Computer Graphics Project
Visual Effects

Andreas Van Barel

May 22, 2023

1 The raytracer

The implemented components of the raytracer are as follows. The analytic shapes that the raytracer can handle are spheres, boxes, cylinders, planes, and triangles. It can also load triangle meshes from .obj files. The triangle meshes can use a normal map. A bounding volume hierarchy can be generated to speed up ray tracing of the triangle meshes. An object in a scene consists of a combination of a shape and a material. The materials can have different BRDF (Bidirectional Reflectance Distribution Function) models, they can be textured, reflective, refractive, or glossy. For light sources, there is a choice of directional light sources, point light sources, and area light sources. Other features include anti-aliasing, reflection maps (per object and/or for the entire scene), and depth of field.

2 BRDF models

2.1 Phong and glossy variant

First, let's compare the Phong BRDF model with a glossy variant of it. The diffuse lighting is calculated in the same way for all the models discussed in this paper:

$$L_d = k_d c_{light} L_{light} c_d \cos(\theta)$$

L_d	Outgoing radiance due to diffuse component
k_d	Diffuse coefficient
c_{light}	Color of the light source
L_{light}	Incoming radiance from the light source
c_d	Color for diffuse illumination
θ	Angle between the normal and the direction of outgoing radiance

The specular shading is performed in the Phong BRDF model as follows:

$$L_s = k_s c_{light} L_{light} c_s \cos^e(\alpha) \tag{1}$$

L_s	Outgoing radiance due to specular component
k_s	Specular coefficient
c_s	Color for specular illumination
α	Angle between the direction of incoming radiance and the perfect specular direction

In the glossy variant, the diffuse lighting is calculated in the same non-recursive manner. The specular lighting is computed by shooting rays around the direction of perfect specular reflection according to a distribution d (samples per solid angle) proportional to

$$d \sim \cos^e(\alpha)$$

According to section 7.2 of the handbook, such a distribution can be obtained by mapping two random numbers $(r_1, r_2) \in [0, 1]^2$ as follows:

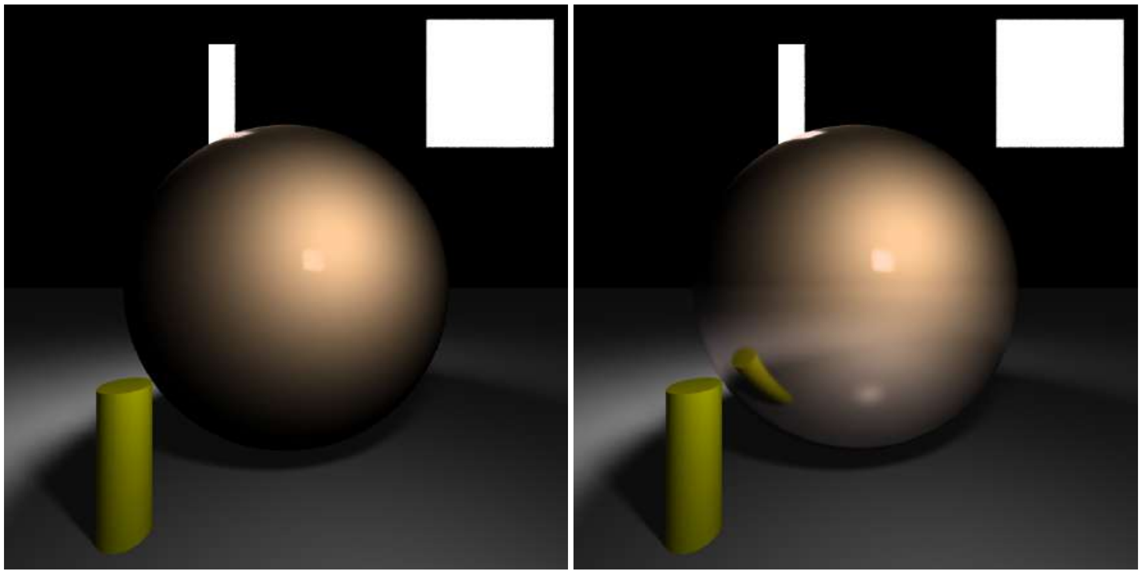
$$\begin{aligned}\phi &= 2\pi r_1 \\ \theta &= \arccos((1 - r_2)^{1/(e+1)})\end{aligned}$$

Here, ϕ represents the angle with the vector in the direction of perfect specular reflection, and θ represents the angle around this direction. In other words, (ϕ, θ) are polar coordinates in a coordinate system where the azimuth lies according to the perfect specular reflection. When generating such samples, it is possible for them to fall back onto the surface they should depart from. These poor samples are simply not used. For each of the samples that are used, the shading is computed using the specular Phong shading of equation 1.

Both models use the following concrete parameters:

$$\begin{aligned}c_d &= (1, 0.8, 0.6) \\ k_s &= 0.2 \\ c_s &= (1, 0.9, 0.9) \\ k_s &= 0.8 \\ e &= 1000\end{aligned}$$

The result can be seen in figure 1. Both were rendered with 50x anti-aliasing. The Phong



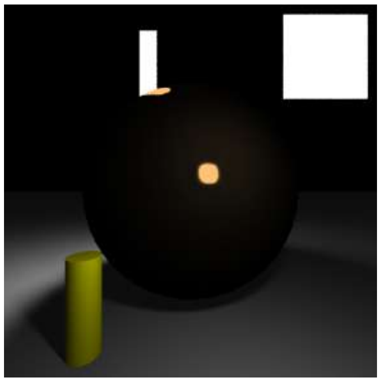
(a) Sphere with Phong specular reflections (b) Sphere with glossy Phong specular reflections

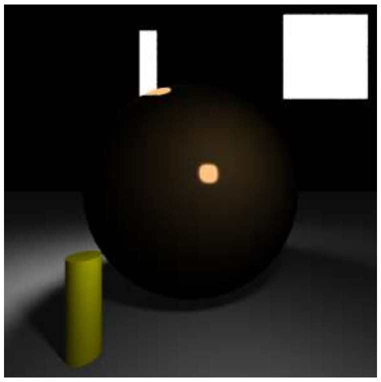
Figure 1: Effect of glossiness on specular reflection

