# *Sphere and Cone Composite Realtime Shadows in OpenGL*

Aldina Pljaskovic, Dzenan Avdic,
Department of Technical Sciences
State University of Novi Pazar
Novi Pazar, Serbia
apljaskovic@np.ac.rs

Petar Spalevic
Faculty of Technical Science
University of Pristina
Kosovska Mirovica,Serbia
petar.spalevic@pr.ac.rs

Dejan Rancic
Faculty of Electrical Engineering
University of Nis
Nis, Serbia
dejan.rancic@elfak.ni.ac.rs

***Abstract*—Adding shadows to 3D objects, in computer graphics, is very important for increasing realism of scene. OpenGL does not support shadowing methods directly. There are several methods for implementation of the shadows, but they are struggling with problems such as high demands for rendering scenes. On the other hand, methods, that do not require a lot of resources for implementation and rendering, do not give realistic shadows. This paper represents one method for efficient real-time shadowing of a couple of objects, and emphasis is on their common characteristics. Since shadowing method for one primitive could be used with little improvement to make shadowing method for other object, this method is called composite method.**

***Keywords—shadows, OpenGL, circle, triangle, sphere, cone, point light source.***

## I. INTRODUCTION

Making a virtual reality using computer graphics is a fundamental part of designing computer games, software for modeling the interior and exterior, software dedicated for various types of simulations etc. Existence of shadows in scene is essential for the improvement of visual perception. Since using shadows depicts relations between objects on scene, users can sense more exactly the distance between two virtual objects. So shadows enhance the 3D impression in order to users can get a better immersive 3D feeling [1].

Physically, shadow consists of two components: the umbra and the penumbra. The umbra is dark component of shadow, and its creation is consequence of the fact that there is the area of a shadowed object that is not visible from any part of the light source. Light sources with their own area can make another component of shadows which is less dark from umbra. It happens when area of a shadowed object that can receive some, but not all, of the light from light source [2]. So, a point source light has no penumbra, since no part of a shadowed object can receive only part of the light [3].

OpenGL does not support shadows directly, it can be used to implement them a number of ways. The methods vary in their difficulty to implement, their performance, and the quality of their results. These qualities depend on two parameters: the complexity of the shadowing object and the complexity of the scene that is being shadowed [4].

Approach described in this paper supports makes formation of shadows easier, taking into account the events that occur when objects are exposed to light in reality. In this example, the light source is a point source of light, and the surface, on which the light rays to form a shadow, is flat.

The aim is to determine the minimum number of characteristic rays coming from the light source, and points out that these rays pass, which are essential for the formation of shadows on the surface. Shadows are created for the four different types of objects (circle, triangle, sphere and cone), but shadowing method made for one type of object is used, with small additions, in order to get the method for creating shadowing method form another type of object. The goal is to find the equations for shading and other primitives. Also, if complex object consists of a number of primitive objects, each of them will cast its own shadows, which will form a shadow of a complex object.

The paper is organized as follows. The first section provides an overview of the existing methods, with emphasis on their advantages and disadvantages. The next chapter presents the mathematical basis for the derivation formula which allowed finding characteristic points for drawing shadows. Then follows the description and view the experiment in which an application is created on the basis of the formula, and the results are displayed. Finally, conclusions and trends in future work are given.

## II. RELATED WORK

There are three main strategies to make shadow in OpenGL: projected shadows, shadow volumes and shadow maps.

Projected shadows are easier for implementation, but those shadows are not real. Idea is projecting the shadowed object to one of the axes. Moving of light source or of the shadowed object does not change shadow form. Trade-off is that this shadow does not behave like real shadow [5].

Shadow Volume is constructed by rays from the light source (Ray Tracing) that intersect the corners in the object that casts a shadow, and then continues on out of the scene. This creates a polygonal surface that contains objects that is shaded or partially shaded. Stencil buffer is used to calculate

This paper was supported by the Ministry of Science and Technology of the Republic of Serbia within the projects TR35026 and III44006.

what part of the objects that is inside the shadow volume. For every pixel in the scene the stencil value will be increased if the border for the shadow volume is intersected on the way into the shadow, and decreased if the border is intersected on the way out. The stencil operation is set to only increase or decrease when the depth test is passed.

The result of this is that all the pixels in the scene with stencil values not equal to zero will identify the parts of an object that is inside the shadow. This method has problems with shadowing complex objects and its implementation has high demands.

