Master Informatique

Deep Shadow Maps
IG3D Course Sorbonne University

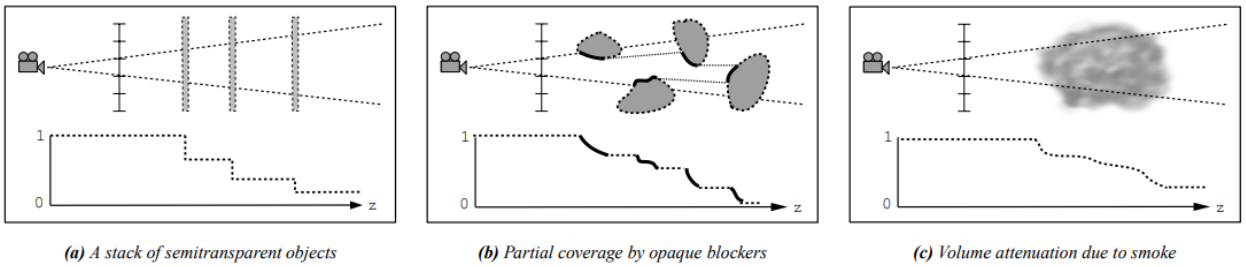
Étudiant: Basci Onur
N°: 21309649

Contents

1 Introduction

Deep shadow maps is an efficient shadow rendering technique, especially useful for producing fast and high-quality shadows for primitives such as hair, fur, and volumetric objects. In comparison to traditional shadow maps, deep shadow maps are smaller and significantly faster to access, making them ideal for achieving high-quality shadows. Additionally, unlike traditional shadow maps, they support partially transparent surfaces, volumetric objects, and motion blur.

A deep shadow map essentially consists of a rectangular array of pixels positioned adjacent to the light source. Each pixel contains a visibility function, which represents a beam of light originating from the shadow camera's position and passing through the corresponding pixel. The value of this function at a particular depth indicates the fraction of the beam's initial power. The visibility function may diminish due to total or partial obstruction by objects, intersection with semi-transparent or volumetric objects.



(a) *A stack of semitransparent objects* **(b)** *Partial coverage by opaque blockers* **(c)** *Volume attenuation due to smoke*

Figure 1: Visibility Function representation

To represent this beam, we randomly select a set of points within each pixel and compute the transmittance function for each point. The transmittance function determines the amount of light transmitted at a given depth. It is defined by a set of depth values obtained from surface intersections and volumetric sampling. The visibility function is then calculated as a weighted sum of these transmittance functions.

Given that a visibility function comprises numerous depth points from transmittance functions, it can result in a large number of vertices. To address this issue, the author suggests a compression method aimed at reducing the vertex count while preserving the essential characteristics of the functions. This compression technique not only minimizes the number of vertices but also significantly speeds up visibility function lookups.

2 Methods

Deep shadow maps can be added to any kind of renderer. I choose to implement it on my ray tracer that I coded during the lab session with some adjustments. I added different type of objects such as plane, rectangle and cylinder. I also added volumetric objects. The volumetric objects are defined by density fields generated using Perlin noise with slight adjustments (absolute, max, etc.). Rendering volumetric objects involves utilizing the ray marching technique. For calculating light absorption, I sample the object along the camera's ray direction and compute the transmittance for each sample. However, for calculating in-scattering, I utilize the visibility function instead of sampling the volume in the light's direction. This approach reduces complexity, as there's no need for a second sampling towards the light direction. While the results are satisfactory, there's a noticeable gradient as we move further from the light source.

This occurs because we calculate a weighted average of neighboring visibility functions during lookup. To mitigate this artifact, one can reduce the filter size, though this may introduce color discontinuities. The code is configurable according to the users choice.

