Real-time Computation of Area Shadows - A Geometrical Algorithm

Michael Boehl
Department of Computer Science
University of Karlsruhe, Germany
Michael.Boehl@stud.uni-karlsruhe.de

Wolfgang Stuerzlinger
Department of Computer Science
York University, Toronto
http://www.cs.yorku.ca/~wolfgang

Abstract

The computation of soft shadows created by area light sources is a well-known problem in computer graphics. Due to the complexity of the problem, soft shadows commonly are generated only for images that are rendered in an off-line process. In interactive virtual environments, where images have to be computed in real time, soft shadows are mostly replaced by hard shadows as this takes much less time to compute.

This paper presents an algorithm that uses a geometrical method to generate a triangulated approximation of the soft shadow cast by a polygonal object very quickly. The algorithm can simulate effects such as the varying width of penumbras depending on distance to the light and can be extended to support non-convex light sources as well. We mention the artifacts arising from triangulation and give discuss how to alleviate such problems.

1. Introduction

In our visual perception of the daily world, shadows do not seem to play an important role as we barely notice them consciously. In some situations they are more of a nuisance (clouds hiding the sun; taking pictures of people against the light and getting only black silhouettes), especially our own shadow never seems to be any good for anything.

In Computer Graphics worlds without shadow can be constructed. Looking at those scenes, most of us probably will not realize immediately what is missing. Yet it is noticeable that there is something odd about an image. Objects without shadows definitely look less real, which is one major reason why image generation programs, seeking to create a perfectly real looking artificial world, use more or less sophisticated shadow generation techniques. In interactive virtual environments, quick image generation is more important that a perfect illusion – if the feedback on the user's actions is noticeably lagging behind or the frame rate is too low (because image generation takes too long), the desired immersion of the user into the virtual environment just will not happen. Shadow computation is only a secondary problem for scene realism; on the other hand, it can demand considerable computing effort. Therefore, if the available computing power is scarce, shadows are often omitted in favor of a faster simulation (e.g. in computer games).

For certain tasks in virtual environments, shadows are very important. They give the user enhanced visual feedback on the position of objects in space. This is essential when interacting with a three-dimensional virtual world on a two-dimensional display. Ambiguous situations can often be resolved with the correct shadow information. Soft shadows give even more accurate impression of object placement than simple hard shadows, but take more time to generate.

2. Shadow Computation

The Computation of realistic soft shadows from extended light sources is a complex problem, considerably harder to solve than the generation of shadows from point light sources. Basically, the dimensions corresponding to the spatial extent of the light source increase the complexity. Soft shadows can be modeled as the result of a convolution of two three-dimensional functions, a light source function, and an occluding function.

For a real time solution, the computational effort has to be reduced, e.g. by simplifying the soft shadow problem to a problem of multiple hard shadows. For this simplification, the area light source is divided into smaller patches and a sample point represents each patch. The resulting grid of point light sources approximates the area light source.

A shadow is the area on an object surface from where a light source is occluded by other scene objects. For extended light sources, there exist regions of the shadow called penumbra where the light is only partially blocked by the occluder. The core of the shadow from where the light source is not visible at all is the umbra. The penumbra gives the shadow a soft appearance, caused by gradually fading intensity around the shadow borders. In contrast, shadows produced by point light sources have sharp borders, hence the notion of soft and hard shadows.

2.1 Anatomy of Soft Shadows

Soft object shadows are produced by an area light source or, more general, by a light source with one to three dimensions (point light sources have no extension, they produce hard shadows). The simplest case is a (almost) linear light source (like a fluorescent strip light), which can be modeled as a one-dimensional light source. Furthermore we assume that the occluding object is two-dimensional, in a plane parallel to the projection plane and to a plane containing the light source. Then the penumbra of the shadow produced by

this linear light source changes in width along the shadow boundary. A long object edge parallel to the axis of the light source will cast a sharp shadow boundary whereas if the orientation of the edge is perpendicular to the light source axis, the shadow boundary it casts will appear blurred. As explained, only a part of the light source is visible from a point in the penumbra area. Because this illuminating part changes linearly with the movement on the plane along the light source axis, brightness increases and decreases in a linear fashion within the penumbra zone.