Since the shape of the shadow volume is decided by the corners of the object that casts the shadow, it is possible to construct complex shadow volumes [1] [6] [7].

Shadow Maps uses depth buffer and texture projection to create shadow in a scene. The scene is transformed to put the eye in the same position as the light. The objects in the scene are rendered and the depth buffer is updated. This creates shadow map. The shadow map is used for placing textures on the region in the shadow.

This method doesn't depend on finding the silhouette for the objects that are casting shadow. Nor is it necessary to clip the result of the shadow. Every object that can be rendered can throw shadow with shadow maps. But, this method also has disadvantages which are aliased shadow edges, and self-shadowing effects [8].

## III. COMPOSITE SHADOWS

As indicated in the previous section, all the methods for implementing shadows are compromising because of problems rendering the scene. Neither of these methods has not take place in some OpenGL library. Also, they all represent alternative ways for the formation of shadows, and some of them are good in certain circumstances, while under other conditions they show their weaknesses.

The goal of this research is to establish a single library which use will be possible to add a shadow after drawing primitives. As the shadows are realistically drawn for each of the primitives, complex objects consisting of 2D and 3D primitives will have his shadow which will be formed by the shadows of its component objects. How would be solved the problem of rendering? This approach uses the phenomena that occur in real-world in case when scene is lighted by point source of light, which has ability to move along the axis of the co-ordinate system [9].

But the effectiveness of these way of rendering of shadows is in the fact that there is need to calculate a minimum points relevant for drawing shadows. Second circumstance is the use of the fact that there are similarities between shading the primitives. Sphere is made by the rotation of circle around one of axes; cone is made by rotation of triangle. So with finding method for forming the shadow for 2D objects, it is easy to create the shadow for the 3D object.

The paper describes the following analysis. For four objects (circle, triangle, sphere and cone) that are located away from the surface for a distance d, the shadow is on the surface that is created by the point light source with the possibility of movement. The surface is flat. The light source has the ability to move the X and Y axes.

When the light source is closer to the Y axis, the shadow is increased, and vice versa. When the light source moves away from the object in X axis, the shadow objects are also moving in the opposite direction to the movement of the light source.

Due to the large number of options and restrictions for showing 3D system on 2D paper, some less relevant parameters for calculating the are taken as fixed:

- The light source is always above shadowed object ($y_i > y_p + h_p$)
- X and Z coordinates of the object is point 0.
- $x_i$, $y_i$, $z_i$ are the coordinate position of the light source ($z_i = 0$)
- $h_p$ is the height of the object (height of the isosceles triangle and of cone, distance between diameter of circle and sphere and the light source).
- $r$ is the radius of the circle and the sphere.
- $d$ is distance between the object and the surface.
- $x_p$, $y_p$, $z_p$ are coordinates of center point of object (for an isosceles triangle that is the center of the base, for circle it is the circle's center, also for the sphere, and for the cone it is center of its base). ($x_p$, $z_p = 0$).

### A. Circle

When a point light source illuminates the circuit installed parallel to the plane of the surface, there is only the umbra. Two cases are considered: when the light source is just above the circle ($x = x_p = 0$) (Fig.1.a) and when the light source to the left or right of the shadowed object(Fig.1.b).

Important point is the center of the shadow. Also, important information needed for rendering shadows is the size of the radius of the shadow. It is well known that the shadow of the circle in this case is also a circuit. Coordinates of the center of shadow circle will be found using the equation of line that passes through the two points. These points are the center of the light source and the center of shadowed object.

If the source of light just above the object, coordinates of center of shadow are known and they are coordinates of beginning pint (0, 0, 0). If not, the formula for calculating the center of the shadow is given with equation (1).

$$x_s = \frac{-d \cdot x_i}{y_i - d} \tag{1}$$

For the case $d = 0$, $x_s$ is zero, which means that there is no shadow if the object is on the surface and the light source is above it.

Calculating the length of the radius of the shadow is done using the theorem on similarity of triangles. *Figure 1* shows the relevant triangles for calculating the increasing factor for

original radius. The equation for calculating the radius of the shadow is given in (2). From this formula it can be seen that the length of the radius depends on $r$, $y_i$, and $d$.

$$r_1 = \frac{r \cdot y_i}{y_i - d} \tag{2}$$

The shadow of the circle is a circle with its center at the center is ($x_s$, 0, 0), and which radius is $r_1$.