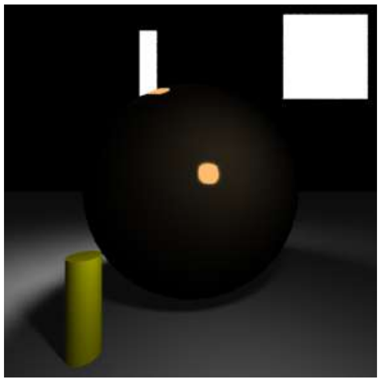
highlights were rendered with 25 samples per area light source. The glossy variant was rendered with 10 glossy samples per intersection of the sphere. In this case, the area light sources were not sampled. Their contribution is implicitly taken into account through the glossy samples. The glossy specular reflections are visually much more appealing. Not only the light sources but the entire scene is reflected in the glossy sphere. A cylinder was specifically placed to make this clear. The disadvantage is, of course, that glossy reflections take approximately 10 times longer to compute. It is also interesting to note the glow at the edge of the sphere produced by the area light source behind the sphere. This is mainly the result of the diffuse lighting caused by the light source.

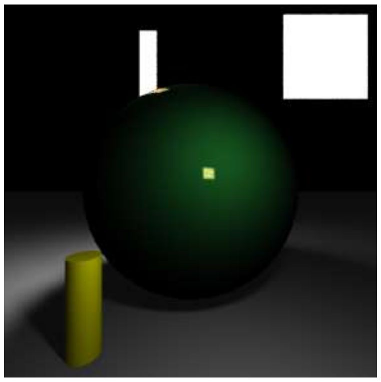
2.2 Fitted BRDF Models

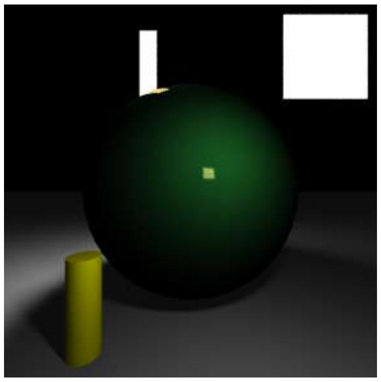
Three BRDF models from the paper were implemented: the Ward model, the Ward-Duer model, and the Blinn-Phong model. In each of the three models, the diffuse lighting is still calculated in the same way. They were compared for the fitted materials Nickel, Acrylic-Green, and Derlin. For Derlin, the front light source was slightly dimmed to avoid color saturation. Color saturation

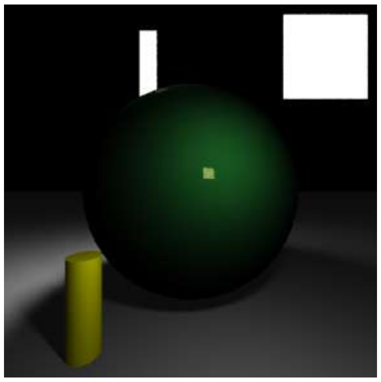

(a) Ward: Nickel

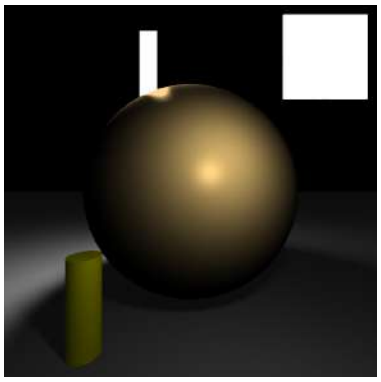

(b) Ward-Duer: Nickel

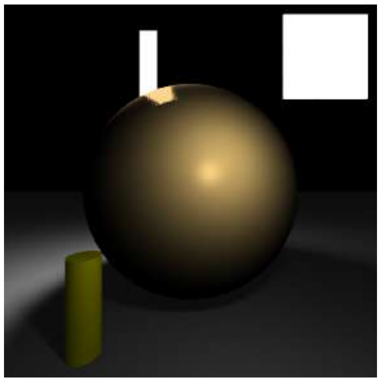

(c) Blinn-Phong: Nickel


(d) Ward: Acrylic-Green


(e) Ward-Duer: Acrylic-Green


(f) Blinn-Phong: Acrylic-Green


(g) Ward: Derlin


(h) Ward-Duer: Derlin

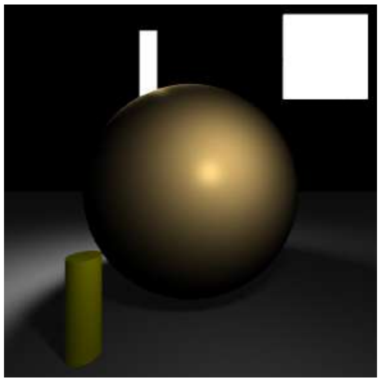

(i) Blinn-Phong: Derlin

Figure 2: Comparison of the fitted material models.

actually occurs in Nickel as well, but since the rest is already dark, we did not dim the light sources there. The biggest differences occur when the viewing ray hits the surface at an angle close to 90°. Most physical materials become more reflective in this case. In the Blinn-Phong model (and also in the regular Phong model), these effects are not taken into account. Ward and Ward-Duer produce better results.

3 Physically Correct Reflection and Refraction

First, we use an environment map and a sphere to discuss these effects. We start by examining a reflective sphere with a reflection coefficient of 0.5 and a brown color. The result is shown in Figure 3.

