

Ray-tracing with selective visibility

Andrei Sherstyuk
Department of Computer Science
Monash University

August 15, 1997

Abstract

We propose a simple yet effective extension of ray-tracing algorithm that allows selective visibility of objects for specific rays. Typically, rays are classified as pixel, shadow, reflected and transmitted. We suggest adding a visibility mask to material descriptions, which controls how these rays interact with the material. This addition can be incorporated seamlessly into any ray-tracer and may help to generate visually interesting images.

1 Introduction

The general ray-tracing paradigm incorporates hidden surface removal, reflection, refraction and shadow computation in a single framework. The price of this generality are the simplistic models of light propagation and light/surface interactions, which leaves out many interesting phenomena such as dispersion, caustics, iridescence and others.

Ray-tracing employs the linear model of light propagation – a ray represents an infinitesimally thin path of light energy that obeys the laws of geometrical optics. Also, rays normally carry the information about intensity of the light and its spectral characteristics. Therefore, rays have both geometrical and photometrical properties. Similarly, the environment is modeled geometrically by a number of primitives and photometrically via specifications of materials and a lighting model.

The overall success of a particular implementation of a ray-tracer largely depends on the versatility of these two models. The more details about the processes of light/surface interaction are accounted for, the more realistic resulting images are. It is usually implied that visual realism is a measure of image quality and the ultimate goal of computer-

generated imagery.

We suggest shifting the focus of attention from pursuing realism in ray-tracing to exploring the hidden capabilities of the existing ray-tracing concept. Working with light and objects via their models, we are able to change the rules of their interactions in a way that best suits the current task. These rules must not mimic the actual laws of physics, therefore, ray-tracing may be used as a flexible modeling tool.

We introduce a visibility mask and visibility weights as means of additional control over intersection and shading processes, respectively. Also, we provide a number of examples that demonstrate the modeling capability of this technique and discuss some situations when it may be used.

2 Visibility mask

Typically, the geometrical part of ray-object interactions is controlled by a simple ‘hit-or-miss’ scheme. All rays, regardless of their type, attempt to intersect all objects in the database. In the special case of shadow rays, the first hit with a non-transparent object terminates the traversal of the database. Even if penumbrae are supported, the ‘hit-or-miss’ strategy still holds for each individual shadow ray. We propose to include a third alternative, ‘ignore’, into this scheme. Rays are allowed to ignore certain objects, if the material of these objects is said to be undetectable by rays of specific types.

In most ray-tracers, there are four types of rays: pixel, shadow, reflected and transmitted. Therefore, every material used in the model may be made selectively visible for each type of ray. A four-wise visibility mask yields 16 possible ways the objects may behave during finding the location

of the surface and during shading. In the following examples, zeros and ones represent visibility for pixel, shadow, reflected and transmitted rays, left to right:

Mask	Object appearance
1111	Conventional fully visible objects
0000	Completely undetectable objects
1000	Objects can be seen normally, but they do not cast shadows, cannot be reflected and cannot be seen through a glass wall
0100	Objects are invisible, but they cast shadows
0010	Objects can be seen only if reflected in a mirror. No shadows.
0001	Objects can be seen only if put behind a transparent substance. No shadows.

In the beginning of the ray-object intersection test, the ray type is checked against the visibility mask of the material associated with the object. If the result is positive, the test proceeds normally, otherwise the object is rejected. The object is assumed to have a reference to some material. This may not be true for complex objects, like grids, lists and octrees. In this case, the visibility test is postponed till the the reference to the material is available.

3 Visibility weights

The visibility of objects can also be controlled by real-valued weights. These weights may be arranged in some array $W[4]$, one value for each ray type. It is convenient, but not necessary, to assume that weights range from 0 (fully invisible) to 1 (completely visible).

Unlike bitwise masks, visibility weights $W[i]$ must be used during photometrical resolution only (shading). The ray-object intersection test cannot ignore the object, because non-zero partial visibility requires calculation of the surface location.

For color-bringing rays (pixel, reflected and transmitted), the shading routine scales the diffuse, reflected and transmitted color components by $V = W[i]$, where i is the type of the incident ray. Then one more ray is fired from the hit-point in the direction of the incident ray. The amount of color the new ray brings is scaled by $T = 1 - V$ and added to the resulting color. The pseudo-code for this process is shown in Figure 1.

```

determine diffuse color Cd
if material reflects light
    find reflected color Cr
fi
if material transmits light
    find transmitted color Ct
fi
/* check visibility of the material */
V = material->W[incident_ray->type]
if V <> 1
    /* partial visibility: trace a new ray */
    new_ray.origin = hit_point
    new_ray.direction = incident_ray.direction
    new_color = trace(new_ray)
    T = 1 - V
    total_color = V * (Cd+Cr+Ct) + T * new_color
else
    /* full visibility */
    total_color = Cd + Cr + Ct;
fi

```

Figure 1: Algorithm for shading partially visible objects.