The shape of a soft shadow becomes more complex if the light source is two-dimensional. Now the shadow does not have sharp boundaries any more, the penumbra region totally surrounds the umbra. The question is how the shape of the light source influences the appearance of the shadow. Clearly, the size of the penumbra region will depend on the size of the light source. If the light source has a long and thin shape, the penumbra region will be stretched in the direction parallel to the light source's long axis. It will be narrow in the other direction, perpendicular to that axis.

The gradient of brightness in the penumbra regions generally is not linear for area light sources. To understand this, we take a triangular light source. When going from umbra to light on the projection plane, in some configurations one edge of the triangle (almost parallel to the occluding object edge) will become visible first. The further we advance, the narrower new areas of the light source becoming visible get. In the end, the third tip of the triangle appears, not adding significantly to the brightness of the light – the gradient will be very low compared to the region near the umbra. The gradient function directly depends on the width of the light source where it 'touches' the occluding object edge, as seen from a point on the projection plane.

For three-dimensional objects, another property of soft shadows becomes apparent: the penumbra of the shadow gives an impression of the object distance from the projection plane. The soft shadow of a pole standing on the projection plane will have sharp borders close to the pole base and the shadow border of the pole top will appear fairly blurred in contrast. This property gives the viewer an important feedback on the relative position of objects in a scene that otherwise cannot be easily deduced from the two-dimensional image.

2.2 Previous Work

Real time algorithms can be divided in two groups: geometrical and image-based methods. Image-based algorithms create shadow maps. These are then directly applied as textures, shading the surface appropriately. The ways of creating shadow maps range from simple projections, possibly using graphics hardware buffers [2], to FFT operations. Image-based methods are often used because they are fast, easy-to-implement algorithms; their running time is fairly independent from scene complexity and the results can be quite good. One problem is the trade-off between memory and bandwidth costs for a high resolution of the maps and pixel-based artifacts with low map resolutions. The simpler algorithms for soft shadow generation essentially employ brute-force operations, approximating soft shadows by adding up multiple shadow maps from point light source samples. This leads to discontinuity artifacts unless a very high number of samples are used.

Woo et al. [1] give an excellent overview on shadow generation algorithms. While their work was written more than a decade ago, it is still a comprehensive description of the basic techniques used in shadow algorithms. Its emphasis lies on ray tracing methods; real time algorithms became an important field of research only recently with the increasing availability of computing power and the enhanced capabilities of new generations of graphics hardware.

A good example for current real time image-based methods is the algorithm developed by Soler & Sillion [4]. They exploit the fact that shadows from area light sources can be modeled as a convolution if the light source and the occluder are two-dimensional. Their method does not produce discontinuity artifacts and penumbras appear smooth. Light source and occluders are represented by a projection on two-dimensional maps parallel to the receiver plane. These maps are the operators of an FFT operation producing the shadow map. The dimensional flattening causes problems with occluders and light sources having a great extension in the light direction axis. To alleviate that problem, Soler & Sillion introduce a hierarchical sub-partitioning of the occluders into multiple distinct parts, each represented by one map. The method produces good results, also for complex shapes and clusters of objects. Since FFTs can be computed by special hardware (DSPs), there is a high potential for further speedup of the algorithm. Yet, the use of shadow maps also has its disadvantages. Image-based methods are not exact. For low map resolutions the shadow shows pixel artefacts or has to be blurred to avoid them. High map resolutions are mostly not feasible due to scarce computing resources, especially in scenes with many objects. For most of the shadow maps a high resolution would waste resources

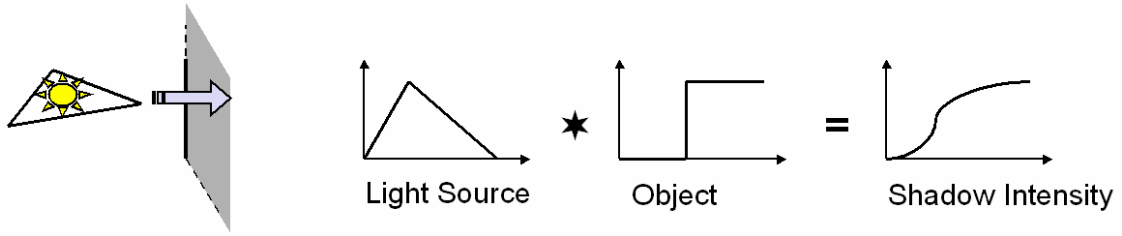

Image 1 - Modeling a Shadow as a Convolution Operation

because they are displayed only as small (distant) effects, reduced to a few pixels in the rendered image.

