Real-time Soft Shadows with Shadow Accumulation

Barnabás Aszódi, László Szirmay-Kalos

Department of Control Engineering and Information Technology, Budapest University of Technology, Hungary
Email: szirmay@iit.bme.hu

Abstract

This paper presents a physically plausible soft-shadow algorithm that can be executed real-time by current GPUs. The method works with a single shadow map, requires no pre- or post-processing, and can also handle self shadowing. The main novelty of the method is in the interpretation of the shadow map. A lexel with depth information is considered as the geometric definition of an elementary shadow caster. When the shadowing of a point is computed, the algorithm decides whether these elementary casters occlude the light source area from the shaded point, and the shadowing factors of the elementary casters are accumulated.

1. Introduction

Realistic light sources have non zero area, resulting in *soft shadows* having continuous transition, called *penumbra*, between the fully illuminated region and the occluded region, called *umbra*. Soft shadow algorithms should handle the visibility between shadow receiver points and all points of area light sources. Visibility information of a single point can be encoded by shadow maps, shadow volumes, or can be detected on the fly by ray tracing [LAA*05], similarly to hard shadow methods.

Multi-sample approaches [HLHS03, YTD02, AAM03] compute hard shadows from different light source samples, and average the partial results. While this approach works well for linear [HBS00] and small area lights, for large area lights, either the rendering speed or the quality degrade.

Single-sample approaches, on the other hand, process the scene only from one light source sample, and apply this information for all light source points. Smoothing may based on simple intuition. For example, *percentage closer filtering* (PCF) [RSC87], which has been originally developed for shadow map antialiasing, does not use additional geometric data, but softens shadows since it reads more than one lexels then applies low pass filtering. On the other hand, smoothing can be executed with rendering additional geometry, called smoothies [CD03]. Falloffs can even be computed analytically for spherical light source and a planar receiver [PSS98]. Realizing that the width and the density of the penumbra depend on the size of the area light source, on the distance between the light source and shadow caster object, and on the distance between the shadow caster and shadow receiver object, fragment programs may take additional geometric information written into textures [BS02, AW04, CD03, KJ03, WH03, KD03].

Single sample algorithms often ignore an important fact. A shadow map describes the visibility *from the light source*, but physically plausible soft shadow generation requires the visibility *from the shaded point*. While point-to-point visibility is mutual and symmetric, point-to-region visibility is not (left of figure 1). Thus the blurring is not physically plausible.

Our proposed algorithm is a single sample approach. However, it does not apply heuristic PCF to simulate soft shadows. Instead it morphs the shadow map and estimates the solid angles in which the total and the occluded light source are visible from the shaded point.

1.1. The new algorithm

Let us consider a homogeneous area light source illuminating receiver point $\vec{r}$ (right of figure 1). If the light source partially or fully occluded from point $\vec{r}$ by shadow caster *c*, then the reflected radiance decreases. If the light source emission is approximately uniform, the surface is diffuse or moderately glossy, and the light source is not too close to the illuminated surface, then the ratio of the radiances in the occluded and non-occluded cases is equal to the solid angles in which the not occluded and the total light sources are visible

from point $\vec{r}$. The difference between 1 and this ratio is the *shadowing factor*, and is denoted by $s(c,\vec{r})$.

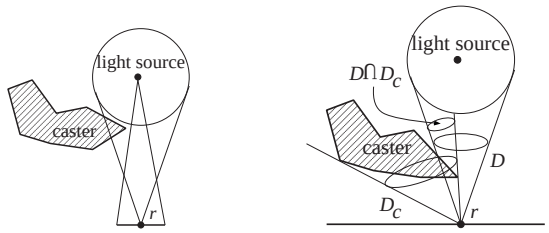


Figure 1: *Left: Point-to-region visibility is not symmetric. The center of the light source is visible from all points of a surface element, but not all points of the light source is visible from the center of this surface element. Right: Computation of shadowing factors as the ratio of solid angles of the occluded ($|D_c \cap D|$) and the total ($|D|$) light source part.*

Let D be the set of those directions from where the light source may illuminate shaded point $\vec{r}$ if no occlusion occurs. This set can also be imagined as the set of those rays that originate at $\vec{r}$ and intersect the light source. Similarly D_c is the set of directions where caster c is visible, or the set of those rays that originate at $\vec{r}$ and intersect caster c. A natural measure $|.|$ of these sets is the solid angle. According to its definition, the shadowing factor can also be expressed as:

$$s(c,\vec{r}) = \frac{|D_c \cap D|}{|D|}.$$

Lighting is an additive phenomenon. Shadows are due to "missing lighting", thus they are also additive if their casters do not occlude each other. Let us consider two shadow casters c_1 and c_2 and express their joint shadowing factor:

$$s(c_1 \cup c_2,\vec{r}) = \frac{|(D_{c_1} \cup D_{c_2}) \cap D|}{|D|}. \quad (1)$$

