CSG 96 Set-theoretic Solid Modelling: Techniques and Applications
Winchester, April 1996, pp 275-289

Rendering CSG scenes with general antialiasing

J-M. Hasenfratz D. Ghazanfarpour

Abstract

Ray-tracing is one of the most popular techniques for rendering 3D images. Effects such as shadows, reflection, refraction and so on can be produced with this technique. However, ray-tracing is a point-sampling technique with well-known aliasing problems. In particular, small objects and small shadows can be hidden between rays and not be detected. No ray-tracing method, even using oversampling, can solve this problem entirely. The solution is to use an extension of ray-tracing in which the concept of the infinitesimal ray is replaced by that of the beam, which has a volume of the scene. Beam-tracing is more complex than ray-tracing: in particular because of the beam-object intersection computations. So beam-tracers are usually limited to polygonal objects. The method presented here is a beam-tracer with no explicit beam-object intersection computations; so it can be used for rendering CSG scenes with antialiasing.

1. Introduction

Ray-tracing is one of the most interesting computer graphics algorithms for generating realistic images of three-dimensional scenes. It is easy to use it for rendering CSG models. However, ray-tracing is a point-sampling process: rays pass through the centre of pixels only, with a resulting aliasing problems. Some antialiasing techniques such as the classical over-sampling method [*WHIT 80*], or stochastic over-sampling [*COOK 86*], can solve the problems of “jaggies”. But all conventional ray-tracing techniques suffer from the possibility that small objects and shadows may disappear. This problem is due to the infinitesimal thickness of rays: some small

objects and shadows may simply have no intersection with the set of rays, so they cannot be detected. A solution to this problem is to replace the infinitesimal rays by beams, which have finite volumes.

Different beam-tracing algorithms are proposed [*AMAN 84*, *HECK 84*, *KIRK 87*, *SHIN 87*, *GHAZ 92*] with different shapes of beam: such as *cones*, *pencils* or *pyramids*. Most of these beam-tracers are limited to polygonal objects because of the complexity of ray-object intersection calculation for non-polygonal objects such as CSG models constructed from curved primitives.

The method presented here is the first beam-tracer with antialiasing compatible with CSG scenes. It is an extension of the method developed by one of the authors [*GHAZ 92*] for polygonal objects. The beams have *pyramidal*, *semi-pyramidal* or *pseudo-semi-pyramidal* shapes. They are considered as bounding volumes during the detection of aliased regions, especially for small objects. So, unlike other beam-tracers, no explicit beam-object intersection or 3D clipping is necessary. This characteristic allows us to use this new approach for rendering CSG scenes with antialiasing that accounts for small objects.

Our beam-tracer is first presented in the context of opaque models rendered without reflections. Theoretical solutions for reflections and refractions are studied later.

2. Beam-tracing for opaque scenes

As reported in an earlier paper [*GHAZ 92*], beams can eliminate jaggies, and detect small objects and shadows in CSG scenes. The method uses an adaptive recursive subdivision of the image space. In the first step, a region corresponding to the entire screen is considered. Then, in a second step, this region is recursively subdivided into four subregions (*Figure 1*). The recursive subdivision is stopped when a uniform region is found or when a maximum number of subdivisions is reached.

In a uniform region, there are no aliasing problems, so no subdivisions are required. Other regions are called *ambiguous regions* and do require subdivisions. When all ambiguous regions have been detected, a conventional ray-tracing is used to compute the colour of each pixel.

2.1 Vision pyramids

A *vision-beam* is a primary beam, traced from the eye into the scene (*Figure 1*). It is defined by the image region (the *base*) and four rays passing through the corners of the region (the *edges* of the beam). A list of primitive objects (spheres, cones, cylinders and polyhedra) is associated with each beam. These primitive objects may be partially or totally within the beam being considered. In this way, subdivision of a region produces four new sub-pyramids and four correspondingly updated sublists of primitive objects.