Including the equation$h = y_i - d$, equations 1 and 2 can be expressed as *eq.3* and *eq.4*:

$$x_s = \frac{-d \cdot x_i}{h} \tag{3}$$

$$r_1 = \frac{r \cdot y_i}{h} \tag{4}$$

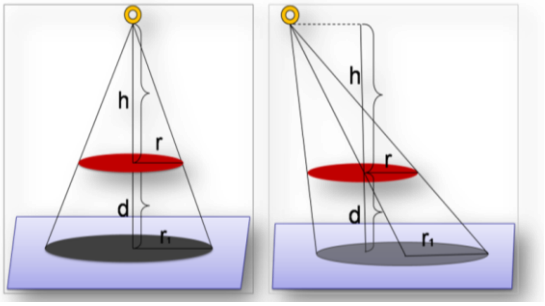


Fig. 1. Shadow of circle on plat surface using point light source: left )$x_i=x_p=0$ right) $x_i \neq x_p$

## B. Triangle

A shadow for an isosceles triangle whose base is parallel to the X axis is line. Characteristic points for drawing the shadows are the points where the beam of light that passes through the vertices of the triangle cut surface.

Since the triangle is upright, and Z coordinates of the light source and the shadowed object is zero, it will be all three points lie on the x-axis and the Z and Y coordinate of them will be 0 (Y coordinate of the shadow is always 0 because the surface is located at the Y = 0).

So these three characteristic points are collinear, i.e. they do not make a triangle but a line. A mitigating circumstance is that we calculate the shadow of the circle. That can be used to facilitate this work on the following way.

There is correspondence between the base of triangle and its center and diameter of circle and its center, because shadow in both cases is a line.

Half of the base is radius. Thus we find the center of the shadow and the length of the new basis (eq. 5 and eq.6).

Now the problem is reduced to the finding of one characteristic point, which is the triangle peak that is common point of two equal edges. It will be calculated according to the eq. 7.

If the base of the triangle parallel to the Z axis, then the characteristic points are not collinear, i.e. the shadow of the peak is on the X-axis, but the basics are propagating along the Z axis.

Those tree points make the shadow which is also a triangle. In this case, coordinates of important points for drawing shadow of triangle are: $P_1(0, 0, z_{s1})$, $P_2(0, 0, z_{s2})$, $P_3(x_{s3}, 0, 0)$, where $z_{s1}$ and $z_{s2}$ are calculating similar as $x_{s1}$ and $x_{s2}$. The two cases: when light source is just above the triangle and when light source is right/left are shown on Figure 2.

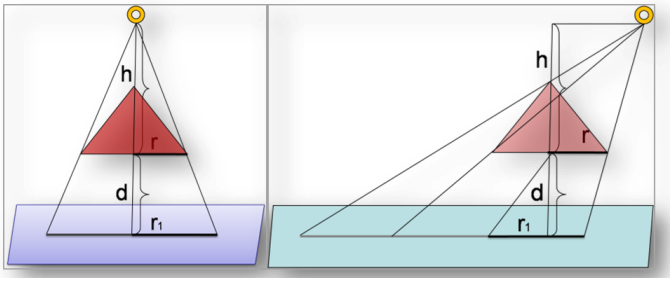


Fig. 2. Shadow of triangle on plat surface using point light source: left) $x_i=x_p=0$; right) $x_i \neq x_p$

If $a$ is base length, where $r=a/2$, coordinates of important points for drawing shadow of triangle are: $P_1(x_{s1}, 0, 0)$, $P_2(x_{s2}, 0, 0)$, $P_3(x_{s3}, 0, 0)$.

$$a_1 = \frac{a \cdot y_i}{y_i - d} \tag{5}$$

$$x_s = \frac{-d \cdot x_i}{y_i - d} \tag{6}$$

$$x_{s1} = x_s - \frac{a_1}{2} \tag{7}$$

$$x_{s2} = x_s + \frac{a_1}{2} \tag{8}$$

$$x_{s3} = \frac{x_i \cdot (h_p + d)}{d + h_p - y_i} \tag{9}$$

## C. Sphere

The sphere is a 3D object that is created by circle rotation around the X axis, but shifted to r on Y axe. Only difference is distance between circle and the surface. By the distance between those two objects we consider distance between center of sphere and the surface. Therefore, the equations for the shadow of the sphere are given before in circle part. The two cases: when light source is just above the sphere and when light source is right/left are shown on Figure 3.