The geometrical approach promises an exact representation of soft shadows, at any scale. Stark et al. [5] examined the theoretical basis of that approach. The authors analyze the irradiance distribution of soft shadows from polygonal area light sources. They describe a method yielding an exact polyhedral spline representation of soft shadows cast by a polygonal occluder. Worrall, Hedley and Paddon present an implementation of a geometrical algorithm in [6]. They base their algorithm on research done in the field of discontinuity meshes. Those meshes can geometrically describe the brightness distribution of shadows cast by polygonal occluders. In this context, Worrall et al. also develop a method of updating shadow information for moving objects. In certain situations, this can be a rewarding alternative to a complete shadow re-computation. They show that the reuse of shadow information can introduce a high level of independence from scene complexity. This result is significant because in general the runtime of geometrical methods directly depends on scene complexity, which is one of their biggest drawbacks.

3. Algorithm Description

The method proposed for soft shadow creation makes use of an approximation of the light source through multiple point light sources. The hard shadows of the sample point light sources are combined and interpolated to form the soft shadow area. As a further simplification, it is assumed that the projection surface of the shadow (the ground) is planar.

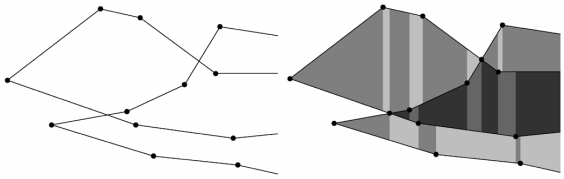

Figure 2 - Sweep Line Algorithm

The shadow generation method consists of multiple algorithms. As a first step, the hard shadow boundary for each light source sample has to be computed geometrically, these silhouettes then will be combined to generate the appropriate soft shadow.

3.1 Geometrical Computation of Soft Shadows

Input to the algorithm is a triangle model of the occluding object that has to be topologically correct (the object is closed, all triangles have an inside/ outside orientation). A structure storing all object edges (triangle boundaries) and linking to the adjacent triangles is built in memory once when the model is loaded. As long as the meshing of the object does not change, this structure can be reused in every algorithm cycle.

All projections of object edges are located within the shadow area. At the same time, any point on the silhouette boundary is also a point on a projected edge. The silhouette boundary is the minimal hull enclosing all projected edges. A straightforward approach to geometrical shadow computation would be to project all object edges onto the plane and construct a minimal line loop enclosing all of the edge line segments. Yet, in addition to the wasted computation effort for many unnecessary edge projections, this approach neglects holes in the object – they get filled.

To reduce the number of projections, only edges that can possibly be a part of the silhouette should be processed. Each edge is a boundary line of two triangles on the object surface. If both triangle outsides face the light source or point away from it (cross product evaluation), the common edge cannot be part of the silhouette boundary. In an average organic model, the number of edges to process can be reduced by two-thirds this way.

The projection of all remaining edges forms a pattern of several closed line loops. These loops enclose shadow areas or hole areas. For differentiation of the two cases, inside/outside information has to be stored for each projected edge. Introducing directions for each achieves this and by defining that e.g. the right side (following the forward direction of an edge) is the side where the adjacent triangles would cast their shadow (inside).

The area of the hard shadow is the composition of all parts enclosed by those line loops. If we give every shadow region an index of 1, every hole region an index of –1 and superimpose all regions by summing up the indices at any point, the combination of all areas with a sum of at least 1 equals the shadow area. The algorithm computing the shadow area using the described method is a sweep line algorithm. All projected edges are first sorted by their minimal X-Coordinate. The sweep line (parallel to the Y axis) is set to the absolute minimum of all X-Coordinates.

In each step, the sweep line is advanced to the next starting position of an edge. The algorithm keeps track of all edges cutting the sweep line – this edge list is updated every time the sweep line advances. When an edge is inserted, it gets checked for intersections with other edges in which case both edges are split into nonintersecting parts. To find out which parts of the sweep line lie in the shadow area, a winding number counter is used: beginning at the minimal Y position and a winding number of 0, the winding number is increased by one for every edge cutting the sweep line where the ‘inside’ of the edge is entered with increasing Y values, otherwise it is decreased by one. Edges of interest for shadow generation are only edges where the winding number changes from 0 to 1 respectively from 1 to 0; those edges are part of the shadow silhouette boundary.

