Real-Time Glossy Reflections on Planar Surfaces

Anton L. Fuhrmann* Robert F. Tobler† Stefan Maierhofer‡
VRVis Research Center

Abstract

In this paper we concentrate on an optical effect well-suited for architectural or game-related renderings: glossy reflections on planar surfaces. We demonstrate a physically reasonable approximation of glossy reflections at slightly imperfect reflecting surfaces — e.g. most floors — that can be implemented at minimal performance penalty when compared to perfect reflections. An evaluation of our implementation using only standard graphic API functions in OpenGL 1.2 concludes the paper.

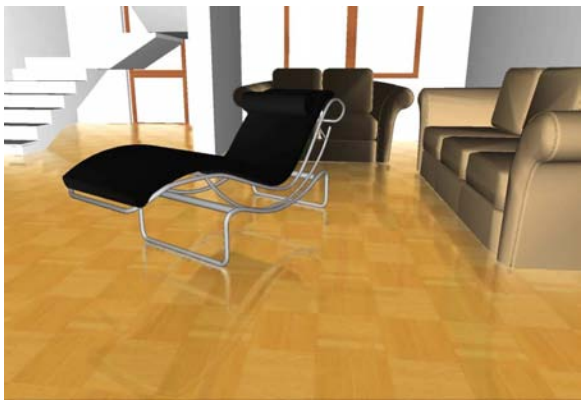

Figure 1: Glossy reflections in an architectural scene.

1 Introduction and Related Work

Almost all effects formerly associated with ray-tracing algorithms can now be reproduced or at least approximated in real-time using hardware-accelerated z-buffers: texturing, transparency, local light models, point light shadows, and reflections in plane mirrors. Another class of effects similar to those produced by distributed ray-tracing can be implemented by merging multiple z-buffer renderings: motion blur, depth of field, and glossy reflections [Diefenbach 1996] can be approximated by multi-pass rendering techniques. In this paper we concentrate on an optical effect well-suited for enhancing architectural renderings or game engines: glossy reflections at planar surfaces (see figure 1 for an example).

*e-mail: fuhrmann@VRVis.at
†e-mail: rft@VRVis.at
‡e-mail: maierhofer@VRVis.at

Glossy reflections generally occur on most polished floors and help — together with shadows — in visually "anchoring" objects to the floor. The main visual difference between glossy and perfect reflections lies in the visual subtlety of glossy materials. A perfectly reflecting surface overloads the image with unnecessary detail. Even when overlayed with a floor texture, the resulting sharp and detailed view in the mirror distracts from the actual scene.

As mentioned above, glossy reflections can easily be implemented using path tracing [Kajiya 1986]. Although some advances in real-time raytracing have recently been made [Wald et al. 2001] [Purcell et al. 2002], exact glossy reflections require sampling the BRDF and shooting multiple reflection rays, bringing even these algorithms out of the realm of real-time.

Diefenbach [Diefenbach 1996] demonstrates how glossy reflections can be implemented using the OpenGL accumulation buffer in combination with a high number of rendering passes which also precludes its use in real-time rendering. A fast approximation of glossy reflections on curved surfaces can be produced by pre-filtering environment maps [Kautz and McCool 2000], but this approach cannot be extended to planar surfaces and inherits the well-known artifacts of environment maps. Bastos et.al. [Bastos et al. 1999] have demonstrated how to apply image-based techniques in combination with 2D convolutions to produce glossy reflections on flat surfaces. Their approach produces high-quality images, but at considerable computational load on the CPU. Furthermore they deliver constant rendering time per reflector, but at additional memory cost depending on the reflector area, which can be considerable in the case of building floors. Their approximation of the blurring of the glossy reflection consists — similar to ours — of just an image space convolution of a perfect reflection.

In designing our approach, we considered the following criteria important:

- adjustable approximation of blurring and angle dependent fading of reflected image
- minimal runtime and memory overhead opposed to perfect reflections
- easy integration into existing rendering solutions

We want to apply our algorithm on a wide range of commodity graphics hardware, so reliance on vendor-dependent solutions or high-end hardware was not an option. Specifically, we want our method to work on any 3D accelerator produced in the last few years.

2 Approximated Effects

Generating glossy reflections needs to account for two major effects: the scattering of light due to the microfacet distribution of the surface and the increase of reflectivity near grazing angles due to the Fresnel function.

