Real Time Shadows, Reflections, and Transparency using a Z buffer / Ray Tracer Hybrid

Abe Megahed
Hypercosm, Inc.

Abstract:
We describe techniques for merging ray tracing with real time hidden surface algorithms such as the z buffer algorithm. The technique offers a smooth transition from interactive display with approximate soft shadows, reflections, and transparency to full ray tracing with all of its associated effects. This is done by using the ray tracer only for tracing shading rays at the vertices of polygons and using another hidden surface technique for hidden surface removal in conjunction with Gouraud interpolation.

Keywords: ray tracing, z buffer, Gouraud shading, real time rendering, and real time ray tracing

Author's Address:
Abe Megahed
2104 Mill Street
Cross Plains, WI
53562

1. Introduction

As computer graphics have evolved, we have seen a schism emerge in rendering techniques between the so-called 'real-time' techniques and the 'photorealistic' techniques. This has lead to most interactive computer graphics having a characteristic look, with diffuse shading and highlights included because they can be done quickly and easily, but a noticeable absence of shadows and reflections. Photorealistic images, on the other hand, tend to be characterized by having a lot of reflective surfaces in addition to shadows and transparency, simply because these effects are possible. The approach we have taken has been to integrate these techniques in a uniform way so that as faster hardware and multiprocessing become available, we can have a smooth transition from the traditional z buffer to more realistic ray tracing algorithms.

2. Background

Although the performance of current workstations has increased considerably, they still lack the performance necessary for real time ray tracing. Several notable attempts have been made at building massively parallel ray casting machines [NISH83a] [POT89], however, these have been specialized and rather expensive systems. The problem is basically that we do not have enough computing power to trace rays at every pixel in an image at interactive rates. So, we must be satisfied by tracing a smaller number of rays than we would like. If we perform a coarse ray trace and magnify the size of the pixels, an unacceptably crude and blocky image results. Ray tracing is simply too slow to use as a hidden surface algorithm. If we replace the tracing of primary rays with another hidden surface algorithm such as the z buffer, then the ray tracer is used only for the shading. Although this helps considerably, if we trace secondary rays at every pixel of a reflective object, then the technique will still not be fast enough for interactive display of useful scenes.

3. The Rendering Algorithm

If we trace rays at select points distributed across the surface of the object and interpolate the color across the object, then we may only need to trace a few hundred or thousand rays per frame. This can be done at interactive rates. In addition, we can use high performance z buffer and Gouraud interpolation hardware to perform the tasks of hidden surface removal and color interpolation, respectively. This is the basis for our technique. With this technique, we have found that on current workstations, it is possible to compute shadows, reflections, and transparency in scene with a few hundred or thousand polygons at acceptable interactive rates.

3.1 Benefits of Interpolation

One of the most striking things about the images that are produced is that the interpolation technique causes reflective objects to have the appearance of being diffuse reflectors. The same effect causes shadows and transparency to appear ‘soft’. Previously, this effect could only be achieved through extremely expensive distributed ray tracing. Although the diffuse reflections, transparency and shadows computed through interpolation are not physically accurate, they give a close enough illusion of the phenomena to be convincing.

3.2 Disadvantages of Interpolation

There are two major disadvantages with ray tracing at select vertices and interpolating the color in between. The first problem is that the appearance of the object is dependent upon the underlying tessellation. The second problem is that we are using a smooth interpolation to approximate the specular and transmitted components of the surface of the object which often change very abruptly compared to the diffuse component. This leads to shading anomalies.

1. **Effects of Tessellation**
   Because we trace rays only at the vertices, the appearance of an object is highly dependent upon its tessellation. This means that where an object is tessellated more finely, it will appear more specular and where the tessellation is coarse, the objects will appear dull. In addition, shadows and transparency will be sharper in regions of high tessellation which does not make sense from a physical standpoint. There may be ways to combat these effects, such as tessellating overly large polygons in screen space. It has been demonstrated that this can be done in real-time, for example, by Silicon Graphics in their early texture mapping systems (VGX).