The algorithm can be used for two different operations. If the hard shadow has to be displayed, it can supply a triangulation of the shadow area. The triangulation is not minimal but it requires almost no additional computation effort. While the sweep line advances, for every shadow boundary edge ending (and therefore falling out of the edge list) a trapezoid strip is added to the shadow area. If only the shadow boundary is needed (e.g. for soft shadow computation), the algorithm can

generate a chain of all boundary edges, structured as a linked list.

3.2 Merging Silhouettes

The silhouette algorithm described above is used to create multiple hard shadow silhouettes that are merged into a soft shadow. For the generation of each silhouette, a sample point of the light source is used. The distribution of the sample points has a strong influence on the appearance of the resulting shadow. If the sample points are badly chosen, the light source may not be represented correctly, or in the worst case, artifacts can increase dramatically. Positioning multiple sample points in line, for example, can lead to extensive step artifacts because multiple silhouette lines will overlap in certain areas.

When the silhouettes have been computed, several things have to be performed to join them into a single data structure. First, intersections have to be found and resolved. Second, the line data has to be reorganized into 'rings' – the outmost ring marks the absolute shadow border, the beginning of the penumbra region, while the last ring encircles the umbra region where the shadow reaches its full intensity (with a value of 1). At the first ring the intensity value is 0. The idea behind the gradient fill algorithm is to give every point on any silhouette line an intensity value and fill the space between the rings using linear interpolation. For a number of *n* silhouettes (and consequently n rings), the intensity value for a point on the i^{th} ring (counted from the outside) is *(i-1)/(n-1)*. Intersection points between two silhouettes are located on two rings; they receive an average intensity value between the values of the two rings.

The algorithm for the reorganization of the data is another sweep line algorithm. The algorithm has to keep track of the winding number in order to mark all silhouette points with the right intensity value, which is one reason for the use of a sweep line. Intersections between silhouette lines can also be found by means of the sweep line. The merging algorithm is simpler and quicker than the silhouette algorithm – first, because there is less data to process and no edges have to be discarded, and then because the silhouette data structure can be reused, most edge elements are used, as they are stored in memory.

The silhouette data consists of edge elements organized in linked lists. Each element points to the next edge in the silhouette line. The linked lists have no end elements because the silhouette lines are closed. For the sweep, the edge elements first have to be sorted. The sweep line starts in a similar way as described above. Every element gets an intensity value based on the winding number count (equals 'i', the ring count). If the algorithm finds an intersection, a new vertex is created and new edge elements are inserted into the silhouette lists. Then, the lines are cut apart at the intersection point and joined the other way so that the two lines will touch, but not cross each other. When the algorithm finishes, the silhouettes have been transformed into a ring structure, essentially linked lists with additional cross-links at the intersection points.

The ring structure describes an approximate shadow distribution function in the penumbra region. The outmost ring demarks the outer border where shadow intensity is zero. The last ring on the inside lies on the border between penumbra and umbra – the intensity of the shadow reaches its maximum there as well as in the whole umbra region. Since the light source is approximated through samples, the rings do not fully match the true shadow borders – the resulting penumbra region generally is too small. This becomes quite obvious when looking at the way the sample silhouettes (hard shadow silhouettes from single light source samples) overlap. Most silhouettes have about the same size (when the area light source is oriented parallel to the ground plane) and a similar shape. For almost every silhouette there exists a line part that lies outside of any other one of the projected silhouettes. This means that this line part belongs to the outmost ring. So if another light source sample point is added, in many cases the corresponding silhouette expands the shadow area – more samples naturally represent the light source better and a more precise reproduction of the shadow is achieved.

3.3 Filling the Shadow

From the silhouette-merging algorithm we obtained a structure describing the shadow boundaries and an approximate intensity distribution. To display the shadow, the shadow area has to be filled – which is the crucial part of the whole shadow generation algorithm. The requirements a suitable algorithm has to meet include defined computation time bounds due to the real time application and a visually satisfying result (including a smooth interpolation).

