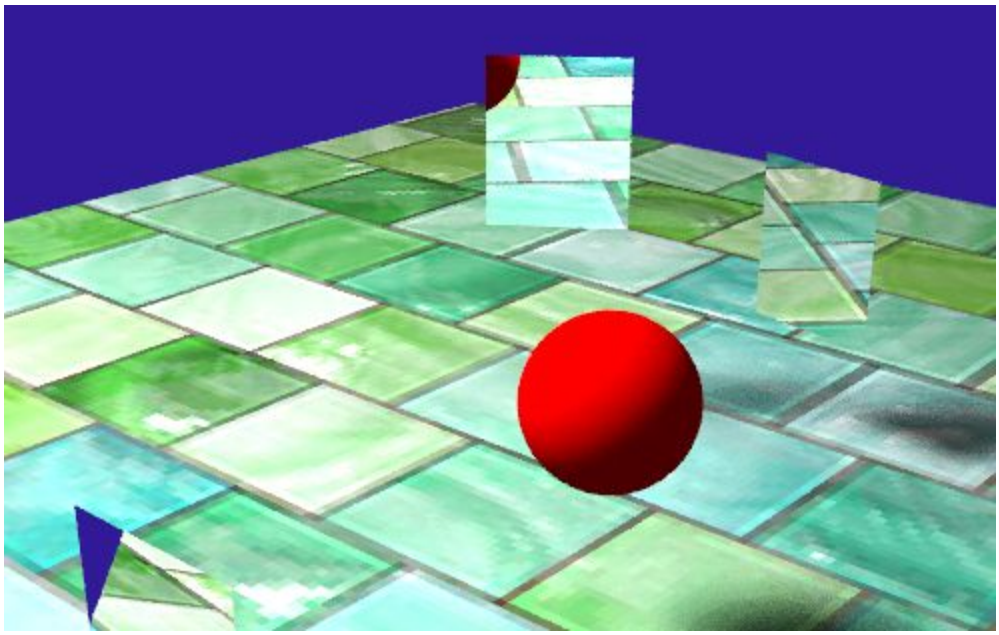

Figure 1: A view of rays passing through portals, creating shadows and showing other parts of the scene.

1. ABSTRACT

In this paper, we describe a method for rendering a raytraced image that contains portal planes. Every portal planes is connected to another portal plane in such a way that the two portal planes may be said to occupy the same area, despite being in two different locations. In other words, any ray that strikes a portal plane will be instantly transformed so that it is exiting the related portal plane at the same relative position and relative angle. This allows viewing the area behind one portal plane through a different portal plane, as well as other related effects.

1.1. Introduction

Simulating portals with raytracing is in many ways simpler than simulating them with more standard rasterized graphics. Rather than requiring multiple render passes, alternate cameras, and creative clipping planes, raytracing through portals is a conceptually simple process. We are already tracing rays through space and seeing where they hit; if that hit happens to be a portal plane, create a new ray starting at the equivalent point and with the equivalent angle on the related plane, and continue the trace from there.

Despite this conceptual simplicity, there are still several challenges involved. While a simple modification of the ray casting function will suffice for simple images, lighting and global illumination will need to be modified for fully correct results, and care must be taken to avoid unbounded runtimes or memory usage. Light passing through a portal or set of portals can enter an infinite loop, and the number of angles which must be traced for correct lighting results can grow exponentially, or even become infinite. Boundaries must be set, and approximations must be made, as in any area of rendering.

1.2. Related Work

Portals are not a common object in either real-time or pre-rendered graphics, being an imaginary concept with no real life analogue currently existing. Naturally research into rendering portals is somewhat sparse. Young and Stolarsky[1] provide a method for ray tracing through viewing portals in a scene. Their

portals are all circular and they do not take global illumination into account. Subileau et al. [2] transport photons through portals in order to edit the illumination of a room, however their methods are oriented more towards editing lighting in a scene for artistic effect.

2. APPROACH

2.1. Portal Definition

The general idea of portals is to connect two disjoint points in space as though they were right next to each other. Our portals are sets of two planar portal sides. Rays entering one of the portal sides will be transported out of the other. Each portal side is represented as an affine transformation of a unit square. As Subileau et al. point out, it is important that these transformations correspond to invertible matrices[2]. In our case these are stored as four by four homogeneous matrices in order to simplify the math slightly. In practice our portal sides also keep a reference to both their parent portal and their corresponding portal side for convenience.