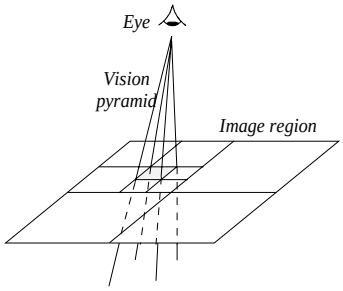
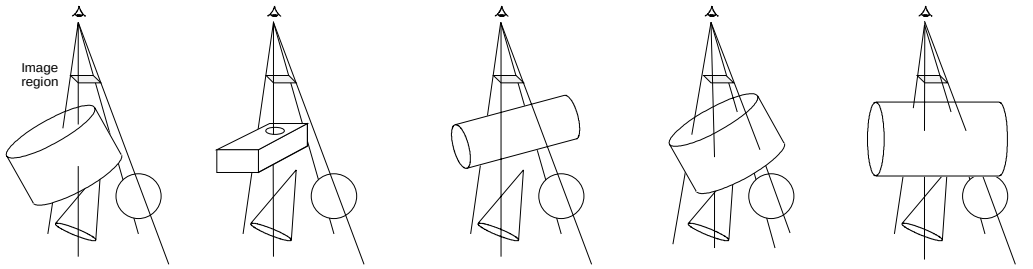


Figure 1: Adaptive recursive subdivision and vision-pyramid construction.

To determine the type of region, we first build a list containing all objects that can have an intersection with the beam corresponding to that region. To obtain this list, we find - for each object - a plane that divides space into two subspaces, one containing the whole of the beam and the other containing the whole of the object. If such a plane exists, it means that the object cannot be entirely or partially within the beam. So, this object is not added to the list. If we cannot find such a plane, the object is added to the list, as it may have an intersection with the beam.

We are going to discuss in *Section 2.3* how one may find a suitable plane and how to find easily the position of a primitive object relative to this plane. In order to detect whether an image region is uniform or not, a set of simple tests can be performed on the associated pyramid and its corresponding list of primitive objects. *Figure 2* shows some different examples of uniform and ambiguous regions.



1. Only two edges intersect an object. | 2. No intersection between edges and objects. | 3. Objects pass through the edges. | 4. Intersections with different sides of the same object. | 5. Intersections with the same side of an object.

Figure 2: Examples of uniform (5) and ambiguous (1, 2, 3, 4) regions.

- If the list of primitive objects is empty, the region is uniform.
- If it is not empty, four rays corresponding to the four edges of the pyramid are traced through the CSG scene. We considered for each ray the closest intersection points with an object, when it exist.
  - If these four intersection points are not in the same side of the same composite object (*Cases 1 and 4*), the region is ambiguous.
  - In the other cases, the same four rays are now traced through the associated list of primitive objects. Here, we do not consider CSG operations. If no ray intersects an object, there is an object in the beam (remember that the associated list is not empty), as in *Case 3*. If the four rays do not intersect the same number of objects and the same facets of each object, all in the same order, then the region is considered to be ambiguous (*Case 2*). In all other cases, the region is considered to be uniform (*Case 5*).

Some unnecessary subdivisions may be made by these tests but we notice that all aliased regions, in particular small objects and small

shadows, are detected. *Images 1 and 2* show the results of subdivision on three primitive objects.

2.2 Light pyramids

When the previous tests lead to a uniform region after the first step, then no further subdivision is required for the corresponding vision beam. This means that only one primitive object is visible in the pyramid. The intersection *ABCD* of this primitive object with the pyramid (*Figure 3*) is used to produce a light beam (secondary pyramid). A light beam is a pyramid that has *ABCD* as its intersection and a light source as its apex (*Figure 3*). As the vision pyramid, this beam is used exclusively as a bounding volume to determine if the region is uniform.

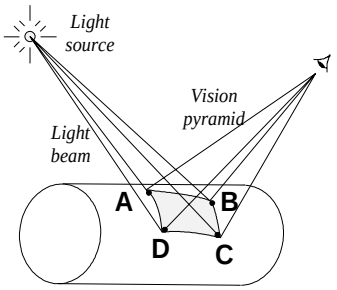


Figure 3: Light beam.