3.3.1 Filling the Penumbra

As described before, the shadow area falls into two parts, the umbra and the penumbra. Let's consider how to fill the penumbra area first. The penumbra lies in the region between the rings. Since the shadow density is defined on the ring lines, this data should be used for filling. Hence, the basic algorithm fills the space between two adjacent rings, a strip, by drawing triangles. It starts at one ring point on each strip side, ideally an intersection point where both rings meet. For the next triangle to draw, the algorithm compares the next point on either ring and uses the point that is nearest to get a well shaped triangle. It is important that the triangles have points on either side; otherwise they will have a plain density (color) value and not a gradient as is needed for interpolation. Yet it is not always possible to build only such triangles. If one of the delimiting rings forms a bay, some points of that bay may not be reachable in a straight line from the other ring without cutting an area outside of the strip. In those cases, the bay is filled with a plain density value. Another problem lies in the nature of the algorithm. Since it progresses in one direction and considers only the direct forward edges form the current points, it may happen that an edge in 'forward' direction runs back into a region already filled. This means that the algorithm has filled some part outside of the strip,

drawn triangles have to be removed because they must not overlap with other triangles. For that reason, triangles are not drawn directly but stored in a list first. That way they can be easily removed and the algorithm corrects the triangulation, filling some corner parts with plain density triangles.
The gradient of the triangles drawn is always linear; OpenGL does not allow more complex transitions. The triangulation of the penumbra can therefore not accurately display the real shadow distribution. As noted above, the gradient differs with the shape of the light source. Furthermore, the gradient is a continuous function at any point. The gradient function filling the penumbra triangles in contrast can not be continuous at an edge where two triangles meet – unless the gradient points in the same direction for the adjacent triangles. The discontinuity unfortunately is directly noticeable for the human eye, our visual system even emphasizes it (similar to the 'Mach Band' effect). Figure 3 above show some examples: on the left there is a 90 degree change in the gradient at the center, in the middle the change is alleviated by a third triangle inserted in between. The correct gradient distribution around the corner is shown in the right part of the figure, shadow intensity values are the same for all points with the same distance to the corner point. This radial distribution cannot be constructed with triangles. Yet the effect gets less visible if the angle between the gradient directions in adjacent triangles is small. The discontinuity effect in conjunction with Gouraud shading has also been described in [3].

Figure 3 - Artifacts caused by Gouraud Shading

Most noticeable are the gradient transitions around intersection points in the rings. Since an intersection point gets a different intensity value than its neighbor points on the ring, the gradient direction of triangles with a corner at the intersection often differs strongly from the adjacent triangles. To solve this problem, one could try to eliminate intersection points from the ring structure. Instead of inserting an intersection point where two lines intersect, the points around the intersection could be reconnected in a different way (see Figure 4). The gradient of the triangles around an intersection will be more homogenous with this transformation. The rings have been turned into 'iso-density lines'; the shadow has the same density value at all points on a ring line.
A problem of the transformation is the insertion of new lines. Those lines have to be checked for intersections with lines on other rings. If there is an intersection, the inserted lines have to be replaced again and the check has to be repeated. For the projected silhouettes, intersection points often cluster in certain regions, so the described case of producing new intersections by inserting new lines is common. The intersection checks force the algorithm to parse the line structure repeatedly, making it hard to adhere to given limits for computation time. That is why I decided not to apply this method in the real time algorithm.
The appearance of the interpolated penumbra region also depends strongly on the choice of the light source sample points. If the sample points lie on a regular grid, a straight line can connect many of them. This may lead to artifacts since object edges fairly parallel to that line get projected in the same region for all light samples positioned on the line. A high density of silhouette lines means a high gradient in shadow intensity – this is visible as a 'step' in the penumbra region, especially if the penumbra is relatively broad. Therefore it is advisable to choose a 'random' distribution of the light source sample points. Yet at the same time, points positioned on the border of the light source should be preferred, to maximize the penumbra region.