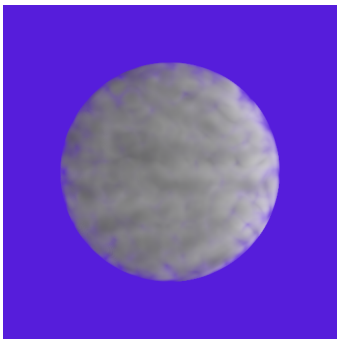


Figure 2: Shadows with ray marching

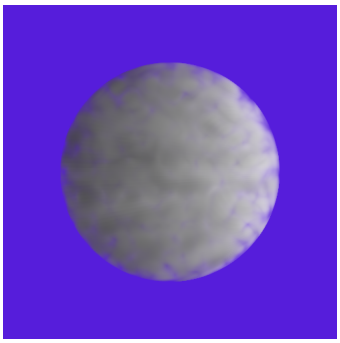


Figure 3: Self shadowing with averaging Filter

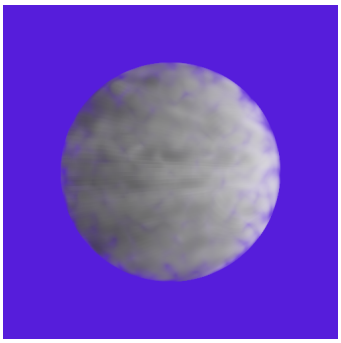


Figure 4: Self Shadowing without averaging filter

Shadowing without filtering can cause rough edges and discontinuities in the shadows. Therefore, filtering the visibility function is necessary. I've experimented with multiple kernels, such as uniform and Gaussian filters. While both of them solve the edge problem, the Gaussian filter seems to yield better results by preserving the characteristics of the shadow.

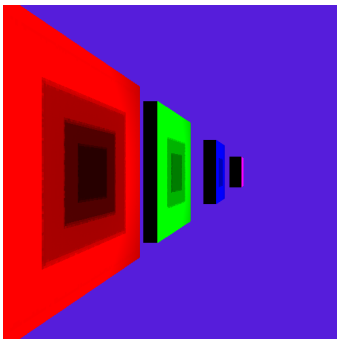


Figure 5: No Filtering

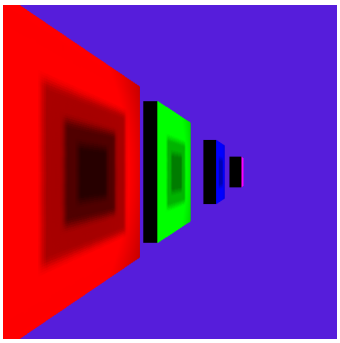


Figure 6: Uniform Kernel

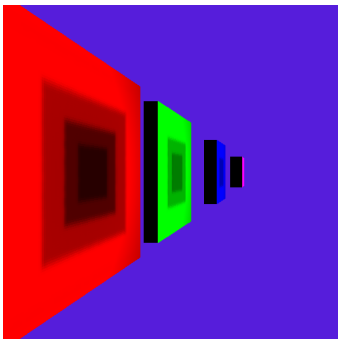


Figure 7: Gaussian Kenrel

3 Results

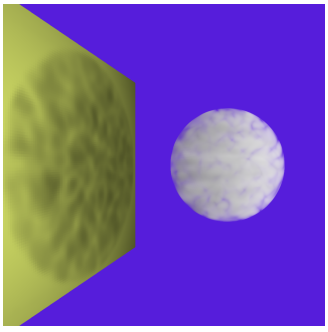


Figure 8: Shadow of a volumetric object on a solid object

Here is a result showcasing the shadows cast by a volumetric object onto a solid object. This image was generated using a ray tracing camera resolution of 500×500, with a deep shadow map resolution of 128×128 and 8 samples per pixel. Notice the noisy shadows produced by the density field of the volumetric object. Additionally, you can observe self-shadowing on the volumetric object itself. Rendering this scene takes approximately 11 seconds

I also compared the efficiency of compression in terms of both time and space. For this comparison, I created a scene with randomly placed multiple thin cylinder objects. To conduct the test, I calculated the execution time and the average size of the visibility function for deep shadows with varying numbers of objects in the scene. The compression was performed with a 5% margin of error.