This type of scaling allows smooth blending of the object with the background (including all objects that are located behind), because the light energy is conserved along the incident ray. Using different scaling values T may produce interesting effects, for example, light amplification for $T > 1 - V$. The new ray behaves very much like a transmitted ray and allows to see through objects with partial visibility.

For shadow rays, visibility weights result in transparent shadows. That may slow down the shadow calculations, because intersection with an opaque object does not terminate the process. Instead, the ray continues traversing the database, until accumulated opacity reaches some critical value, usually 1.

4 Results

To demonstrate the above technique we computed several pictures. In Figure 2 two types of visibility masks were used. The word SOFT is modeled with a material, undetectable by reflected rays, so it is not reflected in the mirror plane. Similarly, the word DRINK is invisible for pixel rays. Both objects occupy the same physical space, but because their visibility masks have no common bits, they do not interfere with each other. The glass and the mirror plane are modeled with unmasked materials.

The visibility masks of two human characters in Figure 3 are complementary: materials of the male character are detectable by all rays except

shadow rays while the female object is detectable by shadow rays only. Again, both objects overlap in physical space.

Figure 4 shows four frames from animated sequence "Mirage in Louvre", computed with visibility weights. The material of Nefertiti's bust becomes gradually invisible for reflected rays. Notice that the mirrors do not change their properties – the rest of the scene is reflected normally.

5 Conclusions

We have presented a method to create visually interesting effects by manipulating the visibility of objects in four different planes, corresponding to four ray types. This method requires little computational effort and allows greater control over object appearance. We have shown that ray-tracing as a rendering technique has its own modeling capabilities to create and render unusual optical phenomena.

Besides modeling optical effects, selective visibility may be used in the following situations:

- Rendering light sources made of transparent materials, such as an electric bulb. Normally, ray-tracing of such objects requires a computationally expensive support for shadows of transparent objects. Using selective visibility mask, these objects may be rendered more efficiently – it suffices to specify that glass material is undetectable by shadow rays.
- Selective shadows. In certain situations it is useful to disable shadows, completely or partially. This can be achieved by using 'non-shadowed' light sources, which affect the whole scene. Using visibility masks or weights provides local control over shadows of individual objects.
- Non-significant reflections/refractions. Sometimes it is possible to estimate which parts of the scene are unimportant for secondary rays. For example, dark objects are almost invisible in dim mirrors, so these objects may be explicitly made undetectable by reflected rays. Another example is described in "The Invisible Man" by H. G. Wells: glass can not be seen when put in the water. In other words, materials of close optical density and high trans-

parency may be safely ignored by transmitted rays.

6 Acknowledgements

Thanks to Peter Tischer for for a critical reading and discussion. The model of a male character in Figure 3 was provided by ViewPoint Datalabs and the model of the female character was provided by Tomwoof. The author would also like to acknowledge the contribution of an anonymous referee who made a number of useful suggestions which helped to improve the presentation of the paper.



Figure 2: Ray-tracing 'wrong' reflections with visibility masks.



Figure 3: 'Wrong' shadows (two extended light sources).

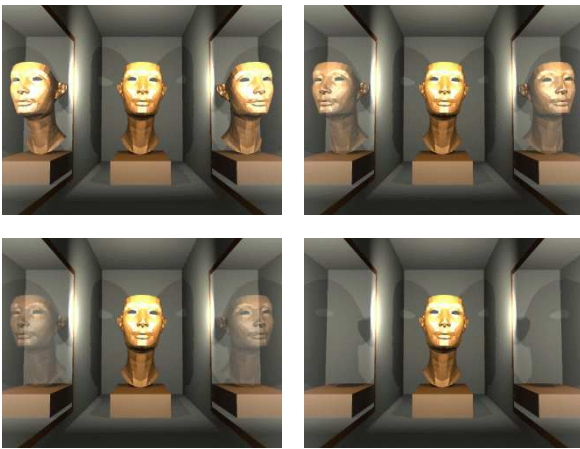


Figure 4: "Mirage in Louvre". Visibility weights, left to right: 1, 0.5, 0.25, 0.