Збірник матеріалів Міжнародної науково – практичної Інтернет – конференції (Вінниця, ВНТУ, Украина, грудень 2014 року). – С. 30 – 37 .
–Режим доступа: https://drive.google.com/file/d/0B1ZDWosDI6cvVXJ1SVlBSlNwOWM/view?pli=1

Sergey I. Vyatkin

Alexander N. Romanyuk

Sergiy V. Pavlov

SHADOWS GENERATION USING GEOMETRY SHADERS

Institute of Automation and Electrometry SB RAS, Novosibirsk, 630090 Russia

Abstract

Algorithms for simulating shadows are considered. A shadow volume method using geometry shaders is proposed.

Introduction

Shadows are one of the most important features in computer graphics, both for the viewer to detect distance relationships between objects, and to make the scene more realistic.

Sharp shadows are kinds of shadows with a point light source. They have a sharp border outline, and they include fully shadowed regions. Soft shadows are another kind of shadow, which are caused by wide light sources. Soft shadows are more realistic. In the real world, most of light sources are wide and create soft shadows.

Soft shadows include three parts, umbra, penumbra and antumbra. The umbra is the innermost and darkest part of a shadow, where the light source is completely blocked by the occluding body. The penumbra is the region in which only a portion of the light source is obscured by the occluding body.

The antumbra is the region from which the occluding body appears entirely contained within the disc of the light source. Ray tracing is another concept, which can be used to create soft shadows. It is possible to create a soft shadow by distribute ray tracing using uniform voxel structures [1]. Radiosity is another technique to create soft shadows by computing the diffuse interrefections between different parts of the scene [2]. In paper [3] was proposed an algorithm for wide light sources. In work [4] was proposed an algorithm on soft shadows.

An algorithm for simulating a soft shadow for a wide light source was presented. They could generate soft shadows with complex occluders on arbitrary objects. In work [5] was used simple sampling with different colors to create soft shadows. In work [6] a geometric approach to create soft shadows was considered. In this method, a plane is a receiver, and an attenuation map produces soft shadows. The attenuation map is generated using modeling texture, which was created by converting the occluder's edges into a volume. Another technique is physically exact. In this case, shadows are created by extending the shadow pattern outward from the center of the light source [7]. Ray-casting based on shadow volume algorithm was proposed in [8]. An algorithm based on shadow volume using stencil buffers was presented in [9]. There are other works devoted to the generation of shadows [10-15].

In this paper, we propose a method of shadow volumes using geometry shaders.

Method description

The method of rendering shadows with shadow volumes has been around for quite some time, but it became practical only recently with the enhanced graphics hardware support. Modern hardware enables programmable primitive creation on the GPU in a new pipeline stage called the geometry shader. Geometry shaders are placed between the vertex shader (VS) and the fragment shader (FS). Previous approaches to creating shadow volumes entirely on the GPU required complicated operations with vertex and fragment shaders. A geometry shaders provide a "free" solution to this problem.

Geometry shader has the following capabilities and features:

1. Operates primitives.
2. Allows you to change the topology.
3. Creation of new primitives.
4. Removal of existing primitives.
5. You can access the adjacent primitives.
6. Special types of primitives.

7. Programmed in step "primitive assembly."

8. You can create new primitives!

9. Can use semantics SV_PrimitiveID.

10. Generating shadow volume.

11. Identifies the edges of the silhouette, generates faces.

12. The number of output attributes is limited -

Maximum - 1024 scalar attribute in D3D10.

Remarks:

Do not use for tessellation! And the importance MaxVertexCount.

The classic algorithm for generating shadows "Shadow Volumes" based on the construction of the shadow volume on points silhouette object casting the shadow. For convex objects is proposing to build a projection on a plane perpendicular to the direction of the light source. Then, the obtained points to build the envelope polyline. The resulting polygon is a silhouette of a convex object.