If $D_{c_1} \cap D_{c_2} \cap D$ is empty, that is, the casters do not occlude the light source from each other, then

$$s(c_1 \cup c_2,\vec{r}) = \frac{|D_{c_1} \cap D| + |D_{c_2} \cap D|}{|D|} = s(c_1,\vec{r}) + s(c_2,\vec{r}). \quad (2)$$

On the other hand, when caster c_1 occludes all points of c_2, that is $D_{c_1} \cap D_{c_2} \cap D = D_{c_1} \cap D$, then

$$s(c_1 \cup c_2,\vec{r}) = \frac{|D_{c_1} \cap D|}{|D|} = s(c_1,\vec{r}). \quad (3)$$

To compute the shadow factors, we need the visibility information *from* each shaded point $\vec{r}$, but the shadow map provides visibility information obtained from the light source. However, distances stored in lexels together with the center of the light source and the positions of the lexels can also be interpreted as a discretized version of the shadow caster geometry. The main idea of our method is to take a single shadow map as the geometric description of the scene, assume that each lexel represents a small surface area, compute shadowing factors of these elementary surfaces, and finally add up their shadowing factors using equation 1, or preferably equations 2 and 3 if their requirements are met. From another point of view, this means that not only a single lexel is read when a point is shaded, but all those lexels that possibly contain geometric information relevant for the shadowing of the given point.

The evaluation of the shadowing factor requires the identification of the direction sets and the execution of set operations. If the ray origin is fixed, then the ray direction can be defined by a point on a plane not containing the ray origin. To discretize this set, a raster grid is imposed on this plane, thus each pixel corresponds to a small directional set. Note that the shadow map also uses this representation.

To represent the visibility information *from* shaded point $\vec{r}$, we need a plane with a raster grid. A straightforward solution is to use the *same plane and raster grid* for this purpose as was used to identify the points visible from the light source. This means that in our solution a shadow caster point may correspond to two points of the projection plane. The information stored in the lexels of the plane must be transformed to reflect the differences of the two tasks.

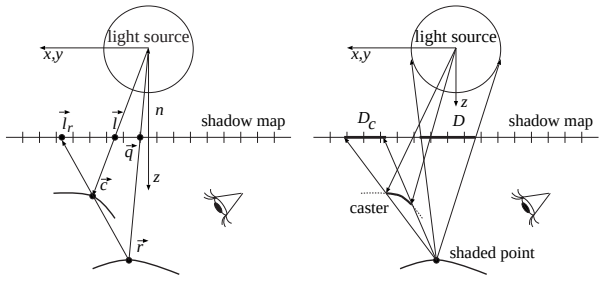


Figure 2: *Notations used in the discussion of the shadow accumulation algorithm. An elementary caster is a surface area visible from the light source center in a single lexel. This caster is seen from the shaded point in solid angle D_c defined by other lexels on the shadow map plane.*

Let us examine point $\vec{c}$ on the shadow caster surface, which is projected onto $\vec{l}$ of the shadow map plane taking the light source as the center of projection, and onto $\vec{l}_{\vec{r}}$ if shaded point $\vec{r}$ is the center of projection. Suppose that these points are given by Cartesian coordinates in the light's camera space. In this coordinate system the origin is the center of the light source and axis z is perpendicular to the plane of the shadow map (figure 2). Denoting the distance of the projection plane and the light source center by n, the projection of caster point $\vec{c} = (c_x, c_y, c_z)$ from the light source center onto the shadow map plane is

$$\vec{l} = \vec{c} \cdot \frac{n}{c_z}. \quad (4)$$

When the same point is projected from shaded point $\vec{r}$, we compute the intersection between line of parametric form $\vec{c}\cdot t+\vec{r}\cdot(1-t)$ where t is a scalar parameter, and the shadow map plane of equation $z=n$, resulting in

$$\vec{l_{\vec{r}}} = \frac{n-c_z}{r_z-c_z}\cdot\vec{r} + \frac{r_z-n}{r_z-c_z}\cdot\vec{c}.$$

Substituting this into equation 4, we obtain:

$$\vec{l_{\vec{r}}} = \frac{n-c_z}{r_z-c_z}\cdot\vec{r} + \frac{r_z-n}{r_z-c_z}\cdot\frac{c_z}{n}\cdot\vec{l}.$$