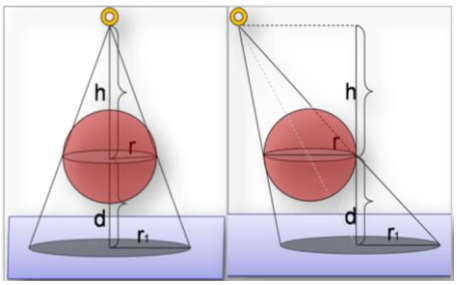


Fig. 3. Shadow of the sphere on plat surface using point light source: left) $x_i=x_p=0$ right) $x_i \neq x_p$

### D. Cone

Cone is a 3D object that is created by rotation of a triangle whose base is parallel to the Z axis. To calculate the necessary shadow points, three characteristic pieces of information are need: projection of top of cone, base center projection and the radius of the shadow. It is obvious that the shadow can be already calculated using 2D shadow elements that are its foundation, and those are the triangle and the circle. Shadow of the base is the same as the shadow of the circle. Top projection is the same as the projection isosceles triangle top which base is parallel to the Z axis, which has been already analyzed. The shadow is an union of circle and triangle. If the light source is just above the cone, shadow consists just of circle (Fig. 4). Calculation for the shadow of circle is already given above, and triangle part of the shadow is determined by $P_1(0, 0, z_{s1})$, $P_2(0, 0, z_{s2})$, $P_3(x_{s3}, 0, 0)$, where $z_{s1}$ and $z_{s2}$ are calculating similar as $x_{s1}$ and $x_{s2}$.

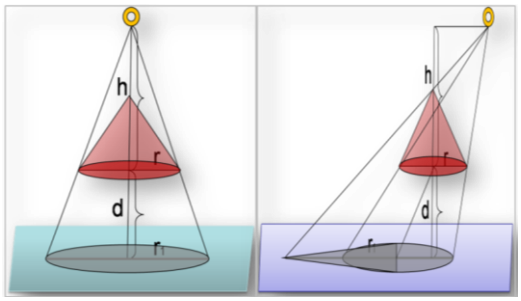


Fig. 4. Shadow of the cone on plat surface using point light source: left) $x_i=x_p=0$ right) $x_i \neq x_p$

## IV. Experiments

To confirm the validity of practical formulas and check the speed and the reality drawing shadows created a C ++ application that uses OpenGL. The application takes into account the above limitations. The application has following features:

- Changing of shadowed object (keys K, L, T, U)
- Changing the position of camera (keys Y, X, C, A, S, D)
- Changing the position of the point light source (keys V, B, N, M).

The results for each element are given with following screenshots.

