OVERVIEW OF THE SHADOW ALGORITHMS

Bisera Trpcevska
European Univercity
Skopje, Macedonia

Vladimir Nasteski
European Univercity
Skopje, Macedonia

ABSTRACT

Realistic artificially rendered shadow has an important role in the field of computer graphics. Hence, it is one of the most attractive research topics in computer graphics. In this paper we present an overview of several types of shadow rendering algorithms. We present their characteristics, their development and their goals. The goal of this paper is to summarize the advantages of every shadow rendering method, so that the appropriate approach for rendering shadows can be chosen according to the specific requirements. As a practical example, or a software suggestion, we suggest the 3D design software packet Autodesk Maya, as software that uses a combination of algorithms for quality generating a realistic shadow.

Key words: shadow; computer graphics; shadow volumes; projection shadows; ray-tracing; 3D

I. INTRODUCTION

Shadow provides the users with visual clues about information on the scene, shapes and geometries, relative positions among objects and complexity of shadow occluders. In practice, rendering shadow is complex since there are a lot of constrains involved in creating it, such as planar or non-planar receivers, single or multiple light sources, hard-shadow or soft shadows. Thus, modeling and rendering shadows is always one of the most challenging issues of the research fields of computer graphics.
In the recent years the development of the technology of computer graphics hardware has been highly increased, and people continuously look for more real shadow drawing. With this development, many problems have emerged that are related with the shadow rendering. Today, many computer graphic scientists hard work to improve some of the existing algorithms and to solve issues in some practical applications.
The main ingredient for generating a computer image is rendering of shadows. The shadow is increasing the level of realism of any spatial object on the scene. For real-time rendering, variants of the shadow mapping algorithm [1] and the shadow volume algorithm [2] are among the most popular techniques.
The shadow volume algorithm divides the virtual world in two areas: ones that is in shadow and others that are not. Shadow volume algorithm is used in 3D computer graphics that adds shadows for rendering a scene. The computer game Doom 3 is an example of using the technique of the algorithm. The shadow volume technique requires the creation of shadow geometry, which can depend on the implementation of the CPU.
In this paper we will show different techniques of many shadow algorithms, including their advantages and disadvantages. Also, we will define which of them faster generate shadow.
This paper is organized as follows. In section II we discuss about the shadow volume algorithm and its characteristics. Section III presents practical algorithms which are implemented in many 3D graphical softwares for generating shadows, including shadow mappings, soft shadow volumes, stochastic ray tracing and many miscellaneous techniques and algorithms. In Section IV we propose the 3D design software packet Autodesk Maya, in which we understood the technique and the combination of algorithms for generating a realistic shadow. At the end, we give a conclusion of the analysis that we made about the algorithms.

II. SHADOW VOLUME ALGORITHM

The shadow volume algorithm is a popular technique for real-time shadow generation using graphics hardware. In the 3D space the shadow volume algorithm projects a shadow of the objects at the back of the objects, because of the opaque of the objects. Its position is determined with relationship between the light and the occluders. It can render correct shadows on any surface of the objects and also can cast shadows on itself. The time it costs depends on the complexity of the scene.
The first shadow volumes algorithm is presented by Crow [2] and first implemented by Heidmann [3]. Their algorithm is divided in two steps. First, a shadow volume is constructed by extending silhouette of the occluder as viewed from the light source along the direction of the light source. Then, after all the volumes are calculated, it is determining which object or which part of an object is inside any volume to determine whether they are in shadow. The second process can be executed using the so called stencil buffer [4]. If we want to determine whether an object A in the scene is visible, we follow a ray through a pixel until the ray hits the object, and then we calculate the number of faces of the shadow volume it crosses. While the ray is on its way to this object, we increase the counter in the stencil buffer each time it goes into a shadow volume and decrement the counter whenever the ray goes out of a shadow volume until it reaches A after crossing every volumes. Finally, if the counter is greater than zero, then it indicates that the number of going into the volumes is larger than the number of going out of them.