Note that this equation defines a scaling and a translation between $\vec{l}$ and $\vec{l_{\vec{r}}}$. A lexel is a rectangular area whose points correspond to an elementary shadow caster surface, which are projected back to the shadow map from the shaded point. Thus the mapping of the points corresponding to a lexel results in another, roughly rectangular area representing those directions from where this caster may occlude shaded point $\vec{r}$. The center of this area is obtained by transforming the center of lexel $\vec{l}$. The elementary caster covers a single lexel from the point of view of the light source. Consequently it covers roughly $\left(\frac{c_z-n}{r_z-c_z}\cdot\frac{c_z}{n}\right)^2$ number of lexels when it is seen from shaded point $\vec{r}$.

The shadowing factor caused by this caster is the relative number of lexels in which both the caster and the light source are seen with respect to those lexels where only the light source is seen. In order to process all potential casters, a neighborhood is searched on the shadow plane around $\vec{q}$, which is the projection of shaded point $\vec{r}$. This neighborhood should be large enough to include all possible casters. When processing a lexel of the neighborhood, its potential caster is examined and we check whether it can occlude a light source part from the shaded point. The occluded part is also represented by that lexel on the shadow plane, whose occlusion bit is set (`obit`). Note that a lexel's occlusion bit might be set several times since a light source point might be occluded by several casters from the shaded point, which corresponds to the intersection set operation $D_c \cap D$.

The following algorithm finds the solid angles of the light source (D) and its occluded part (DcD). The solid angles are represented by the lexels, and set operations are executed on the plane of the shadow map.

```
Set obits to zero in D
for each lexel l in D
   for each caster c of lexel l
      size = (r.z-n)/(r.z-c.z) * c.z/n;
      lr = (n-c.z)/(r.z-c.z) * r + size * l;
      for each lexel l' closer to lr than size
         if (l' is in D) obit[l'] = 1;
DcD = 0;
for each lexel l' in D if (obit[l']) DcD++;
s = DcD / number of lexels in D;
```

This algorithm evaluates equation 1 and is quite difficult to implement on the GPU because of the complicated conditions. Another problem is that several casters may correspond to a single lexel, thus classical shadow maps storing only the closest caster cannot be used. Instead, we should store all casters in a kind of layered depth image.

In order to make the algorithm faster and the implementation easier, we take several simplifying assumptions. These assumptions will replace formula 1 by formulae 2 and 3, and allow us to get rid of the complicated conditions:

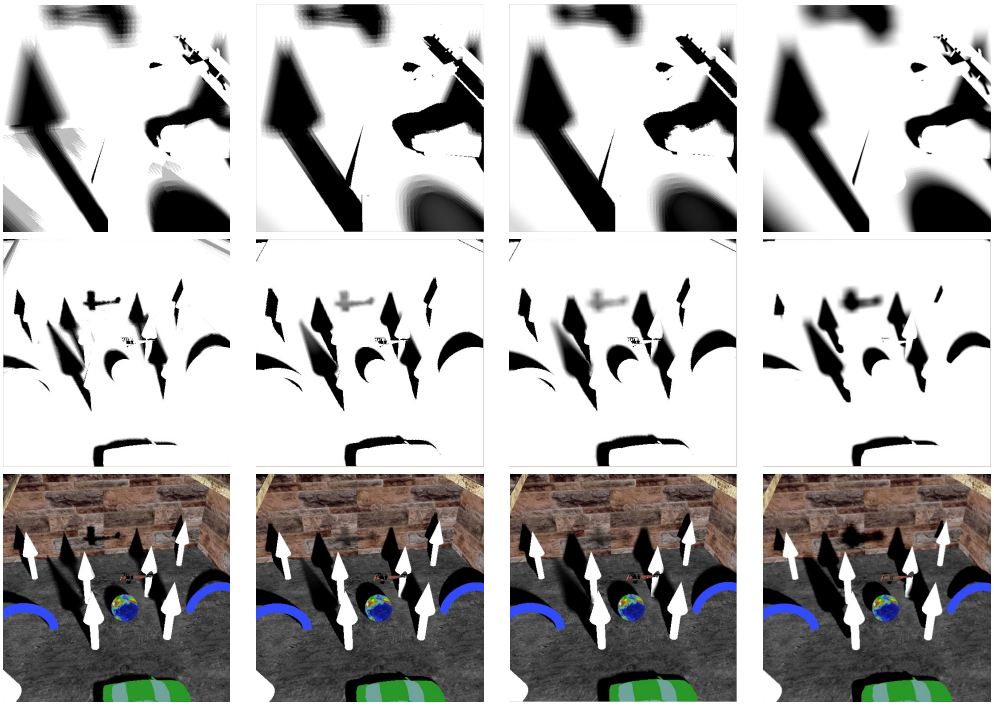
1. The light source is assumed to be a sphere of radius R. Setting shadow map plane distance n proportional to R, the neighborhood in which the light source is visible from $\vec{r}$ can be covered by the same number of lexels, $K\times K$, where K is called *kernel size*.
2. We keep only the closest caster in each lexel and assume that they do not occlude each other. Note that if the light source is reasonably sized, then most of the shadow casters belonging to the second and further layers are occluded from all points of the light source, thus their contribution to the shadowing factors is already zero (equation 3).
3. The casters in the first layer are assumed not to occlude each other from the point of view of the shaded point. This allows the application of equation 2. Note that these casters do not occlude each from the point of view of the light source due to their definition. If this condition holds, then we do not have to compute the occlusion bits of the lexels, but we can immediately add the shadowing factors of elementary casters. Note that this assumption may fail if the shadow receiver is occluded by more than one objects, where one is close to the light source while the other is close to the shadow receiver. In such cases the soft shadow cast onto the object becomes lighter than expected.