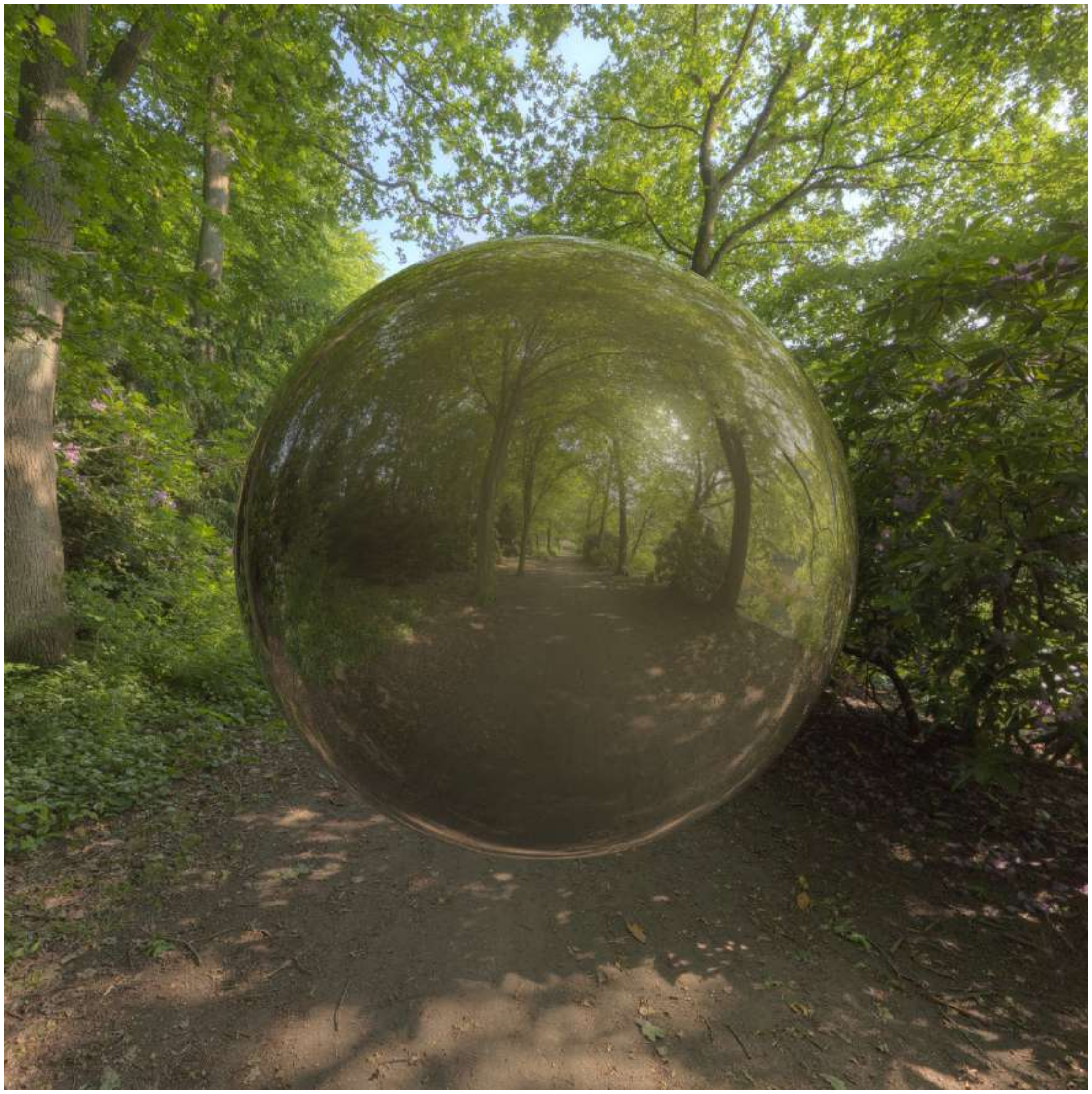
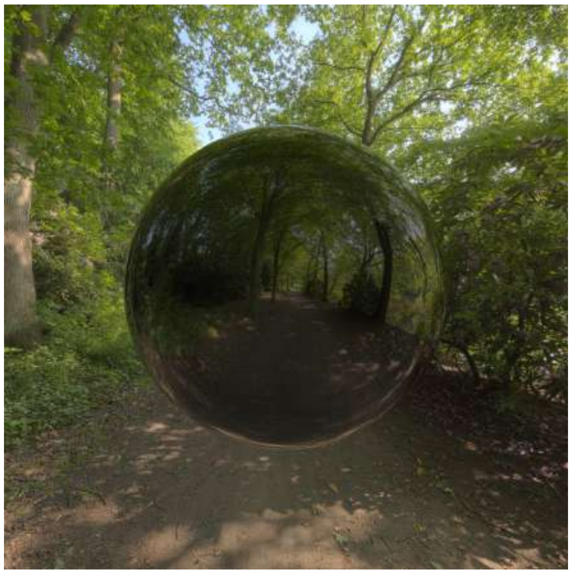
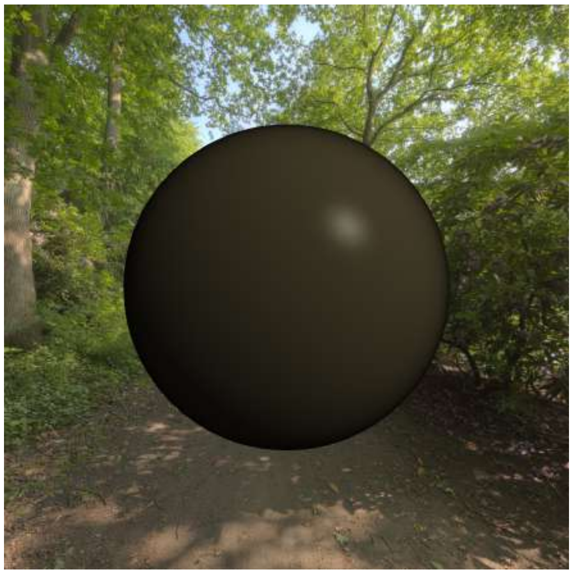


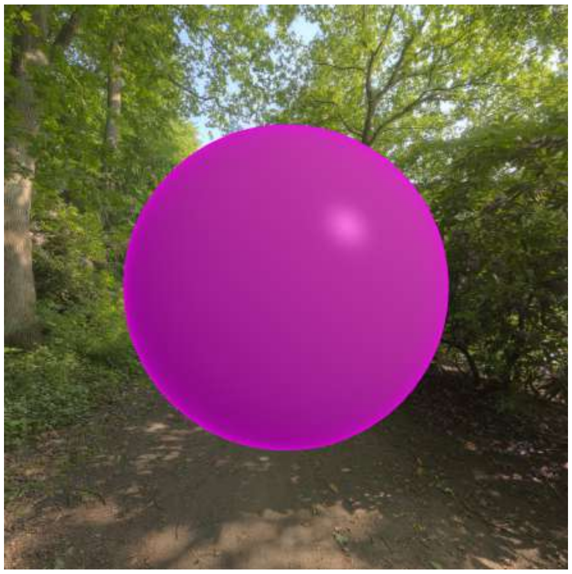
Figure 3: Sphere with reflection coefficient of 0.5.