The visibility of *ABCD* from the light source is determined by new tests. The list of primitive objects associated with the beam is constructed in a similar way to the corresponding list for a vision pyramid. However, the intersection *ABCD* must be considered to determine the visibility. This base is approximated by a plane *P* so that all points on the intersected surface *ABCD* are in the light beam. This approximation allows us to define the light beam by five planes. This beam is a little bigger than the exact beam but includes all objects included by the exact beam.

For a convex intersected surface, three out of the four intersection points are considered (*A*, *C* and *D* in *Figure 4*) to define *P*. If the fourth point (*B* in *Figure 4*) and the light source are not on

the same side of *P*, we consider three other intersection points to define the desired plane. Such a plane allways exist and all points on the intersected surface *ABCD* are on the same side.

If the intersection is with a concave surface, then the plane *P* is tangent to the point *E* (*Figure 5*). Let *r* be a ray in the "middle" of the beam (*Figure 5*); it orientation is defined by the sum of the four vectors ***SA***, ***SB***, ***SC*** and ***SD***. *E* is the intersection point between *r* and the concave surface. Now, a light beam is defined by five planes. To determine the visibility of *ABCD*, three cases need to be distinguished:

1. *ABCD* is entirely visible from the light source, and thus it is directly illuminated;
2. *ABCD* is entirely hidden from the light source, and thus it is inside the shadow.
3. *ABCD* is partially visible from the light source.

As in the case of polygonal objects [*GHAZ 92*], in the first two unambiguous cases there is no problem: the region is uniform and no other subdivision is required. A new light source with its corresponding pyramid is considered. In the third case, the region is partially lighted, and then the image region is subdivided. In this case, we notice that visibility tests on the four new vision sub-pyramids are not necessary. Only tests on the four new light sub-beams are performed. Finally, a region is considered to be uniform if its vision pyramid and all its light pyramids are uniform.

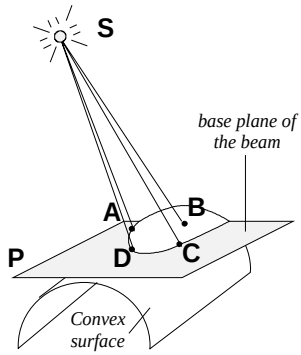


Figure 4: Convex intersected surface

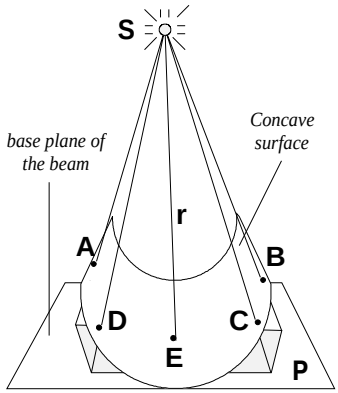


Figure 5: Concave intersected surface.

2.3 Position of primitive objects relative to a beam

A CSG model uses primitive objects combined by the union, difference and intersection operations. Generally, geometric entities are considered to be polyhedra, spheres, cylinders or cones. In this section, we consider curved primitives only; the technique for polyhedra is just an extension of the beam-tracing method for polygonal objects proposed by one of the authors in a previous paper [*GHAZ 92*].

In the foregoing sections, we have seen that it is necessary to know the position of a primitive object relative to a beam. This problem is fairly simple to solve. It consists of determining the position of primitive objects relative to each beam planes. For this, we have developed some simple tests that just require scalar and cross products.

2.3.1 Spherical objects

Simple tests can determine the position of a sphere, centred on *C*, relative to a beam. The first test consists of determining the position of a sphere relatively to each of the four beam planes. Each beam plane divides the space into two subspaces and the beam belongs only to one of these subspaces. If there is no possible intersection between the sphere and one of these planes, and if the sphere is not in the same subspace as the beam, then there is no possible sphere-beam intersection. In other cases, the position of the sphere relative to the beam will be determined by other tests.