In our modified method discarded repeated more than once in the ribs from the complete list of all the edges of all the faces, cast a shadow. Computational complexity is O (n), where n is the number of faces of the object. In contrast to the classical version, this method uses additional information about the object, namely that the object is closed. In our case, this requirement is always satisfied in the construction stage.

In order to increase the realism of the generated image held additional adaptation algorithm for constructing silhouette. Modification allowed building a promising shadow projection. Silhouette of construction including the light source located at a finite distance from the shadowing object. Adaptation consists in a method of determining visible faces from the viewpoint of the light source. The decision as to whether the face is illuminated, taken with the distance to the light source. Calculated normal to the face by interpolation points and normal’s computed scalar product obtained from the normal direction to the light source.

As the shadow volume is made up of faces, shadow volume, like all other objects in the scene, geometric pipeline passes. It often happens that in the viewing pyramid gets only part of it. As a result, the shadow volume is no longer a closed three-dimensional object, as it should be in accordance with the classic algorithm. This leads to malfunction of the classical algorithm, for example, when the observer is located within the shadow (shadow volume). Since in this case is not taken into account the intersection of objects in the scene and the shadow volume that occurred outside the scope. To avoid this, it is necessary to limit the shadow volumes of the near and far clipping plane. Because, firstly, they are not presented in the frame buffer and thus artifacts arising on lateral clipping plane are not visible to the observer. Secondly, these planes are configured so as to include all objects in the scene, and thus shading objects and the upper part are always shadow volume inside. This makes it possible not to worry about the limitation of shadow volumes on the lateral cutting planes.

To close the shadow volume into front and back clipping plane construction used more geometric objects, so-called shadow volumes caps. Cap is a polygon, which is the result of crossing the shadow volume and clipping plane. Since we are interested in two cutting planes, distinguish "near" and "far" caps. Normal cap is chosen in such a way that the shadow volume was always closed. Building caps for arbitrary shapes difficult task of computational geometry, since the shadow volume is a complex polyhedron in general nonconvex and not simply connected.

Z-pass algorithm is an algorithm used for counting the shadows where there is more than one occluder. The first count is done according to view position. Whenever a ray from the light source enters the volume, the shadow count increases by 1. Whenever the ray exits from a shadow volume, the shadow count deceases by 1. If the ultimate shadow count is 0, then the visible pixel is lit, if the ultimate shadow count is positive then the visible pixel count is in shadow.

Z-pass is easy, but there is a problem with it. When the camera is inside a shadow volume, the algorithm yields the wrong results. The approach [9] is the most common way to generate shadow volumes. However, that it works correctly only for closed two-manifold polygon meshes, meaning that objects cannot have holes, cracks, or self-intersections. The algorithm needs to be extended in two ways. We also extrude edges that do not have any neighbor polygons at all. This is an obvious extension needed for nonclosed meshes. We take into account all the polygons in a mesh, not only the ones facing the light, to extrude possible silhouette edges and to draw the caps. The vertices of an entire primitive are available as input parameters.

That this new capability is ideally suited for the dynamic creation of shadow volumes. Silhouette determination must be redone every frame for animated scenes, so it is preferable to move the computational load from the CPU to the GPU. Geometry shaders reproducing the fixed-function pipeline would just take the input primitive and emit it again, in our case generating the front cap of a shadow volume. We will be creating additional primitives for the back cap and extruded silhouette edges.

To compute the silhouette edges of a mesh, the geometry shader has to have access to adjacency information of triangles. In OpenGL, we can pass in additional vertices per triangle. In this mode we need six, instead of three, vertices to complete a triangle, three of which specify the neighbor vertices of the edges.

To specifying the input primitive type, we need to specify the type of primitives geometry shaders will create. We use triangle strips, which lets us efficiently render single triangles for the caps, as well as quads for the extruded silhouette edges. The maximum allowed number of emitted vertices will be set to eighteen (three plus three for the two caps and plus 4 x 3 for the sides).