The individual components in the shading are shown in Figure 4. The reflection component is depicted in Figure 4a, and the regular Phong shading component in Figure 4b. It is clear that reflection becomes more prominent when the ray is incident at a very oblique angle. This can be seen more clearly in Figure 4c, where the reflection is computed as if it were derived from a completely magenta environment map. When this angle-dependent behavior is ignored and there is always 50

Next, we consider a refractive sphere. The sphere has a refractive index of 1.5 (approximately a glass sphere), and the color is pure white, which means there is no attenuation of radiance when a ray passes through the sphere. This is done to make the different parts in the shading more apparent. Figure 5 shows the result.


(a) Reflection only.


(b) Phong shading only.


(c) Reflection in magenta.

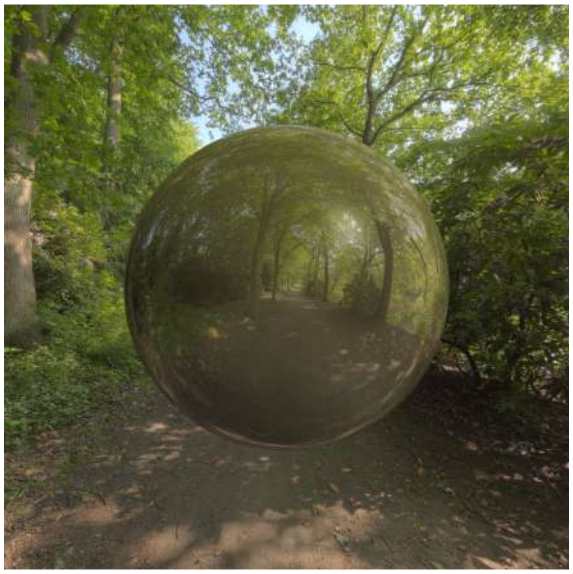

(d) Angle-independent reflection.

Figure 4: Analysis of the refraction terms.

Now, let's examine the individual components in the shading. The reflection component is shown in Figure 6a, and the refractive component in Figure 6b. For clarity, Figure 6c calculates the reflection as if it were derived from a completely magenta environment map. It is clear that reflection is especially significant when the ray is incident at a very oblique angle. If the angle-dependent behavior is ignored and there is always 100

4 Normal Mapping

Normal mapping is achieved by modifying the normal at the intersection point. This can be done by adding values to the normal in a well-defined coordinate system that depend on the intersection point. In this case, the normal is simply overwritten by a normal vector encoded in

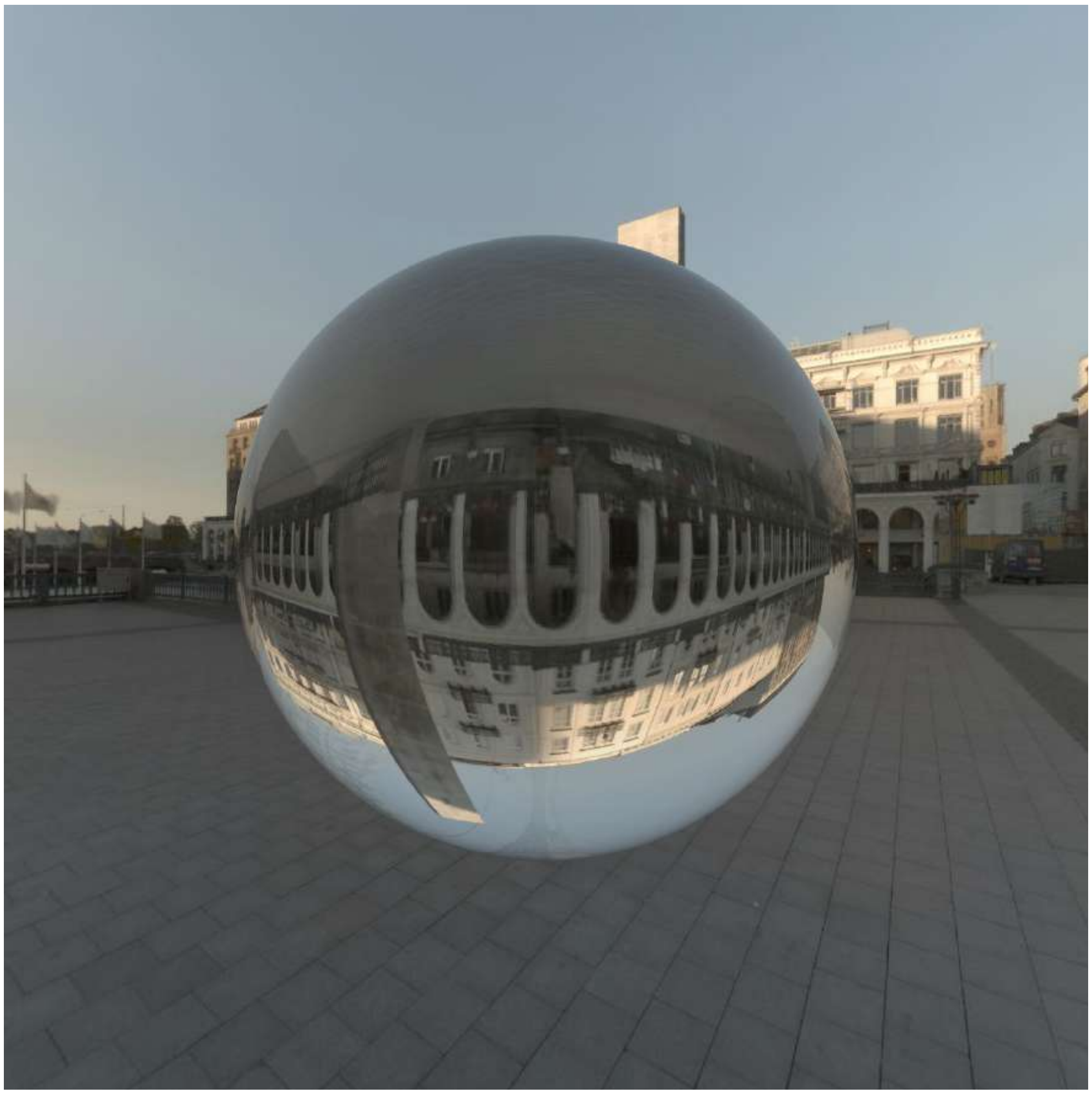
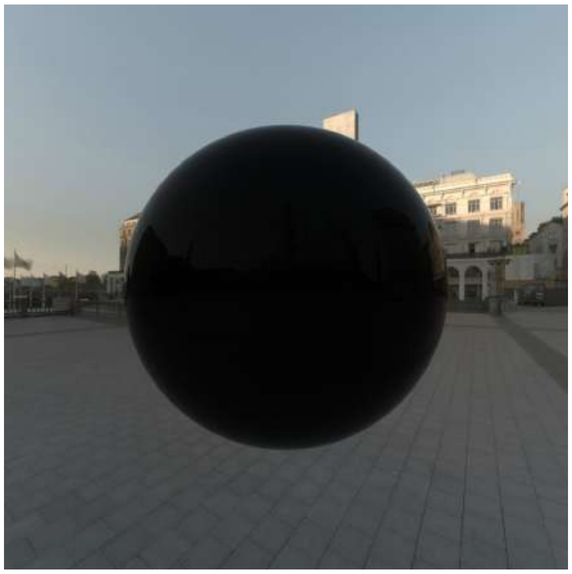
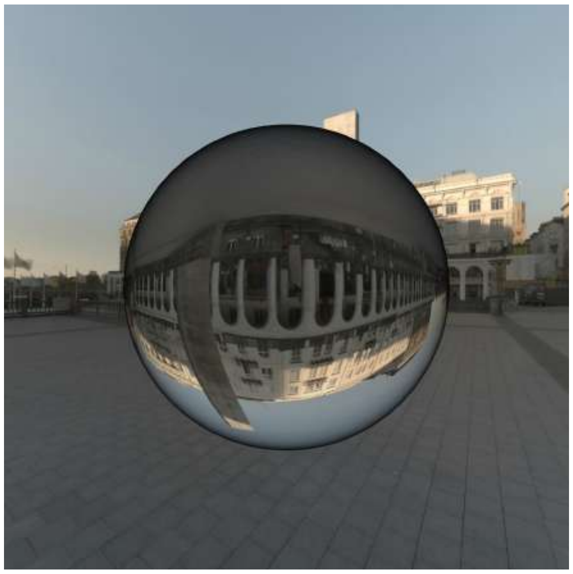
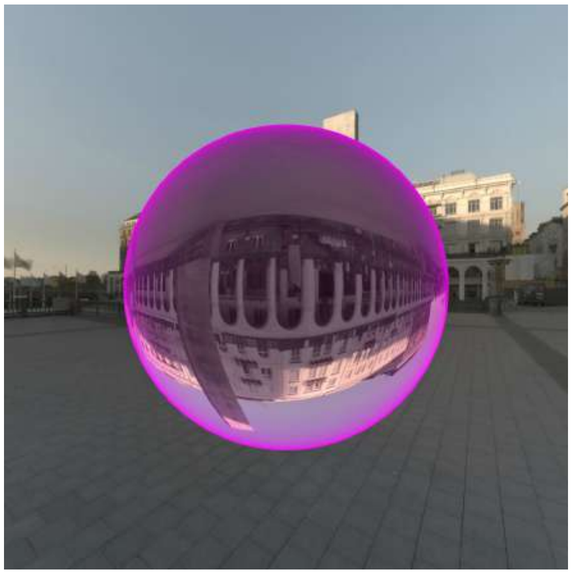