The effect of the microfacet distribution

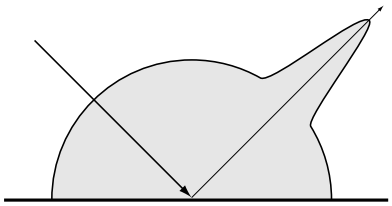

Figure 2: A partially glossy BRDF.

In figure 2 the distribution of reflected light of a partially glossy BRDF is depicted. It consists of a part that is indepedent of the incoming direction based on the diffuse or lambertion component of the BRDF, and a reflection lobe based on the non-diffuse component of the BRDF. In order to mimic the effect of glossy or non-diffuse reflection it is necessary to analyse the effect of this reflection lobe. This reflection lobe is caused by the microfacet distribution of the surface. For a given maximum deviation angle alpha of the microfacet normals from the average surface normal, the resulting reflected light cone is bounded by a cone with its apex in the reflection point and an opening angle of 2α as shown in figure 3.

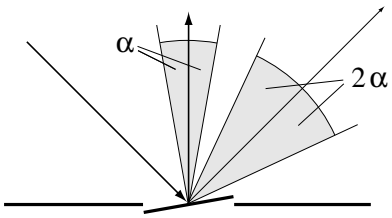

Figure 3: The width of the reflection cone based on the maximal deviation angle α of the microfacet normals.

Based on this reflection cone the reflection scenario on a floor surface is depicted in figure 4. As we are designing an image space algorithm, we consider all viewing rays that affect a single image location – a single pixel.

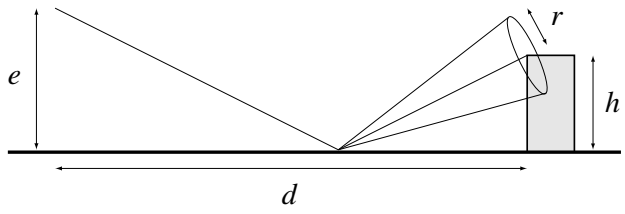

Figure 4: The reflection cone for all viewing rays that affect a single image location. e ... eye height, h ... object height, d ... object distance.

In order to emulate glossy reflection by image space filtering, we need to estimate the size of the filter kernel that we will use. By calculating the relative size of the reflection cone with respect to the viewing distance, r/d, we get an approximation of the relative size of the filter kernel with respect to the projection plane distance. Note that we use the normal distance to an assumed vertical projection plane. This is an assumption that is valid for typical architectural scenes, but of course it is limiting the applicability of the algorithm somewhat.

Figure 5 shows the necessary filter radius r/d for an observer with his eyes 1.65 m above a reflecting surface, a maximal normal deviation of the microfacets of α of 5 degrees, and object heights from 0.5 to 2.5 m in 0.5m steps. These ranges of parameters were chosen to cover typical example architectural scenes. Starting at a distance of about 5 meters from the observer, the filter radius for an object of a given height remains approximately constant.

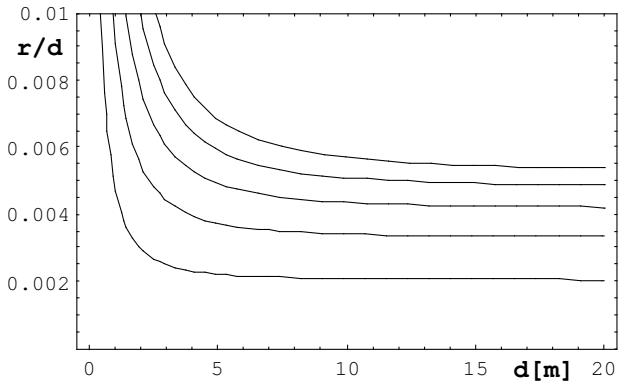

Figure 5: The necessary filter radius r/d depending on the observer distance d for objects of heights h = 0.5, 1.0, 1.5, 2.0, and 2.5 (bottom to top) . Eye height e=1.65m, maximal microfacet deviation α = 0.5 degrees .

The effect of the Fresnel function

The fresnel function describes the energy balance between reflected and refracted light on a transparent surface based on the relative refraction index of the transparent surface at the interface. Glossy materials like polished floors and paints are subject to this effect, as their surface is covered by a thin layer of transparent wax or similar material that is transparent and accounts for the high reflectivity of the material. In figure 6 the effect of the fresnel function as a function of object distance is shown for various object heights.