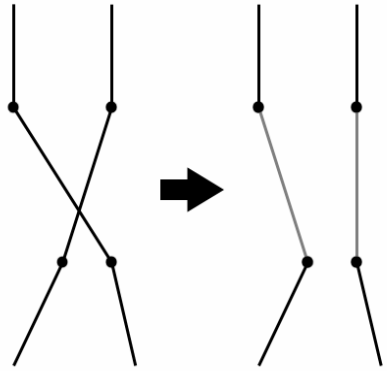

Figure 4 - Possible Optimization for Silhouette Rings (see text)

Even with randomly distributed sample points, there will still exist situations where silhouette lines are densely packed in certain areas of the projection. In order to improve the visual appearance of the interpolation, a single 'average' line might replace lines very close to one another. Whether lines are to be merged could be determined by a threshold value for the distance between a line point and the nearest other line.
At the endpoints of the average line multiple silhouette lines would be joined respectively separated. Advantages are that the interpolation would look smoother due to the replacement of many small triangles by fewer larger ones. Considerable numerical instabilities could be avoided as well, with less extremely narrow triangles in the tessellation. This concept – grouping multiple lines depending on a threshold – is not part of the current implementation.

3.3.2 Filling the Umbra

The umbra is the zone of maximum darkness or highest shadow intensity. The intensity distribution is even; the filling triangles have no gradient but a plain intensity value. Because there are no gradient step artifacts, any triangulation will have the same visual result. As described, a scan line algorithm is used for merging the

silhouettes – a similar filling algorithm is now applied for the umbra; it could be integrated into the other sweep for speed optimization reasons. As before, the algorithm keeps track of a winding number for any area between two edges on the scan line. The umbra lies where the winding number equals the total number of silhouettes / light source samples. When an edge line delimiting the umbra ends and is taken out of the scan line edge list, the algorithm fills the scanned umbra area with a trapezoid (up to the scan line position). The triangulation needs twice the optimal (=minimal) number of triangles, but the algorithm is deterministically one-pass.

One problem not mentioned yet is the intersection points on the umbra border. They do not truly delimit the umbra, their intensity value is not maximal. Yet they are part of the inner ring delimiting the umbra. For an even intensity distribution in the umbra, those 'leaks' need to be fixed – interpolation triangles have to be inserted.

The situation is similar on the outmost ring – at intersection points the intensity is not zero. Here, the same solution can be applied: the insertion of triangles that interpolate from the intersection point to neighbor points on the outer border.

4. Results

A major drawback of geometrical methods is numerical instabilities. Problems may arise with narrow triangles in the triangulation because coordinates are not calculated exactly. More severe are the effects appearing with moving shadows. As explained, the filling algorithm uses plain intensity triangles in some corners of the ring lines. When the silhouettes shift, the local situations change and plain triangles get replaced by interpolating triangles or vice versa. This will make the shadow movement look jumpy – which is really disturbing because it draws the viewer's attention to the shadow. Even if there are no plain intensity triangles, changes in the triangulation are visible because of the gradient step effects shown in (see Figure 5).

To alleviate the problems induced by moving shadows, an algorithm might have to consider the previous shadow tessellation in the computation of the updated shadow. Yet, it may be impossible to sustain a previous tessellation because the shifting silhouette lines caused by the shadow movement change the shape of the penumbra area considerably.

Considering these methodic drawbacks, the visual results are quite acceptable. The artifacts with moving shadows are not as severe as expected. Most noticeable are artifacts due to the 'Mach Band' effect – the triangulation becomes visible where triangles with almost orthogonal gradient vectors join at an edge. This problem might be solved with a different triangulation algorithm.

As for the speed of the algorithm, the performance is quite good. The model used for testing consists of about 5000 vertices. The results indicate a performance of approximately 10 fps on a 1 GHz machine, where the light source consists of three point samples. A faster computation could be achieved by using multi-resolution objects (level of detail – LOD). For shadow computation a coarser resolution of the object can be used than for displaying the object itself. Finer details of the object are blurred by the penumbra of the shadow in many cases – therefore the difference will not be noticeable.

The implemented geometrical method for the computation of soft shadows is quite powerful since it allows the approximation of light sources of any shape. In the current version, the algorithm assumes that light sources are of convex shape – the connection line between any two light source samples is assumed to be part of the light source. Therefore shadow density values are interpolated between any two silhouette lines. A modified algorithm could allow the light source to be composed from several convex shapes (defined by clusters of light source samples). In this case, a separate soft shadow needs to be computed for every cluster – the results are merged without interpolation in a final step. A scene with multiple light sources can be handled the same way.