Figure 5: Sphere with refractive index of 1.5.

the RGB components of the normal map. Here, normal mapping is only implemented for triangle meshes. The lookup in the normal map is performed using the usual UV coordinates obtained from the intersection with the mesh. Figure 7 shows the effect of a normal map on the shading of an apple. The normal map clearly provides the illusion of a lot of detail at a very limited computational cost. Note that the silhouette of the model (and thus the shadow) does not change. Also, observe the shadowed side of the apple. The transition from the illuminated side to the shadowed side remains unchanged and is more pronounced in the model with the normal map. This is because the normal map can perturb the normal even close to the edge in such a way that the shading calculation yields a significant amount of light. However, beyond the edge, this is not possible in the current implementation. Only the normal changes, and when tracing a shadow ray, the apple is always intersected on the shadowed side, resulting in no light. This is true even if the normal at that point is perturbed in such a way that the angle with the ray towards the light source is less than 90°. This could be addressed by adding exceptions when intersecting shadow rays. For example, an intersection point that is very close to the origin of the shadow ray (corresponding to the exit intersection point when the ray exits the apple) could be excluded from


(a) Reflection only.


(b) Refraction only.


(c) Reflection in magenta.

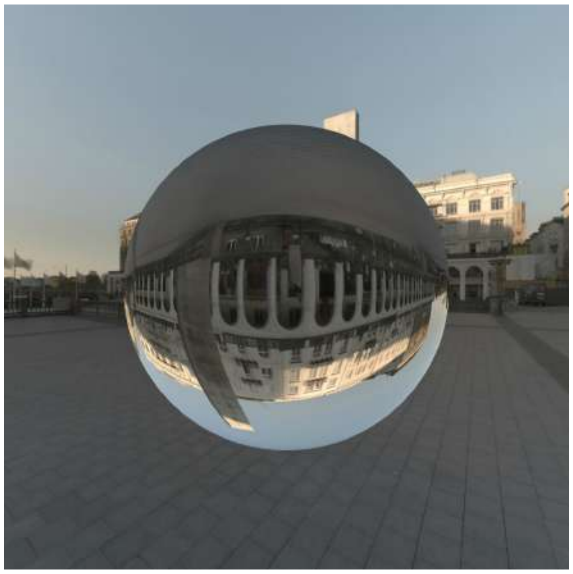

(d) Angle-independent refraction.

Figure 6: Analysis of the refraction terms.