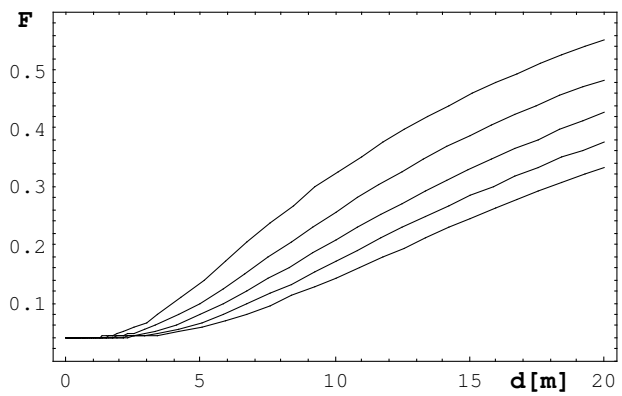

Figure 6: The fresnel factor F for reflected objects of heights h = 0.5, 1.0, 1.5, 2.0, and 2.5 (top to bottom) depending on the observer distance d. Eye height e = 1.65m., the refraction index η = 1.5.

3 Our Approach

We have to realize two different phenomena: the blurring of the reflected image, depending on the distance from the reflector, and the angle-dependent attenuation of the reflection. Our approach is based on rendering the mirrored geometry into an off-screen buffer, and compositing this buffer into the screen buffer.

Fading

We have to simulate two independent fading effects: the fading into a uniform background color caused by the increased blurriness of the reflected scene with growing distance from the reflector (an effect, that Diefenbach [Diefenbach 1996] approximates using hardware fog) and the angle-dependend attenuation of the reflection.

One possible method of implementing these effects would be by implementing them as pixel shaders [Akenine-Moeller and Haines 2002]. However pixel shaders are of limited size, and by using different methods for achieving these effects, we free the pixel shaders for additional uses.

Our algorithm discretizes the effect by simulating the distance and angle dependence using attenuation by partially transparent planes. After rendering the reflected part of the scene, these attenuation planes are rendered into the reflection in a back-to front fashion, resulting in an increasing attenuation effect dependent on the number of attenuation planes between the geometry and the viewer. Aligning these planes parallel to the reflector, results in fading of the reflection with increasing distance from the reflecting surface (see figure 13).

Additionally we want to modulate the intensity of the reflection with varying incident angle. As already described, polished surface materials exhibit higher reflectance at grazing angles, an effect that could be realized similar to the method above by rendering an attenuation texture on top of the reflection.

Such a texture would implement a one-dimensional mapping of angle to attenuation. The textures scale depends on the distance of the viewpoint to the reflector. Theoretically the texture would have to cover the reflector up to the horizon while exhibiting a reasonable high resolution to avoid artifacts in the near field.

We avoid these problems by combining the attenuation factors in one method. By rendering the attenuation planes introduced above not parallel to the reflector but at increasing angles to it (see figure 7), we get an attenuation depending on the incident angle measured parallel to the viewing direction. Note that the parametrization of the fresnel-function in figure 6 has been chosen as a function of distance and height of the reflected object. Thus we can compute a least squares optimization for the position of each attenuation plane by numerically computing all object heigths which result in the same fresnel factor. The attenuation planes serve as a useable approximation, as the functions for different object heights attain the same fresnel factor at nearly equidistant points in the graph.

To correctly implement angle-dependent attenuation one would have to render not transparent planes but coaxial cone segments. In view of the already applied approximations this is an unnecessary refinement.

Blurring

Like Bastos et al. [Bastos et al. 1999], we approximate the the blurred reflection by performing a 2D-convolution with a space-invariant kernel over the perfectly reflected image. The estimated filter kernel sizes have been shown in figure 5. A first approximation for the kernel size is to assume a constant blurring of the reflection between planes at varying height that are parallel to the reflective surface. However comparing an implementation with such different slabs of varying filter size (figure 8, left) with a much simpler implementation of constant filter size combined with attenuation at the slab boundaries (figure 8, right) shows that not much can be gained by varying filter sizes. Although the visible detail for parts of the scene is noticeably higher when fading is used, these details are masked by lower contrast, and often by a texture of the glossily reflecting surface (see figure 12). The attenuation rate and filter size necessary for simulating materials of different glossiness has been determined experimentally.