2.2. Ray Portal Intersection

The implicit equation for a plane is

$$\overline{N} \cdot (\overline{p} - \overline{C}) = 0$$

where $\overline{N}$ is the plane's unit normal vector, $\overline{C}$ is a point on the plane (the portal's centroid, in our case), and $\overline{p}$ is the candidate point. The explicit equation for a ray is

$$\overline{p} = \overline{R_0} + t\,\overline{D}$$

where $\overline{R_0}$ is the ray's origin, $\overline{D}$ is the ray's unit direction, and t is a non-negative real number representing the distance of the point from the ray's origin. We can combine these to produce

$$t = \overline{N} \cdot (\overline{C} - \overline{R_0}) \,/\, (\overline{N} \cdot \overline{D})$$

which calculates the intersection between a ray and the portal side, or more specifically an infinite plane overlapping the finite portal side. A portal side's normal and centroid may be easily calculated by transforming the unit forward direction vector, $< 0, 0, -1, 0 >$, and the identity position vector, $< 0,\ 0,\ 0,\ 1 >$, through the side's transformation matrix respectively. If $\overline{N} \cdot \overline{D}$ is zero, the ray is parallel to the plane and there is no intersection. If t is less than zero, the intersection is behind the ray's origin and there is no intersection. Finally, we transform the hit point by the side's inverse matrix, and then check whether it is within the original unit square. If it is not, there is no intersection. Otherwise, we have found the intersection point.

At this point, transferring the ray through the portal is quite simple. We transform the hit position and ray direction by the portal side's inverse transformation matrix, placing them in the portal's local space. We then transform them by the opposite side's transformation matrix, returning them to world space as an equivalent position and direction relative to the opposite portal.

2.3. Ray Tracing

Our method was implemented on top of an existing Whitted style ray tracer[3]. Consequently it resembles such ray tracers very closely. In fact, rays that do not interact with portals are traced out exactly as such a ray tracer normally would with the exception of added global illumination.

When a cast ray does hit a portal a new ray is cast through the portal and the hit data or color of the new cast is returned as though it were the original ray cast[1]. If the new ray hits a portal, then the process is repeated until either a surface is intersected or the maximum portal recursion depth is exceeded. The portal recursion depth exists to prevent the ray tracer from overflowing the stack or running out of memory. At the maximum recursion depth, portals are ignored so rays pass through portals as if the portal was not there. This can lead to some noticeable artifacts when portals should be visible through other portals and the portal recursion depths, as seen in **Figure 2**.

2.4. Direct Lighting

Direct lighting (**Figure 6**) is traditionally calculated by casting a ray towards a light. If the ray hits the light, the energy of the light is divided by the square of the distance otherwise it is taken to be zero[3]. With portals in the scene, it is possible that a light might be visible through a portal. Thus, we need to cast rays through portals as well. We use Monte Carlo methods to calculate the visibility of the light.

For each light, we cast a fixed number of rays to each light directly and through each portal side. The exact number is specified as an argument to the program. The first raycast always goes to the centroid of the light face. All subsequent rays are cast towards a randomly selected point on the light face. If a cast ray hits the light, its energy is divided by the total traveled distance squared and added to the total energy of the pixel.

Casting to lights through portals involves three steps. First the direction of the light is determined by transforming the light's position through the portal. Then, the original ray is cast towards the portal side to make sure it hits the portal and doesn't hit anything else first. Finally, the ray through the portal is checked to make sure it hits the light. If any of these conditions is not met, the ray is ignored.

2.5. Global Illumination

Global illumination (**Figure 7**) is calculated in our rendering engine using photon mapping first proposed by Jensen[4]. First photons are cast from the lights into the scene. Their positions, energies, and incident directions are recorded in a spatial map. Then the photons are reflected back into the scene again iteratively until they either miss the scene entirely or their energy is depleted. The map in which the photons are stored is very often a *k*-d tree[5] in order to accelerate the gathering process. When photons hit portal sides, they are transported through to the opposite side. The implementation is similar to the backwards ray tracing used for rendering, except the information from the ray coming out of the portal is used instead of the information from the ray being shot into the portal.

Gathering photons is a bit more difficult. Subileau et al. effectively ignore the problem by only using portals for casting photons[2]. Our implementation collects photons through portals one iteration deep, but stops there. To gather photons we first guess a sized sphere to gather photons from. If we don't collect enough photons, we double the size of our guess. To collect through portals, we first transform our point through each portal side and check if the portal falls within the test sphere. If not, it is ignored. For all other portals, we test photons for two conditions. First we test the incident direction $\overline{I}$ and the surface normal $\overline{N}$ for $\overline{I} \cdot \overline{N} < 0$ in order to ignore photons from other surfaces of thin objects. We also make sure that the line segment from the transformed point to the photon intersects the portal side's partner to prevent gathering photons on the wrong side of the portal.

3. CHALLENGES

3.1. Illumination Recursion

Getting illumination to pass through a single portal is relatively simple. When attempting to cast to the light source, perform an additional cast attempt through each portal side in the scene. This allows a portal to add light to an otherwise shadowed area, producing the expected result. However, for full correctness, light should be allowed to pass through any number of portals. This exhibits potentially unbounded behavior, as passing through a portal does not remove it from the list of candidates. It is entirely possible for an illumination ray to pass through the same portal multiple times on its way to a light source. This means that even if we cap the recursion at some maximum depth, the growth rate is exponential rather than linear.