The second test distinguishes between two cases, depending on whether the centre of the sphere is inside or outside the beam. In the first case, if the four edges of the beam intersect the sphere, then the sphere entirely blocks the beam. The region corresponding to the beam is uniform for this sphere and no subdivision is necessary.

If, on the other hand, one of the edges does not intersect the sphere, then it partially blocks the beam and we must use more sophisticated tests. The idea is to find a plane, *P* - if it exists - that divides the space into two subspaces: one containing the whole of the beam and the other containing the whole of the sphere (*Figure 6*). We consider two consecutive beam planes and their common edge *E*. Let

point *A* be the projection of the centre *C* on to the edge *E*. We consider the plane *P*, tangent to *E* and with ***n***=*AC* as normal vector (*Figure 6*). *P* is the desired plane if it has no intersection, either with the beam or with the sphere.

- To test whether *P* intersects the beam, we can compare the sign of two scalar products: ***n.n1*** and ***n.n2***. Where ***n1*** and ***n2*** are the normal vectors of the two consecutive planes of the beam. If both scalar products have different signs, then *P* intersects the beam. In this case, we consider two other consecutive beam planes to find another potential plane in the same way as *P* was obtained. If all these planes intersect the beam, there may be a beam-sphere intersection and then further subdivisions will be necessary. If both scalar products have the same sign, then *P* does not intersect the beam.
- If the sphere does not intersect *P* and it is not in the same subspace as the beam, then there is no possible beam-sphere intersection. In the other cases, we cannot determine the sphere position relative to the beam and we try to find the desired plane by considering two other consecutive beam planes. If none of these potential planes is appropriate, then it may be that there is no beam-sphere intersection and further subdivisions are necessary.

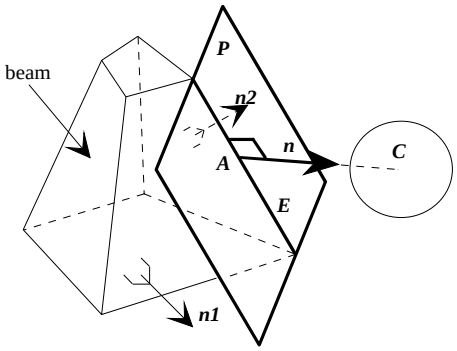


Figure 6: Plane *P* divides the space into two subspaces and the beam belongs only to one of these subspaces.

2.3.2 Cylindrical objects

The position of a cylinder relative to plane *P* can be determined in a similar way; but the tests are not as simple as the ones in the

previous section. A new test determines the position of cylinder bases relatively to *P* in the two following cases:

- If the axis of symmetry of the cylinder is perpendicular to *P*, we consider the centre of each base. If these two points are on different sides of *P*, the cylinder intersects *P*. Otherwise, the cylinder is entirely on one side of *P*.
- If the cylinder is not perpendicular to *P* then, for each base contour, we consider four points *Qi, i*∈[1,4] (*figure 7*). These points are intersections between each contour and the perpendicular plane to *P* containing centres of the cylinder bases. Let ***n1*** be a normal vector to a base, *B*, and ***n2*** be a normal vector to *P*. The cross product ***n3***=***n1***×***n2***, is the direction vector of the intersection between *P* and the plane of *B* (*Figure 7*). The cross product ***n4=n1***×***n3***, is a vector parallel to the plane of *B* and perpendicular to *P*. To find the four desired points, *Qi, i*∈[1,4], we just move the centre of the base of the cylinder by vectors ±*r*****n4***, where *r* is the radius of the cylinder (*Figure 7*). The position of the cylinder relative to *P* depends on the position of the four points *Qi, i*∈[1,4] relative to *P*. If these points are not in the same side of *P*, then the cylinder intersects it. In the other case, the cylinder is entirely on one side of *P*.