Hence, A is located in shadow, or A is located outside the shadow. To implement the shadow volumes algorithm, first render the scene from the light point of view to define all the shadow volumes and then determine whether the surface of an object is in shadow according to the number of polygons of shadow volumes a ray across from the viewer to this object. (Figure 1)

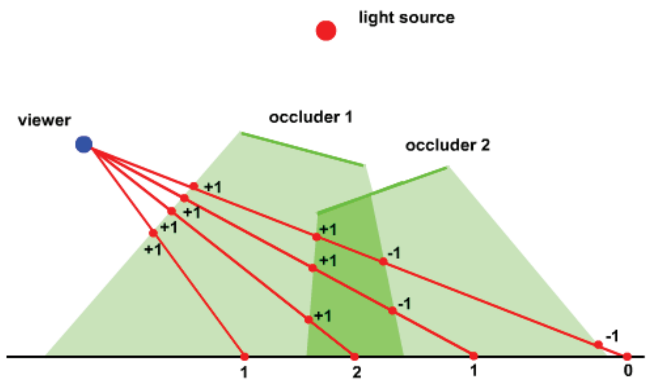

Figure 1: Implementation of the shadow volume algorithm

The shadow rendering algorithms can be divided into different categories according to different classification standards. There are two types of shadow algorithms for light: off-line shadow rendering algorithms and real-time shadow rendering algorithms. Also, this algorithm can be divided to shadow drawing of nontransparent objects and transparent objects, or shadow drawing of polygonal models and complex models such as clouds, fur or grass. There are different types of light sources and they result in hard-shadow rendering algorithms and soft-shadow rendering algorithms.
Many techniques are described in many papers [5], [6] which are used in the process of implementation, where shadow rendering algorithms are separate into scan-line methods, shadow volumes algorithms, shadow mapping algorithms, ray-tracing shadow algorithms and Z-buffer shadow algorithms.

III. PROJECTION SHADOW ALGORITHMS

In this chapter we define how the projection shadows algorithm casts shadows on planar surfaces through projection transformation. Projection shadows algorithm is the earliest shadow rendering algorithm. It is similar to the process of projecting objects onto the 2D screen from the view point. The projection shadow algorithms have simple idea and can be quickly computed, but it is limited only to casting shadows onto planar surfaces rather than general surface on earth.
The idea behind the projection shadow algorithms is very simple, and can be quickly computed. Still, there are some weaknesses for this technique. For instance, the method anti-shadow cannot be avoided if the light source is below the top object and the method fake shadow also cannot be avoided if the object is below the shadow receiver. This problem has been solved by Heckert and Herf [7] proposed a method to eliminate the anti-shadows and fake shadows, in which they used accumulation buffer to produce soft shadows. The cost of this algorithm is not expensive but the effect is less realistic. Another method from Gooch et al. [8] is to approximate the soft-shadows in the scene with a spherical light source to gain speed. They average the projections by moving up and down the receiving plane's location. This method has the advantage that the shadows created are concentric, which generally looks better and so requires fewer samples. But the shadow rendered in this method is usually larger than the object so that it will induce distortion. In reality, if an area light is larger than an object in a certain range of distance, the object will have a smaller or nonexistent umbra region.
In addition, we present the different techniques, improvements, divisions of the projection shadow algorithm, including shadow mapping, soft shadow volumes and stochastic ray tracing. Also, there are many other miscellaneous techniques and algorithms.
One of them is the radiosity algorithm. It computes the diffuse global illumination solution that also includes soft shadows from direct lighting.
The image-based soft shadow algorithm uses layered attenuation maps for fast approximations.
A coherent ray tracer is used for generating higher-quality images. The parallel ray tracer is used for a single sample per pixel and soft edged objects. This algorithm is very fast, and it is not physically based one.
EPI (edge-and-point image) is used to provide sparse sampling of a scene. The silhouette edges and the shadow boundaries are stored as edges in the EPI. With this technique the image can be reconstructed using specialized filter. For area light sources, both umbral and penumbral edges are detected and stored as edges in the EPI.