The improving performance is to organize all objects in the scene in a hierarchical tree structure. During rendering, the tree is recursively traversed in a front-to-back order, and the objects contained in the leaf nodes are rendered. Before a tree node is traversed, however, it is tested for visibility using the occlusion culling feature provided by the graphics hardware. If the node is found to be invisible, the entire subtree can be pruned. We extend this method to shadow volumes as well. The hierarchical culling method does not increase performance in all cases. For some scenes, rendering may even be slower compared to simply drawing all the objects in the scene. However, for the majority of our scenes, hierarchical culling improves performance.

The results of the algorithm presented in the following table.

Tested Configuration: Processor Intel Core2 CPU E8400 3.0 GHz.

Scene	**GPU**	**Resolution**	**Execution time frame, ms (with shadows / without)**
Scene «Town» 567823 faces, hierarchical organization of the database	GPU 9800 GT	640x480x32	18/09
		800x600x32	24/10
		1024x768x32	52/12
	470 GTX	640x480x32	12/10
		800x600x32	14/12
		1024x768x32	23/13
Scene «Town» 567823 faces, common organization of the database	GPU 9800 GT	640x480x32	65/51
		800x600x32	71/53
		1024x768x32	82/70
	470 GTX	640x480x32	65/56
		800x600x32	67/57
		1024x768x32	74/58

All measurements were performed in full-screen mode

Conclusion

We proposed rendering of shadows for complex scenes. By using this method in combination with hierarchical tree structure and geometry shaders, we achieve high performance.

REFERENCES

1. E. Lafortune, and Y. Willems. Bidirectional path tracing, In Proc. 3rd International Conference on Computational Graphics and Visualization Techniques (Compugraphics 93), pp. 145-153.
2. C. Goral, K. Torrance, D. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces, Computer Graphics, Vol. 18, no. 3, 1984, pp. 213-222.
3. Drettakis, George, and W. Fiume. A fast shadow algorithm for area light sources using backprojection, SIGGRAPH 94 Proceedings, 1994, pp. 223-230.
4. P. Heckbert, and M. Herf. Simulating soft shadows with graphics hardware, Carnegie Mellon University: Technical Report CMU-CS, 1997, pp. 97-104.
5. B. Gooch, P. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration, Proceedings 1999 Symposium on Interactive 3D Graphics, 1999, pp. 31-38.
6. E. Haines. Soft planar shadows using plateaus, Journal of Graphics Tools, Vol. 6, no. 1, 2001, pp. 19-27.
7. C. Soler and F. Sillion. Fast calculation of soft shadow textures using convolution, In Computer Graphics (SIGGRAPH 1998), pp. 321-332.
8. F. Crow. Shadow algorithms for computer graphics, Computer Graphics, Vol. 11, no. 2, 1977, pp. 242-248.
9. T. Heidmann. Real shadows real time, IRIS Universe, Vol. 11, 1991, pp. 28-31.
10. J. Hasenfratz, M. Lapierre, N. Holzschuch, and F. Sillion. A survey of real-time soft shadows algorithms, Computer Graphics Forum, Vol. 22, no. 4, 2003, pp. 753-774.
11. S. Morein. ATI Radeon hyperz technology, ACM SIGRAPH/ Eurographics, In Graphics Hardware Workshop, 2000, pp. 855-856.

12. H. Kolivand, and M.S. Sunar. To combine silhouette detection and stencil buffer for generating real-time shadow, International Journal of Computer Graphic, Vol. 2, no. 1, 2011, pp. 1-8.
13. T. Lokovic, and E. Veach. Deep shadow maps, In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000, pp. 385-392.
14. T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developers guide to silhouette algorithms for polygonal models, IEEE CG and A, 2003, pp. 28-37.
15. M. McCool. Shadow volume reconstruction from depth maps, ACM Transactions on Graphics, 2001, pp. 1-26.