For this paper we have elected to cap lighting recursion at a single portal passthrough. Implementing full recursive illumination would increase render times and memory usage significantly, and the improvement would be minimal in most cases considering lighting falloff with distance and the diminishing area of each recursion. We elected to cap photon gathering recursion depth to one for the same reasons.

4. RESULTS

We believe that our method of rendering portals works well and produces effective results. Objects can be viewed through portals. Objects can be viewed through multiple portals. Direct lighting, indirect lighting, and caustics can pass through portals to illuminate distant surfaces. Performance is decent; when direct or no lighting is used, renders with a maximum recursion depth of 100 take not much longer than renders with a maximum recursion depth of 1, largely due to the diminishing area with each recursion.

4.1. Known Limitations

For the reasons mentioned above, our implementation caps illumination recursion at a single level of portal passthrough.

5. CONCLUSION

5.1. Division of Work

Sol provided most of the core code, such as loading portals, storing portals, linking portals, accessing portals, intersecting with portals, and transforming position and direction vectors through portals. He also created most of the scenes used for testing and in this paper.

Max provided the actual implementation of raytracing through portals, adding portal recasting to the raycasting function and extending it to support returning the portal struck, if any. He extended direct lighting to take portals into account and added portal support to the raytracer's photon mapping engine, allowing photons to pass through portals to create new caustic effects.

5.2. Room for Improvement

While our current implementation caps illumination recursion to a single level for reasons of performance, a future implementation could implement it fully for complete realism. Alternatively, using photon mapping or a similar forward raytracing method for direct illumination would simplify the code and potentially reduce memory consumption. Using photon splatting as opposed to photon gathering could potentially improve performance as photon gathering is the current bottleneck of our rendering engine.

6. REFERENCES

[1] Young, C. and Stolarsky, I. 2008. Ray Tracing through Viewing Portals. *RPI ACG class, Spring 2008*.

[2] Subileau, T., Mellado, N., Vanderhaeghe, D., and Paulin, M. 2015. Light Transport Editing with Ray Portals. *Computer Graphics International 2015*.

[3] Whitted, T. 2005. An improved illumination model for shaded display. *ACM SIGGRAPH 2005*.

[4] Jensen, H.W. 1996. Global Illumination using Photon Maps. *Eurographics Rendering Techniques '96*, 21–30.

[5] Bentley, J.L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM18*, 9, 509–517.

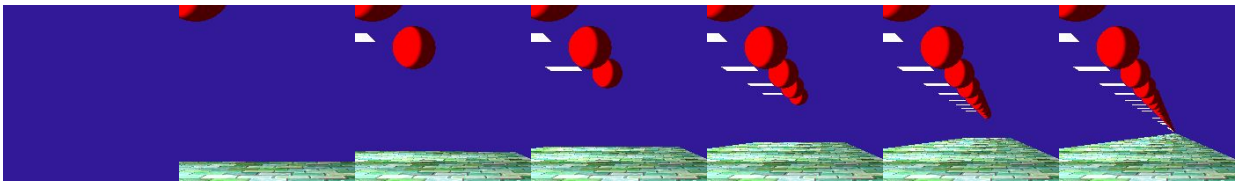

Figure 2: A view of two linked portal planes facing each other, with different maximum recursion depths. From left to right, they are 0, 1, 2, 3, 5, 10, and 100.