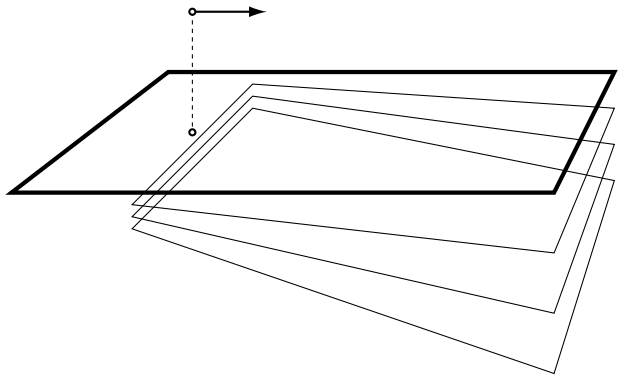

Figure 7: The tilted planes that seperate regions of constant attenuation for a given viewing direction.

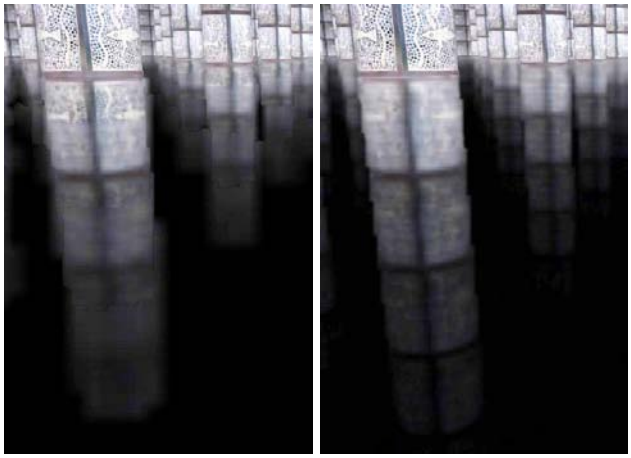

Figure 8: Blurred (left) versus faded (right) slabs.

Combined approximation of fading and blurring

As both effects are now handled by attenuation planes, the location of these plane is chosen using a least squares approximation that is based on the combination of the fuctions depicted in figure 5 and figure 6. The number of attenuation planes can be chosen to avoid any visible steps in the attenuation of the mirrored geometry. In our experiments we chose 64 or 128 planes to avoid artifacts.

The resulting image (figure 14) now combines both attenuation factors in a plausible way.

4 Implementation

For a start, we implemented the standard OpenGL planar reflection technique using stencils planes [Blythe et al. 1999]. Here the scene is rendered normally, except for the mirror, which is rendered into the stencil planes so that it produces a mask for the next rendering steps. Normally one continues with rendering the reflected scene so that it is only visible inside this mask.

We have to perform additional steps to implement our algorithm:

- render reflected scene
- render attenuation planes (cones) depending on viewpoint
- convolve scene with blur filter
- copy blurred scene image into the mirror mask

The location of the attenuation planes have been placed to approximate the reflection characteristic of the surface according to the fresnel function (see figure 6) and the necessary filter widths (see figure 5). This approximation can be performed in advance, and is based on the simplified numerical approximation according to the parametrization given in these two figures. The number of attenuation planes has been experimentally chosen to avoid visual artifacts.

The most time-consuming steps here are the additional rendering pass for the reflected scene and the convolution. At the moment we just render the complete scene geometry in the reflection without additional optimizations (see future work).

Convolution Speed-Up

The convolution can be performed using the techniques presented by Hadwiger et al. [Hadwiger et al. 2001], with a performance dependent on the size of the filter kernel. Most materials exhibit a relatively strong blurring, thereby making a large convolution kernel necessary, leading to fill-rate limitations in the process of convolving the reflected scene. We avoid this by rendering the reflected scene at a lower resolution than the final image. Although this minification is not the same as filtering, hardware antialiasing and texture filtering increases the quality of the rendered image, approaching a correctly filtered version. We now only need a small convolution kernel (we use 4x4) to blur the reflection. To render the reflected scene at a lower resolution one can just use the back-buffer, or — subject to availability — pbuffers, render-to-texture or a similar approach.

Combining Reflection and Scene

Now we combine the reflected, blurred scene with the primary scene by copying an appropriately scaled version of the reflection into the mirror mask. Normally a floor texture has been rendered on the mirror, making it necessary to combine the two elements — floor texture and reflected image — using a blending operation, which results in adding the intensity of the reflected image to the floor texture.