A. *Shadow Mapping*

Shadow mapping is an effective real-time soft shadow method. It has attracted much attention in the field of real computer graphics. Williams [9] proposes that the shadows can be quickly generated on random objects using the common Z-buffer-based algorithm. His idea is based on rendering the scene from the light's point of view. The distance from the light source to the closest objects can be obtained as the z-depth in the Z-buffer. When some objects are rendered by a 3D graphics card, the depth of a generated pixel (z coordinate) is stored in a buffer (the z-buffer or depth buffer). This buffer is usually arranged as a two-dimensional array (x-y) with one element for each screen pixel. If another object of the scene must be rendered in the same pixel, the graphics card compares the two depths and chooses the one closer to the observer. The chosen depth is then saved to the z-buffer, replacing the old one. The entire content which is inside the Z-buffer is called the *shadow map*. Then, we render the scene normally from the view point. For each pixel of the scene, the geometrical position of the object can be well known. If the distance between this object and the light is greater than the distance stored in the shadow map, then the object is in shadow. Otherwise, it is illuminated.

Segal et al. [10] are presenting an idea by using texture mapping technology. As first, from the light's point of view the scene is rendered, and the depth values are loaded into a texture which is also called the *shadow map*. Then, this texture is projected onto the scene as drawn from the eye's point of view using only ambient lighting. This determines which region is in shadow. If the pixel's depth is equal to the corresponding value in the shadow map after transformation from the coordinate system of the light source into the coordinate system of the viewer, then the α value of this pixel is set to 1, which indicates that the pixel is not in shadow. Otherwise, the α value is set to 0 indicating that the pixel is in shadow. Finally, the whole scene is rendered with the global illumination. The final color of each pixel is the color from the ambient pass plus the color from the full rendering pass multiplied byα. If α is 0, the pixel color is taken from the ambient rendering pass, and if α is 1, then the pixel color is that of a normal rendering pass. It should be noticed that when the shadow map is generated, required is the depth buffer, that is, lighting, texturing, and the writing of color values into the color buffer can be turned off. Moreover, as long as the position of the light source does not change, the shadow map can be repeatedly used because the shadows are view-independent.

There is a method called silhouette maps, in which a shadow depth map is augmented by storing the locations of points on the geometric silhouette. With this the visual quality is improved. This algorithm is based on the observation that shadow maps perform well in most areas of the image, but suffer from objectionable aliasing in the regions near shadow boundaries. Algorithm has two stages. In the first stage, the scene is rendered from the point of view of the light to generate a depth map and a silhouette map. In the second stage, the scene is rendered from the viewer's perspective and shadow determination is made. Since the depth map is the same as in traditional shadow map implementations, there is only discuss the generation of the silhouette map and its use in shadow determination. The purpose of the silhouette map is to provide information on the location of the shadow boundary. The shadow boundary can be approximated by a series of line segments.

To determine if a point in the scene is in shadow, we first project it into light space. After that, we compare the current fragment depth with the four closest shadow depth samples. If they all agree that the object is either lit or shadowed, this region does not have a silhouette boundary going through it and we shade the fragment accordingly. This is similar to a standard shadow map depth test. If the depth tests disagree, a shadow boundary must pass through this texel. In this case, we use the silhouette map to approximate the correct shadow edge. Shadow edges separate regions of light and shadow and also separate regions where the depth tests pass and fail.

B. *Soft Shadow volumes*

As defined in computer graphics, ray tracing [11] is a method for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. When the light which is emitted from a light source, it is being reflected or refracted by geometric objects with certain material properties, and finally appearing at the observer's eye, typically on the film of a camera. In order to minimize computational costs and render just the visible parts of a scene, the algorithm works backwards: it starts at the camera and sends a ray in the direction of the current view. If this ray hits an object, then its material is evaluated, and the final color information is returned and stored. In case the material is reflective or refractive, further rays are sent into the scene to compute these contributions to the final color. This process is repeated recursively until a certain traversal depth is reached. This makes ray tracing wide spreading to many applications where the image can be rendered slowly ahead of time, such as real-time applications like computer games where speed is critical. Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering, and chromatic aberration.