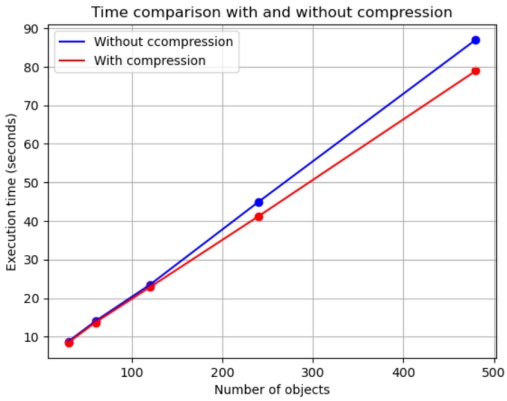


Figure 9: Time comparison

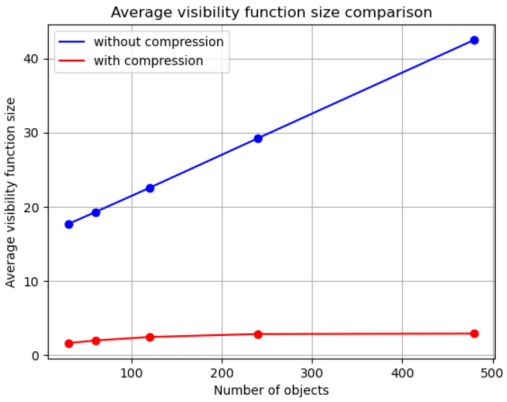


Figure 10: Space Comparison

As depicted in Figure 9, compression does not contribute additional time to the overall rendering process. Instead, it compensates for the time spent on compression by facilitating faster lookups. Notably, as the number of objects increases, the rendering time becomes marginally faster. Figure 10 illustrates a significant disparity in the size of visibility functions. Without compression, the size appears to increase linearly, whereas with compression, it seems to increase logarithmically.

Here are the renders obtained with 240 objects, with and without compression. The compressed version yields quite similar results to the uncompressed one.

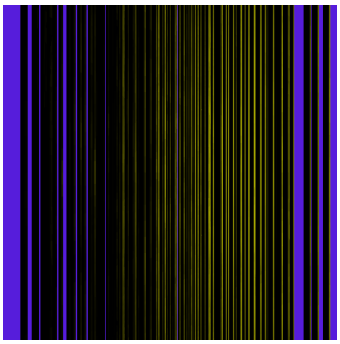


Figure 11: With compression

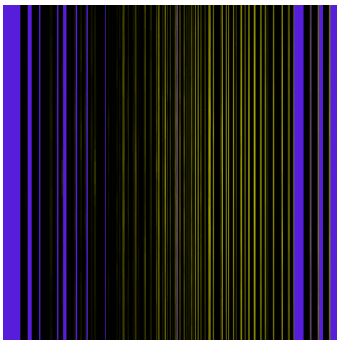


Figure 12: Without compression

4 Limitation

One of the main uses of deep shadow maps is for rendering hair and fur. However, rendering hair is challenging due to the large number of hair strands and the complex structure of each strand. A hair strand is typically represented with quadratic or cubic Bézier curves. However, there is no direct formula to calculate the intersection point with a Bézier curve. Additionally, the likelihood of a ray intersecting a curve is minimal, so adding volume to the curve further complicates the problem. While a geometric representation using polygons could offer a solution, it would significantly increase computation time. To simplify the problem and address partial coverage of opaque objects, I opted for thin cylinders, where intersection points can be analytically calculated (see Figures 11 and 12).

5 Conclusion and Future Work

Deep shadow maps are a technique used for creating realistic shadows, particularly effective for rendering shadows of semitransparent, volumetric, and numerous small objects. With compression, they are also memory-efficient and enable fast rendering times compared to other shadow rendering methods, such as traditional shadow maps.

As future work, I am interested in exploring motion-blurred shadows and mipmapping, both of which are supported by deep shadow maps. Additionally, I would like to work on methods to render efficiently hair in a ray tracer. Lastly, I am intrigued by the possibility of implementing this method on GPU to investigate if real-time results can be achieved with deep shadow maps.