2. **Effects of Sampling**
   Another disadvantage of the algorithm is due to the sampling nature of the algorithm. Since specular reflection, refraction, and shadows change very abruptly across the surface of an object, a reflection passing by a vertex will show up as a brief smear of color, which may appear or disappear depending upon whether the reflection hits a vertex. This shows up readily during animations. The problem is inherent with insufficient sampling. We could perform ‘antialiasing’ by casting several rays around the vertices, however, this would diminish the frame rate and the benefits of interactivity which are prime benefits of the technique in the first place.

4. B-Rep Considerations

The technique is actually more suited to rendering solid models than to rendering surface models for a number of reasons.

1. **Back facing Polygons and Vertices**
   With solid models, we can ignore back facing polygons and their associated vertices. This decreases the number of polygons to scan convert and the number of vertices to ray trace by a factor of two.

2. **Interior Surfaces**
   If a solid model is consistent, it has no surfaces that lie completely in the interior of the solid. This is advantageous because we can avoid shading (ray tracing) and rendering these surfaces. This has additional importance because these interior vertices are more expensive to shade since reflected rays will often bounce around the interior of the object without being able to ‘escape’. This makes these interior vertices slower to shade than vertices which lie on the exterior surface of an object.

3. **Interpenetrating Surfaces**

If we use consistent solid models with no interpenetrating faces, we do not have to worry about shading anomalies that can occur when we have interpenetrating surfaces. To illustrate how these shading anomalies arise, imagine a sphere and a cone with its apex just a tiny bit inside of the sphere. When we shade the cone, the vertices at the apex will be in shadow and reflect only the sphere because they are on the interior of the sphere. When we interpolate the color, the dark shadow and the color of the reflected sphere will be spread out over the surface of the cone, even though only the tip of the cone should be affected. This problem may give rise to serious shading anomalies in models with many interpenetrating objects and coarse tessellation. With solid models, we can be sure that all of the vertices lie on the exterior of the object and these problems do not arise.

5. Comparison with Alternative Techniques

All non ray tracing rendering systems are incapable of rendering true reflections and transparency. In addition, many systems that use the z buffer can not produce true shadows. However, the success and speed of the z buffer algorithm at hidden surface removal and its conceptual simplicity have prompted many attempts to shoehorn these additional features into the system. Many of the effects produced by ray tracing can be simulated through various z buffer techniques.

There are two primary techniques for simulating shadows with the z buffer. The simplest type of shadowing effect is ground plane shadows. These shadows are created by projecting the object onto the ground plane and drawing dark polygons where the projections lie. These shadows have hard edges and are only cast onto the ground plane, which limits their usefulness. A more general solution is available through shadow mapping. Unfortunately, the shadow maps use a large amount of memory and are prone to artifacts.

Reflections are often simulated using environment maps, images of the scene from the viewpoint of the object, which are then mapped onto the object as reflections. Although these look like reflections at first glance, they do not have the proper geometry for true reflections.

Transparency can be simulated by z buffering machines with special frame buffer hardware for 'alpha-blending', however, the distortions caused by the refraction of light through different media cannot be accurately simulated.

Although these techniques are very useful in the hands of a skilled and experienced computer artist, they are not as general purpose and easy to use as a ray tracer. In addition, after these additional features are added, the initial simplicity and speed of the z buffer algorithm is lost. These same effects can all be computed simply and precisely through ray tracing. It therefore makes sense to restrict the z buffer to what it does best, hidden surface removal, and to restrict the ray tracer to the tasks that only it can perform.

6. Conclusions

We have demonstrated the practicality and usefulness of the algorithms described here in a prototype system. We have found that the inclusion of reflections and shadows in a real-time system not only contributes to a better understanding of spatial relationships, but also increases the gamut of possible surface appearances. While this technique may not be appropriate for every real time application, we have found that in certain circumstances, it can be very effective.