C. *Stochastic ray tracing*

Stochastic ray tracing algorithms compute shadows by sampling an area light source using shadow rays. Calculating the overall light propagation within a scene, for short global illumination is a very difficult problem. With a standard ray tracing algorithm, this is a very time consuming task, since a huge number of rays have to be shot. For this reason, the radiosity method was invented. The main idea of the method is to store illumination values on the surfaces of the objects, as the light is propagated starting at the light sources.

For reducing the number of shadow rays by utilizing image-space coherence, Hart et al. [12] proposed a two-pass algorithm. In the first pass, a small number of blocker-light pairs are stored for each pixel. In the second pass, the stored blockers are used for computing the visible parts of the light source. Nevertheless, correct results cannot be guaranteed, because the algorithm does not consider all occluding polygons.

IV. PRACTICAL IMPLEMENTATION CONSIDERATIONS

Over the review on the many techniques and algorithms, we've concluded that the 3D design software packet Autodesk Maya [13] offers a dedicated "hardware renderer" which uses the computing power of the graphics card. This hardware renderer requires a workstation-class graphics accelerator; but according to many experiments, even with some of such expensive graphics cards it does not support features such as shadows. Furthermore, this hardware renderer is not intended to be used interactively, but acts only as a speedy replacement of the usual software renderer. The absence of a hardware renderer does not mean that typical 3D design software does not support vertex and pixel shaders. Rather, in several such software packages, real-time shaders can be used in the *interactive* rendering. This is intended as a preview to assist game designers, who thus are able to build models and materials with perfect visual feedback. Also, as a software solution for rendering, or creating shadows, Autodesk Maya is using a combination of the method shadow map and ray tracing algorithm. Furthermore, we show a faster soft shadow volume algorithm that will create faster, and more realistic shadows.

A pseudocode for the algorithm is given in addition. In a preprocessing stage (Construct Hemicube [14]) we extract silhouette edge information from the entire scene and store it into a static acceleration structure. Then, for each point p to be shaded, the visibility between p and a set of samples on the surface of the light source is computed (Shadow Query). The light samples are the targets of the shadow ray queries that the algorithm effectively replaces.

```
Construct Hemicube
 for each edge E
  if E is a potential silhouette edge from the light source
   W ← wedge planes of E
   add W into all hemicube cells that overlap W
 end if
end for
```

```
Shadow Query(point p)
clear depth complexity counters of light samples
p`← p projected onto the surface of the hemicube
LW ← list of wedges from hemicube at p`
for each wedge W in LW
  E ← edge associated with W
   if p is inside W and E is a silhouette edge from p
    project  E onto the surface of the light source
    update depth complexity counters of light samples
   end if
end for
cast a shadow ray to a light sample with lowest depth comp.
if ray is blocked
return all light samples are hidden
Else
return light samples with lowest depth comp. are visible
end if
```

The pseudocode is a description of the two functions that constitute the shadow algorithm. Construct Hemicube is executed once for every frame and it builds an acceleration structure for finding silhouette edges in shadow queries. Shadow Query is executed for every point to be shaded. It determines which point samples on the light source are visible from the query point.

On Figure 2 we can see a 2D illustration of how the hemicube footprints of penumbra wedges can be used for deciding whether the corresponding silhouette edge may overlap the light source from a given point. Wedge planes are determined according to the light source and the silhouette edge. The intersection of the wedge and the surface of the hemicube is the hemicube footprint of the wedge. To determine if point p may be inside the wedge, the point is projected onto the surface of the hemicube from the center of the light source. If the projected point p` is inside the hemicube footprint of the wedge, point p may be inside the wedge. Otherwise point p is guaranteed to be outside the wedge, and consequently the silhouette edge does not overlap the light source from p.