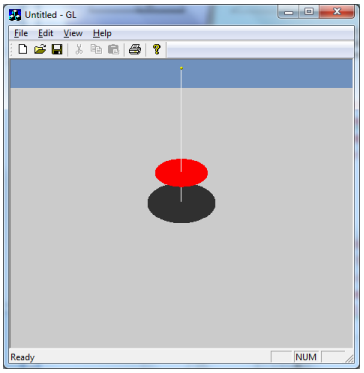


Fig. 5. Implementation of shadows in OpenGL C++ application - circle

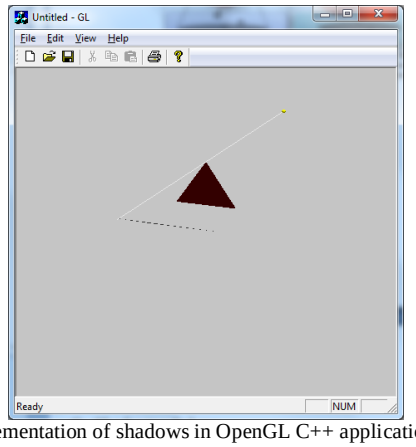


Fig. 6. Implementation of shadows in OpenGL C++ application - triangle

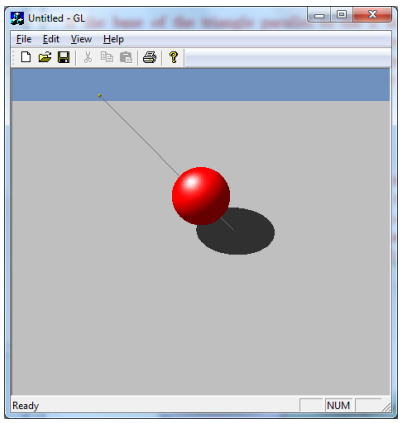


Fig. 7. Implementation of shadows in OpenGL C++ application - sphere

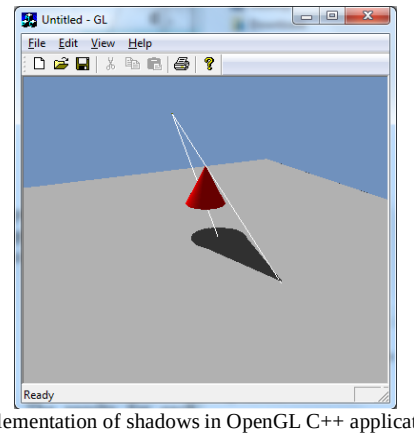


Fig. 8. Implementation of shadows in OpenGL C++ application - cone

## V. Conclusions and Future Work

Methods for obtaining the shadows of different objects need not be independent. It should be used to make the calculation of the shadows of objects easier.

Shadow of a complex object is the union of the shadows of its constituent elements. By projecting all points on the surface slows drawing of the shadows.

Therefore, it is enough to find the minimum number of characteristic points of shadow, and the shadow and reality will not be disturbed, and rendering of the shadow will be easy and cheap.

In the future, the focus will be on the finding methods for shading remaining primitives that are commonly used. And other cases will be analyzed: the case when a shaded object can move, what happens if there are more sources of light, or more objects in the scene, or if the surface is not flat.

After analyzing all of the conditions in which the shaded object can be found, the output should be a unique and efficient method of forming real OpenGL shadows.

## REFERENCES


[1] M. Haller, S. Drab and W.Hartmann, "A realtime shadow approach for an Augmented Reality application using shadow volumes" VRST '03 Proceedings of the ACM symposium on Virtual reality software and technology pp. 56 – 65

[2] T. McReynolds, D. Blythe, "Advanced Graphics Programming Using OpenGL" The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, ISBN: 1-55860-659-9, 2005.

[3] C. Wyman and C. Hansen. „Penumbra maps: Approximate soft shadows in real-time." Proceedings of Eurographics Symposium on Rendering 2003, pp 202–207, June 2003.

[4] E. Angel, D. Shreiner, "Teaching a Shader-Based Introduction to Computer Graphics" Computer Graphics and Applications, IEEE, Vol.31, No.2, pp. 9-13, 2011.

[5] N. Liu and M. PANG, "A Survey of Shadow Rendering Algorithms: Projection Shadows and Shadow Volumes" , Second International Workshop on Computer Science and Engineering, 978-0-7695-3881-5/09 -2009 IEEEDOI 10.1109/WCSE.2009.107 pp: 488 – 492.2009

[6] H. Kolivand, M.S. Sunar, N.M. Jusoh and F. Olufemi, "Real-Time Shadow Using a Combination of Stencil and the Z-Buffer" The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.3, pp 27-38, August 2011.

[7] M. D. McCool, "Shadow Volume Reconstruction from Depth Maps," ACM Transactionson Graphics, Canada N2L 3G1, pp.1-25, January 2000.

[8] X. Hu, Y. Qi and X. Shen, "A Real-Time Anti-Aliasing Shadow Algorithm Based on Shadow Maps" Pattern Recognition CCPR '08. Chine, Octobar 2008.

[9] S. Brabec, T. Annen, and H. Seidel. „Shadow mapping for hemispherical and omnidirectional light sources" Advances in Modelling, Animation and Rendering (ProceedingsComputer Graphics International 2002), pp: 397–408. Springer, 2002. ISBN 1-85233-654-4.

[10] J. Kessenich, D. Baldwin, and R. Rost, „The OpenGL Shading Language" (Version 1.051), 3DLabs, Inc., Egham, Surry, Feb. 2003, www.opengl.org/documentation/oglsl.html

[11] T. Akenine-Möller, E. Haines, and N. Hoffman." Real-Time Rendering". A K Peters, 3rd edition, 2008.

[12] R. Fernando (ed.). GPU Gems:" Programming Techniques, Tips, and Tricks for Real-Time Graphics", Reading, MA: Addison-Wesley Professional, 2004.