Taking advantage of the simplifying assumptions the pseudo-code of the computation of the shadowing factor is:

```
R'= R * (r.z-n)/r.z;
s = 0;
for each lexel l of square KxK
   size = (r.z-n)/(r.z-c.z) * c.z/n;
   lr = (n-c.z)/(r.z-c.z) * r + size * l;
   if (|lr - q| < R') s += size * size;
endfor
s /= R' * R' * PI / K / K;
```

2. Results and discussion

The proposed method has been implemented on an NV7800GT graphics card and compared to percentage closer soft shadows and to a multisample reference solution obtained with 64 light source samples. The images generated with different kernel sizes are shown by figure 3. Note that our results are closer to the reference solution and is faster than percentage closer soft shadows.



PCSF, $K = 4$, FPS=32 | new, $K = 4$, FPS=50 | new, $K = 5$, FPS=32 | reference

Figure 3: *Soft shadows generated with* 512×512 *resolution shadow map with different kernel sizes (K), as compared with percentage closer soft shadows (PCSF) and with a reference solution.*

3. Conclusions

This paper presented soft-shadow algorithm which can be executed on about 50 FPS on current GPUs and results in physically plausible results. The algorithm requires just one shadow map and can be seamlessly included into systems generating depth mapped shadows.

4. Acknowledgement

This work has been supported by OTKA (ref. No.: T042735), GameTools FP6 (IST-2-004363) project, and by the Spanish-Hungarian Fund (E-26/04).

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics* (2003).

[AW04] ARVO J., WESTERHOLM J.: Hardware accelerated soft shadows using penumbra quads. *Journal of WSCG* (2004).

[BS02] BRABEC S., SEIDEL H.: Single sample soft shadows using depth maps. In *GI 2002 Proceedings* (2002), pp. 219–228.

[CD03] CHAN E., DURAND F.: Rendering fake soft shadows with smoothies. In *Eurographics Symposium on Rendering* (2003).

[HBS00] HEIDRICH W., BRABEC S., SEIDEL H.-P.: Soft shadow maps for linear lights. In *Rendering Techniques '2000* (2000), pp. 269–280.

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F. X.: A survey of realtime soft shadow algorithms. In *Eurographics Conference. State of the Art Reports* (2003).

[KD03] KIRSCH F., DOELLNER J.: Real-time soft shadows using a single light sample. Skala V., (Ed.), vol. 11, pp. 255–262.

[KJ03] KIRSCH F., J D.: Real-time soft shadows using a single light source sample. *Journal of WSCG* (2003).

[LAA*05] LAINE S., AILA T., ASSARSSON U., LEHTINEN J., AKENINE-MÖLLER T.: Soft shadow volumes for ray tracing. In *ACM SIGGRAPH* (2005).

[PSS98] PARKER S., SHIRLEY P., SINITS B.: *Single sample soft shadows.* Tech. Rep. UUCS98-019, University of Utah, 1998.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. pp. 283–291.

[WH03] WYMAN C., HANSEN C.: Penumbra maps. In *Eurographics symposium on rendering* (2003).

[YTD02] YING Z., TANG M., DONG J.: Soft shadow maps for area light by area approximation. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (2002).