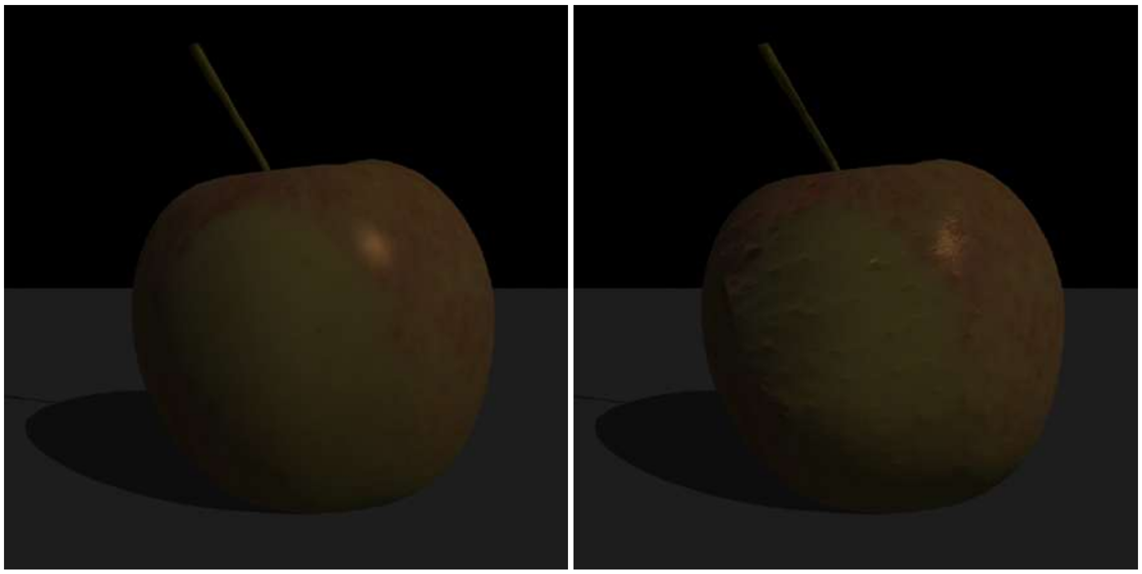
counting when the angle between the normal and the shadow ray is less than 90°.

5 Depth of Field

Depth of field was implemented in the standard way. Some results are shown in Figure 8. Note that the reflections in the sphere are not necessarily in focus when the sphere itself is in focus.

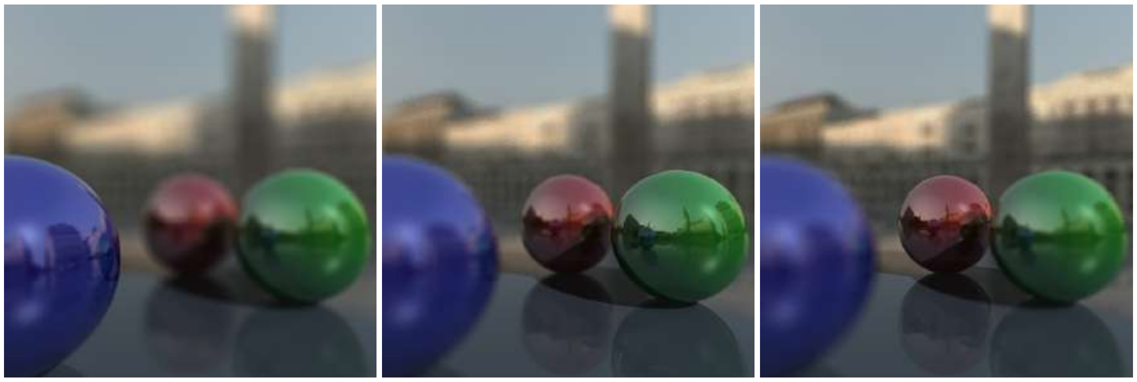
6 Performance

Some attention has also been given to speeding up the raytracer.



(a) Apple without normal map. (b) Apple with normal map.

Figure 7: Effect of a normal map on an apple.



(a) DOF: Close. (b) DOF: Middle. (c) DOF: Far.

Figure 8: Depth of Field.

The loading of object files has been accelerated by using regular expressions. The dragon model can be loaded in just a few seconds.

The bounding volume hierarchy is constructed by repeatedly splitting along the longest axis. All objects within the parent bounding volume are sorted along this axis based on their centroid (the centroid for triangles). The two child bounding volumes are then created as the bounding volume encompassing the first half of the objects and the bounding volume encompassing the second half. Both volumes have an equal number of elements (except when the number of elements in the parent is odd). These bounding volumes can overlap. This approach yielded the best results. The entire dragon model can be decomposed into this bounding volume hierarchy in a fraction of a second, and it can be raytraced with shadows and two point light sources in about 3 seconds. This was achieved on an i5 4690K CPU. A false color image of the bunny using this bounding volume hierarchy is shown in Figure 9.

The raytracer uses a ShadeRec class to pass results (hit point, normal, UV, etc.) from the ray tracing process. However, creating and deleting a large number of objects causes significant overhead. Therefore, ShadeRec objects are reused as much as possible. The methods that perform

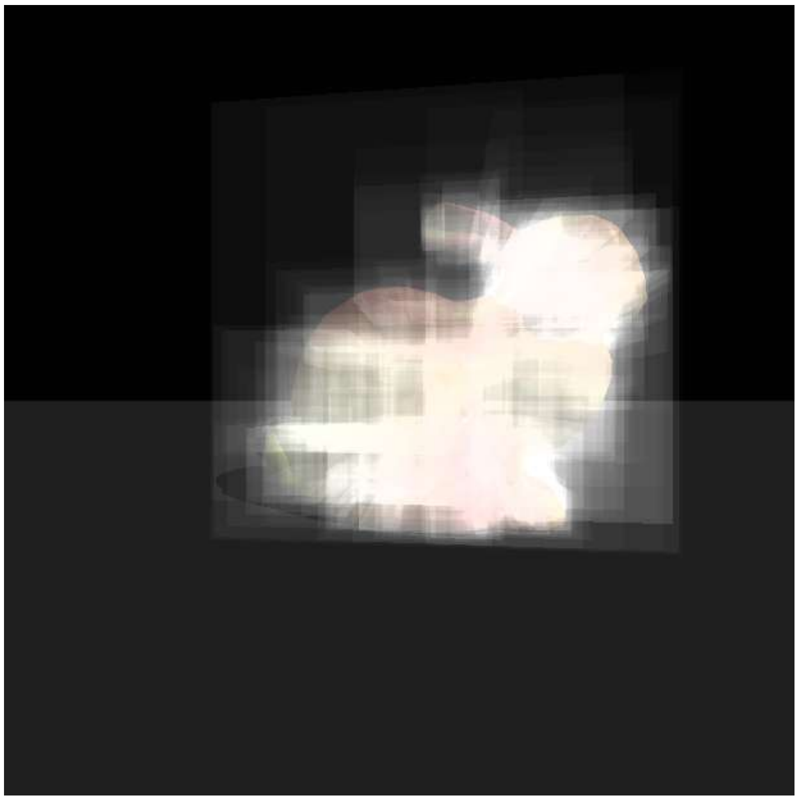


Figure 9: False color image of the Stanford bunny. The whiter the color, the more bounding volume intersections it took to render the corresponding pixel.