5 Results and Evaluation

Our synthetic test scene (figure 10 on colour plate) depicts a cathedral like formation consisting of columns, arches, and a floor, each textured. It consists of 40250 textured triangles. The scene was rendered at a resolution of 1280x1024 on two different platforms. See figure 9 for results.

	without mirror	**perfect mirror**	**glossy 64 planes**	**glossy 128 planes**
triangles	40250	80500	80500	80500
P4, 2.4 Ghz Geforce 4 TI 4600	43 fps	31 fps	30 fps	30 fps
P3 Mob. 1.0 Ghz Geforce 2go	20 fps	13 fps	11 fps	11 fps

Figure 9: Resulting rendering speed in frames per second with different numbers of attenuation planes.

As can be seen in the table in figure 9 , our algorithm introduces no significant overhead to the standard perfect reflection implementation. Although a simple implementation of a mirroring surface by rendering the complete geometry as a mirrored world can potentially increase the rendering time by a factor of two, occlusion culling methods together with viewing portals can be applied. Using these techniques the overall cost can be reduced to be on the order of the visible detail in the final rendered image.

Note that our algorithm does not need to use pixel shaders for generating this effect. We do not see this as a major disadvantage of our algorithm, as the pixel shader programs with their strict size limits are still available for other effects.

In figures 10 to 14 (on the colour plate) we compare different renderings of the same scene and viewpoint. For better visibility the floor texture has been omitted. The figures show renderings without reflection (fig. 10), perfect reflection (fig. 11), attenuation planes parallel to the floor (fig. 13, no fresnel term) and tilted attenuation planes (fig. 14).

Figures 13 and 14 show the significance of the fresnel term. The comparison between figure 11 and figure 14 demonstrates the differences between conventional reflections and our glossy reflection: the blurred reflection enhances the visual quality of a glossy material and the fading reduces unnecessary reflections in the near field, while maintaining strong reflections at grazing angles.

6 Future Work

Since the glossy reflected image normally does not exhibit a large amount of detail, it would be possible to use a lower detailed representation of the scene to render the reflection. This could significantly improve the speed of scenes containing multiple mirrors.

A further speedup could be introduced by clipping all parts of the reflected scene which are attenuated under a certain level. This clipping could not be applied if light sources are to be rendered in the scene, since these would normally show up in the reflection even when highly attenuated. Rendering the lights would imply rendering all potentially occluding geometry between them and the mirror, even when the geometry itself would always be attenuated to invisibility.

Our algorithm can potentially be extended to cover translucent, scattering surfaces. In combination with soft shadows this would allow plausible renderings of frosted glass partitions.

7 Acknowledgements

References

Akenine-Moeller, T., and Haines, E. 2002. *Real-Time Rendering, 2nd ed.* A.K. Peters Ltd.

Bastos, R., Hoff, K., Wynn, W., and Lastra, A. 1999. Increased photorealism for interactive architectural walkthroughs. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, 183–190.

Blythe, D., Grantham, B., McReynolds, T., and Nelson, S. R. 1999. Advanced Graphics Programming Techniques Using OpenGL. *SIGGRAPH '99 Course*.

Diefenbach, P. J. 1996. *Pipeline Rendering: Interaction and Realism through Hardware-Based Multi-Pass Rendering*. PhD thesis.

Hadwiger, M., Theussl, T., Hauser, H., and Gröller, E. 2001. Hardware-accelerated high-quality reconstruction on PC hardware. In *Proceedings of the Vision Modeling and Visualization Conference 2001 (VMV-01)*, Aka GmbH, Berlin, 105–112.

Kajiya, J. T. 1986. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, vol. 20, 143–150.

Kautz, J., and McCool, M. D. 2000. Approximation of glossy reflection with prefiltered environment maps. In *Graphics Interface*, 119–126.

Purcell, T. J., Buck, I., Mark, W. R., and Hanrahan, P. 2002. Ray tracing on programmable graphics hardware. In *SIGGRAPH 2002 Conference Proceedings*, ACM Press/ACM SIGGRAPH, J. Hughes, Ed., Annual Conference Series, 703–712.

Wald, I., Slusallek, P., Benthin, C., and Wagner, M. 2001. Interactive rendering with coherent ray tracing. In *EG 2001 Proceedings*, A. Chalmers and T.-M. Rhyne, Eds., vol. 20(3) of *Computer Graphics Forum*. Blackwell Publishing, 153–164.