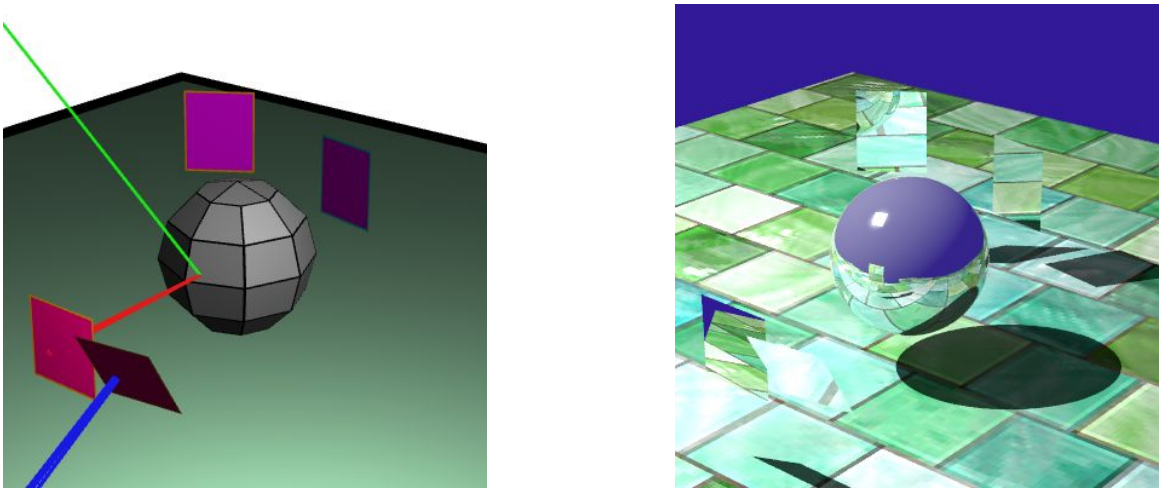

Figure 3: A display of how rays pass through portals. A scene is seen in preview mode on the left, with a ray shown being emitted from the camera, bouncing off the reflective sphere, and being transferred through a portal. The resulting image can then be seen on the right. Note that the struck point on the sphere is displaying a view through the portal.

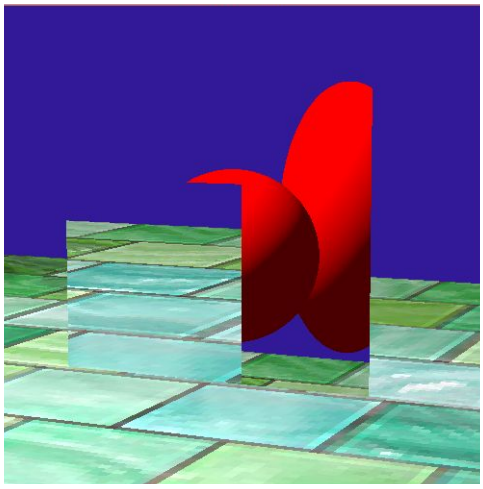

Figure 4: Differently sized portals. The scene contains a red sphere and two linked portal sides, one square and one rectangular. The images seen through the portal are noticeably distorted.

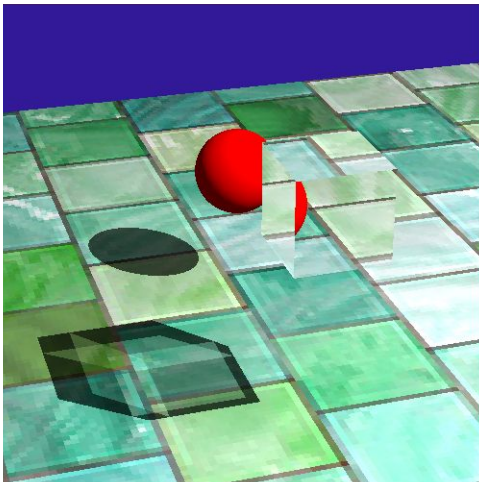

Figure 5: An "invisibility cloak." The scene contains a red sphere and a cube made up of six portal sides, linked so that rays entering the top exit the bottom and so on.

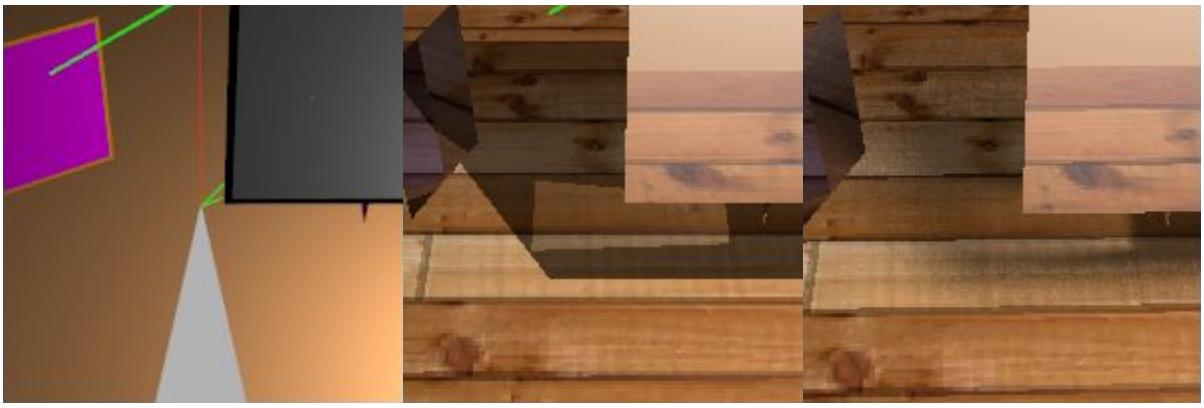

Figure 6: The same scene with a ray tree view(left), rendered with hard shadows(center), and rendered with soft shadows(right). Note the light ray passing through the portal to the point on the floor behind the mirror.

Figure 7: Photon mapping through portals. Note the portal capturing photons inside the ring, the hole in the caustic it creates, and the new caustic outside the ring.