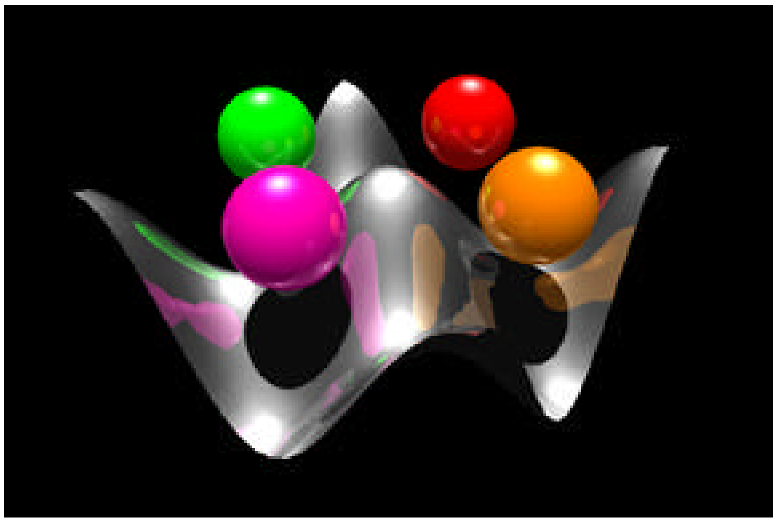

Figure1: An image for which the ray tracer and shading model have been applied at every pixel

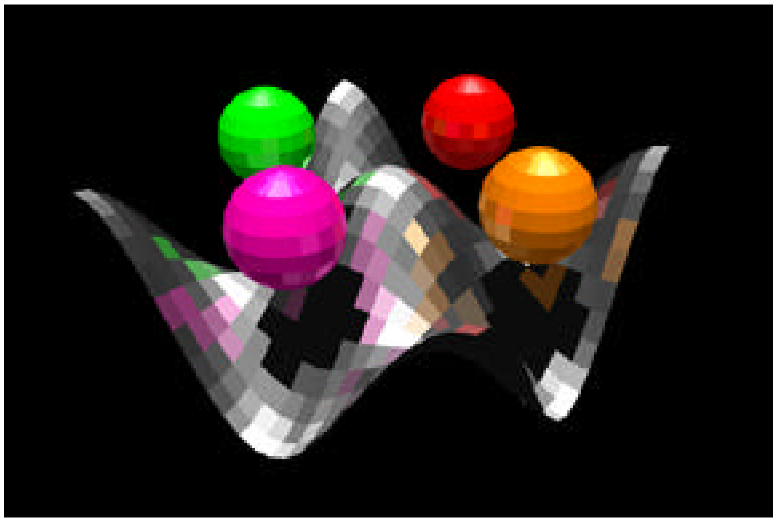

Figure2: An image that has been generated using the z buffer to scan convert polygons and the ray tracer to compute the shading at every face

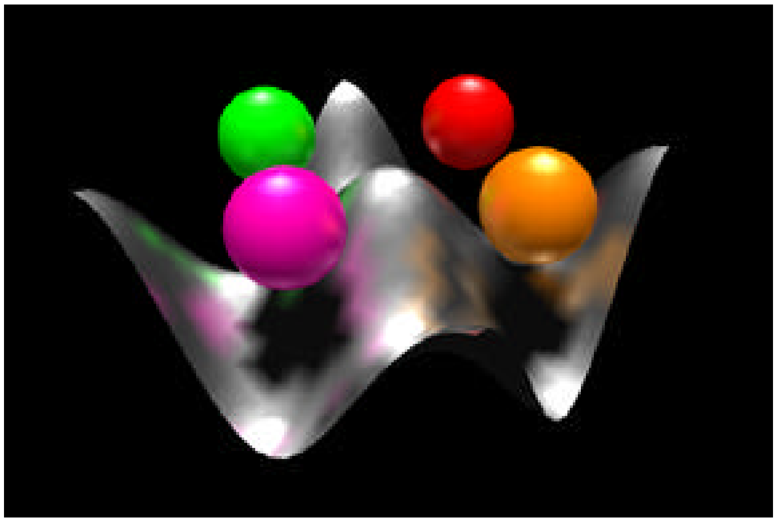

Figure 3: An image that has been generated using the z buffer algorithm to scan convert polygons, Gouraud shading to interpolate the colors between vertices, and the ray tracer to compute shading at every vertex