the heavy lifting do not return anything and have a ShadeRec as an argument. The values of that ShadeRec are then overwritten. This is in contrast to returning ShadeRec objects from method to method. As a result, the same ShadeRec can often be reused between calls to these functions, minimizing the creation of new objects.

7 Other Rendered Figures

7.1 Glass Bunny with Surface Light Source

Figure 10 shows a glass bunny with a refractive index of 1.5 and a green color. Note that there are no caustics (global illumination is required for that). This figure also demonstrates Beer's Law: as the ray travels further through the bunny, its color becomes redder.

7.2 Fractal in Reflection

Consider Figure 11 rendered with this raytracer. The fractal is derived from four reflecting spheres that are touching each other. They are positioned at the vertices of a regular tetrahedron. The camera looks up from below between the three bottom spheres, straight toward the fourth top sphere. Instead of the correct physical reflection where the color diminishes with each reflection, the color is slightly amplified after each reflection in this rendering. Additionally, when the ray intersects a sphere, a bit of green is added for the upper sphere (in the image), red for the left sphere, and blue for the right sphere. Such images can be used to determine the number of times a ray reflects before diverging from between the spheres. It appears that rays reflect many times for points located on Sierpinski triangle-like shapes. The colors indicate which spheres are involved

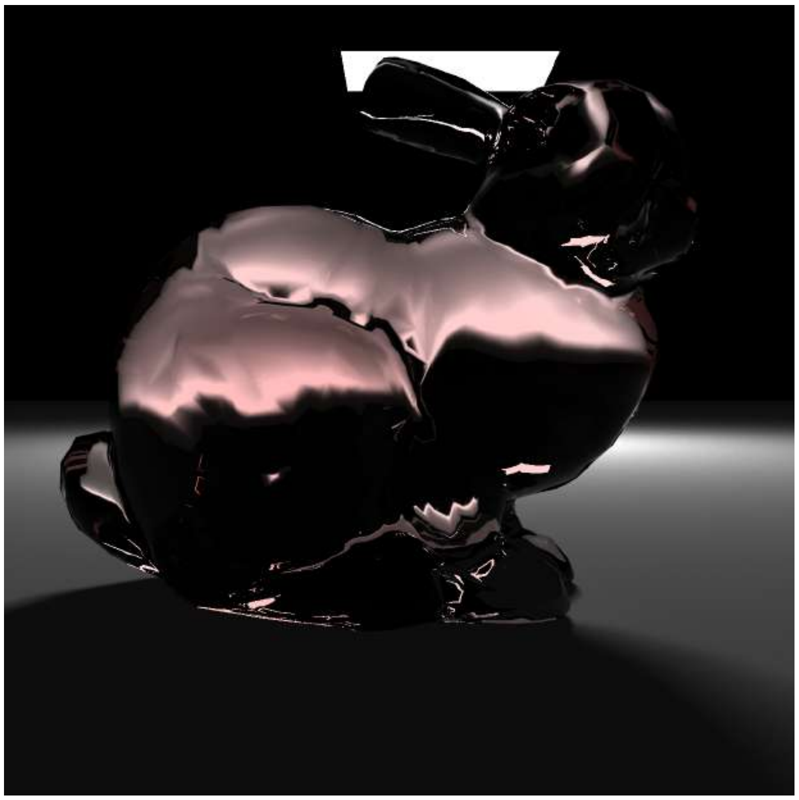


Figure 10: Glass bunny.

in these reflections. For example, bright yellow indicates many reflections with the green and red spheres. The back sphere does not contribute any color when it is involved in reflections.

7.3 Bugs in Refraction Implementation

Due to some bugs in the implementation of correct refraction, the following images were generated:

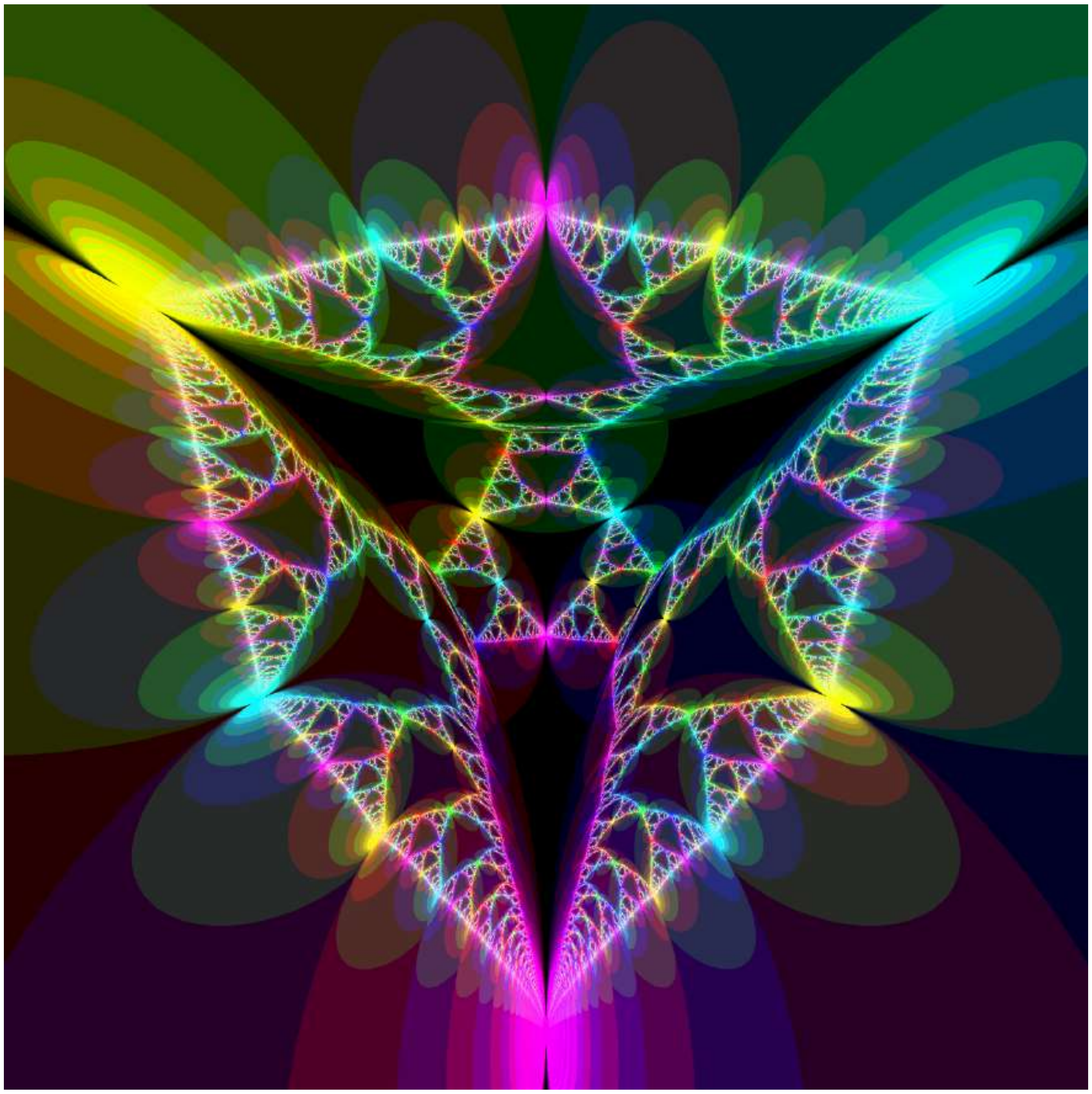


Figure 11: Fractal measuring the number of reflections until divergence from a structure consisting of four touching spheres.

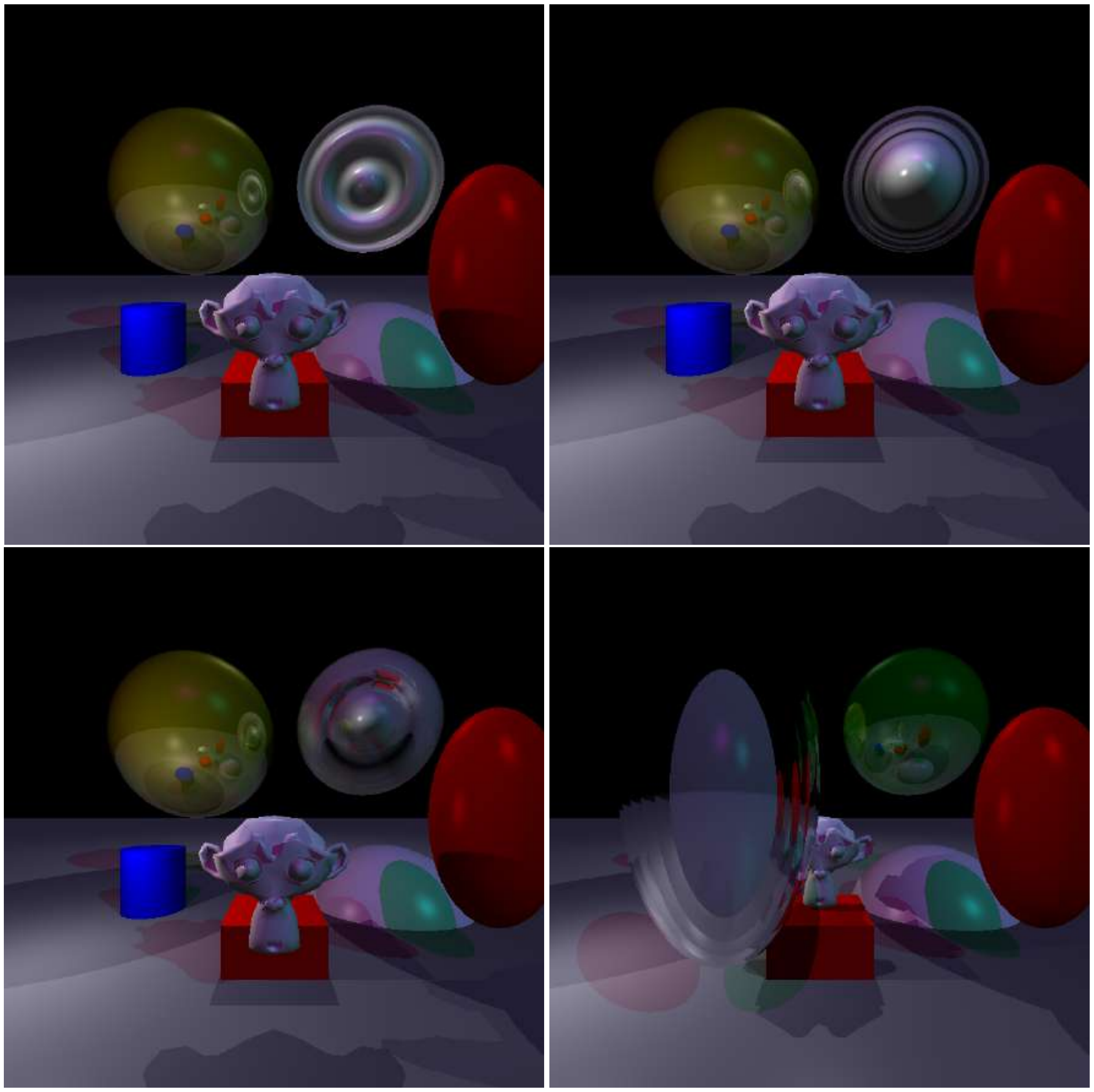


Figure 12: Bugs in the implementation of refraction.