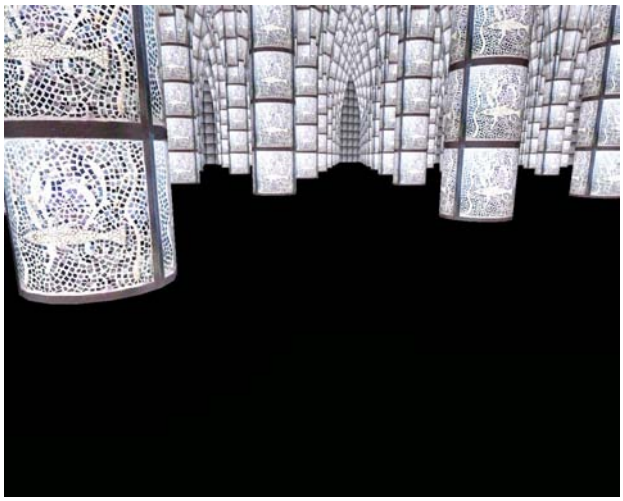

Figure 10: The test scene without reflections.

Figure 11: The test scene with perfect reflections.

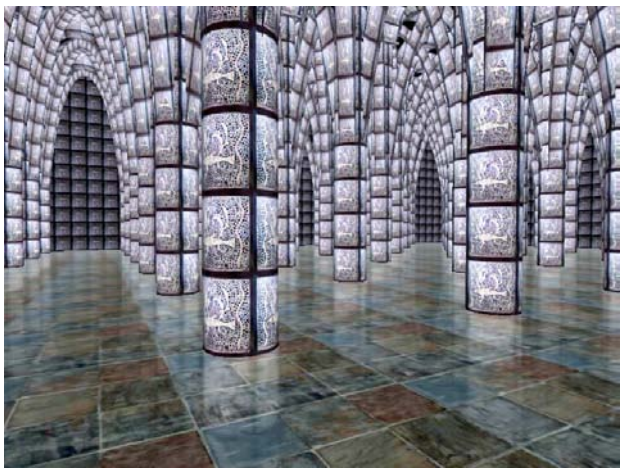

Figure 12: A textured floor masks approximations in the glossy reflection.

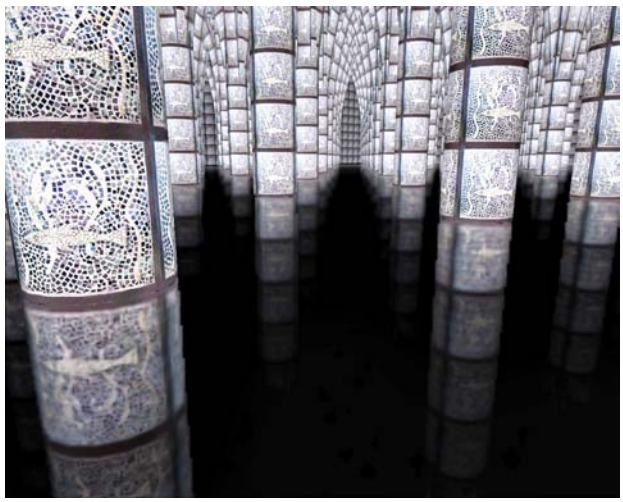

Figure 13: The test scene with attenuation planes parallel to the reflection plane.

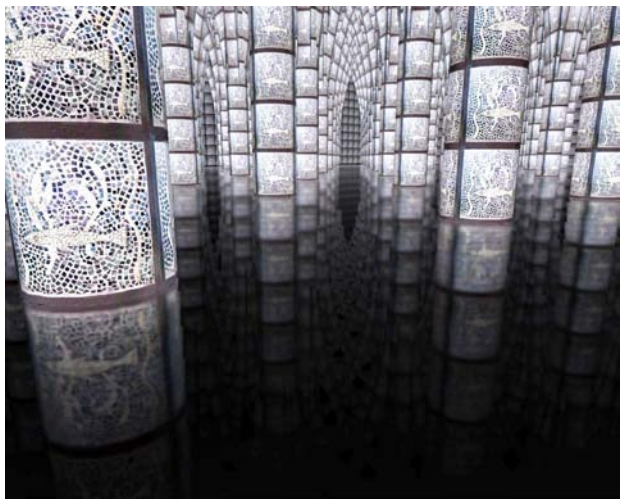

Figure 14: The test scene with tilted attenuation planes.

Figure 15: Another view of our architectural sample scene.