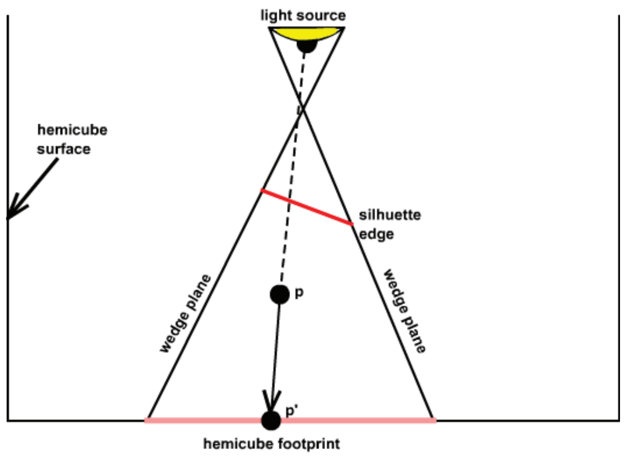

Figure 2: Illustration of the soft shadow algorithm

V. Conclusions

In this paper, we have discussed several important types of shadow rendering algorithms. Projection shadows algorithm is simple to implement without complicated techniques requirements, but confined to the scenes with planar receivers. On the other hand, shadow volumes algorithm is a geometry-based method which can render accurate shadows.

We analyzed the idea of shadowing mapping algorithm. It can cast shadows on arbitrary objects easily and quickly. But also, there are various techniques which are presented to improve it.

Also, in this paper, we've presented series of algorithms that improve the shadow mapped images. Many of them are simple, run in real-time and work very well in hardware.

We've analysed those algorithms for one purpose. We consider that by a combination of many techniques or algorithms, so we can create an algorithm that can be fast and efficient.

As a future work, we'll propose a new algorithm that is a synthesis of almost all shadow algorithms through the years, which will generate shadow very fast and very effective, with a great quality behind.

Acknowledgment

Our work was supported by the Faculty of Informatics of the European University. Thanks to Jovan Pehcevski for the comments and discussions during the research and for full support that he gave us during the work on this paper. Also, we would like to thank Ivan Chorbev for the encouragement he gave us to participate on this conference for informatics.

REFERENCES

[1] Williams L.: Casting Curved Shadows on Curved Surfaces. In Computer Graphics (Proceedings of ACM SIGGRAPH 78) (August 1978), ACM, pp. 270–274.
[2] Crow F.: Shadow Algorithms for Computer Graphics In Computer Graphics (Proceedings of ACM SIGGRAPH 77) (July 1977), ACM, pp. 242–248.
[3] Heidmann T. Real shadows, real time [M]. Iris Universe, Silicon Graphics Inc. 1991, 18: 23-31.
[4] Web link:
http://msdn.microsoft.com/en-us/library/bb976074.aspx
[5] Timo Aila, Tomas A.-Moller. A Hierarchical Shadow Volume Algorithm. Helsinki University of Technology. The Eurographics Association 2004.
[6] Nan LIU, Ming-Yong PANG. A Survey of Shadow Rendering Algorithms: Projection Shadows and Shadow Volumes Department of Educational Technology Nanjing Normal University, NNU. Second International Workshop on Computer Science and Engineering. 2009
[7] Heckbert P S, Herf M. Simulating Soft Shadows with Graphics Hardware [R]. Technical Report CMU-CS-97-104, Carnegie Mellon University, 1997.
[8] Gooch B, Sloan P J, et al. Interactive Technical llustration. In: Proceedings 1999 Symposium on Interactive 3D Graphics [C], 1999, 31-38.
[9] L. Williams, "Casting Curved Shadows on Curved Surfaces", *Computer Graphics (SIGGRAPH '78 Proceedings)*, 1978, 12(3): 270-274.
[10] M. Segal, C. Korobkin, R. V. Widenfelt, et al., "Fast Shadows and Lighting Effects Using Texture Mapping". *Computer Graphics (SIGGRAPH '92 Proceedings)*, 1992, 26(2): 249-252.
[11] A.J. van der Ploeg, "Interactive Ray Tracing", 2006
[12] HarT, D., Dutr´e , P., and Greenberg, D. P. 1999. Direct Illumination with Lazy Visibility Evaluation. In Proceedings of ACM SIGGRAPH 99, ACM Press, 147–154.
[13] Web link:
Alias(R) Maya(R) 5.0. www.alias.com, 2003
[14] Hemicube – 3D computer graphics – Wikipedia http://en.wikipedia.org/wiki/Hemicube_computer_graphics