Figure 5 – Close-up of Gouraud shading artifacts (see text)

Shadow-casting objects can be of any shape, they do not need to be reduced to two-dimensional representations as with other methods (see chapter 2.3). The only restriction is that an object should not contain any parts thinner than the 'resolution' of the light source (distance between two light source samples) – scaled by the relation of the object / projection plane distance to the light source / object distance. Otherwise, problems might arise with thin parts where the algorithm casts silhouette lines that do not overlap. A region surrounded by only one silhouette line will not be visible as a part of the shadow because the merging algorithm gives the outmost silhouette line a shadow intensity of 0 and fills the region with a plain gradient.

Yet, this effect partially is a natural phenomenon. It can be observed with real soft shadows as well – thin objects do not cast any significant shadow when lighted by a larger extended light source.

5. Conclusion

The results of the developed algorithm are promising. The implementation can surely be optimized to a

performance at least twice as fast as realized. Yet even the current state shows the feasibility of the concept.

In order to improve the visual quality of the shadow, the tessellation algorithm for the penumbra region may be a major target of optimization. In its current version, it strives for smooth interpolation by maximizing the fraction of interpolating triangles, minimizing the number of plain-density triangles. Yet, a maximized number of gradient triangles may deliver a worse quality than other triangulations – due to the effects appearing around the triangle edges. A Delauney triangulation might be a better alternative, because it creates well-shaped triangles, which reduces gradient artifacts and numerical problems. On the other hand, any suitable triangulation algorithm should have fixed computation time bounds, independent from scene geometry, given that the complexity of the occluder does not change.

References

[1] Andrew Woo, Pierre Poulin, and Alain Fournier, *A survey of shadow algorithms*, IEEE CG&A 10(6):13–32, Nov. 1990.

[2] Paul S. Heckbert and Michael Herf, *Simulating soft shadows with graphics hardware*, Technical report, CS-97-104, Carnegie Mellon U. 1997.

[3] Andrew Woo, Andrew Pearce, and Marc Ouellette, *It's Really Not a Rendering Bug, You See...*, IEEE CG&A, 16(5):21-25, 1996.

[4] Soler, Cyril, and François Sillion, *Fast Calculation of Soft Shadow Textures Using Convolution*, SIGGRAPH '98, 321-332.

[5] Michael M. Stark, Elaine Cohen, Tom Lyche and Richard F. Riesenfeld, *Computing Exact Shadow Irradiance Using Splines*, SIGGRAPH'99, 155-164.

[6] Adam Worrall, David Hedley and Derek Paddon, *Interactive Animation of Soft Shadows*, Proc. IEEE Computer Animation 1998, 88-94.

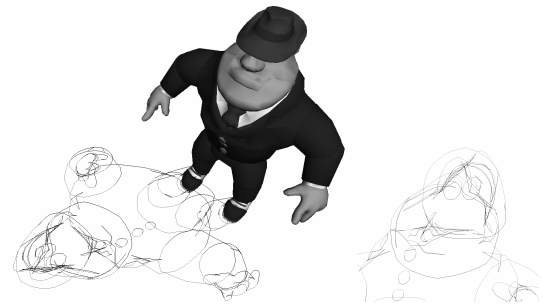

Figure 6 – Outline of Projected Geometry (magnification on right)

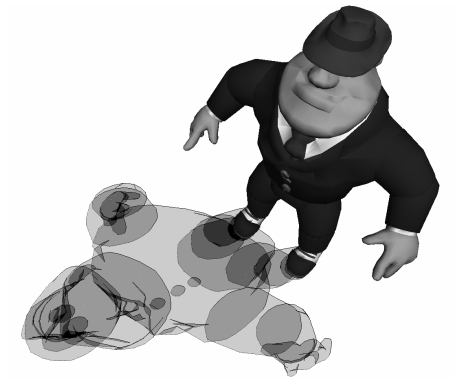

Figure 7 – Shadow areas color-coded according to winding number

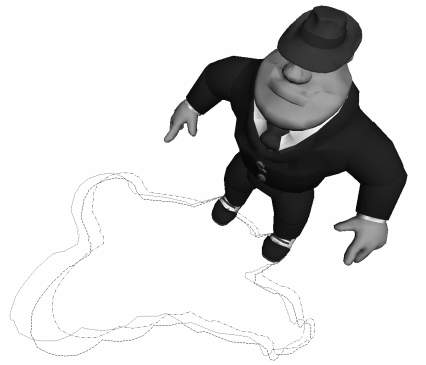

Figure 8 – Shadow outline caused by 3 point light sources

Figure 9 – Shadow with penumbras