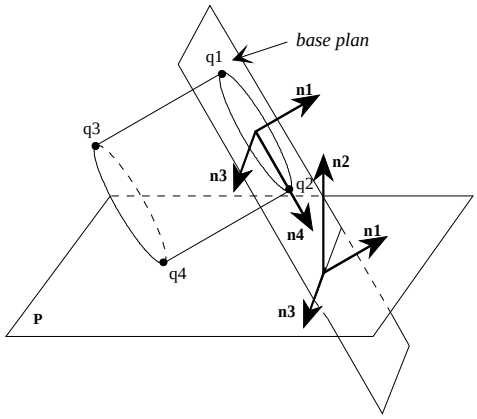


Figure 7: Position of a cylinder relatively to a plane *P*.

The position of a cylinder relative to a beam is determined in a similar way to that of a sphere. Only the determination of the beam

tangent plane for space subdivision (*Figure 8*) is different. Let ***d*** be the direction vector of the cylinder and ***n1*** and ***n2*** be the normal vectors of the beam planes being considered.

Two cases are distinguished, depending on whether the scalar products ***d.n1*** and ***d.n2*** have the same sign (*Figure 8.a*) or opposite signs (*Figure 8.b*). Let *A* be a point on the edge *E* that we are considering, and let ***v*** be the vector director of *E*. In the first case, the desired plane contains *A*, ***v*** and the cross product ***d*×*v***. In the second case, the plane contains *A*, **v** and ***d***.

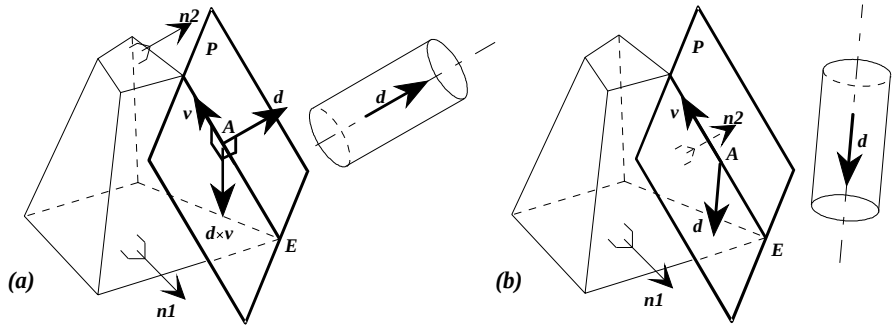


Figure 8: Position of a cylinder relatively to a beam.

2.3.3 Conical objects

The position of a cone relative to a plane is determined by a method very like that used in the previous cases. First, we determine the two points *Q1* and *Q2* on the contour base of the cone exactly in the same way as for a cylinder. Then we determine the position of points *Q1*, *Q2* and the apex of the cone relatively to the plane being considered. If these points are not on the same side of the plane, the cone intersects it; otherwise, the cone is entirely on one side of the plane. The position of a cone relatively to a beam can be determined in a similar way to the position of a cylinder.

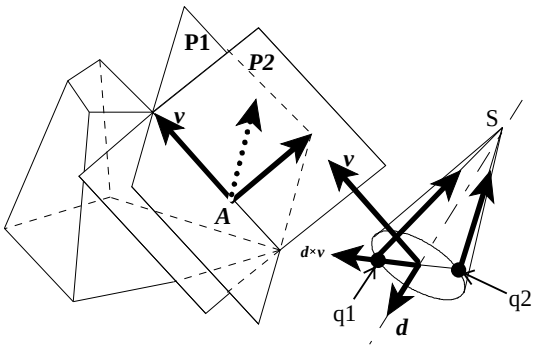


Figure 9: Position of a cone relatively to a beam.

Consider two points *Q1* and *Q2* on the contour of the cone base (*Figure 9*). These points are obtained by moving the centre of the base by the vectors $\pm r*\boldsymbol{d}\times\boldsymbol{v}$, where *r* is the radius of the base of the cone, *d* is its direction and ***v*** the direction vector of the considered edge. Let *A* be a point on the edge *E* and let *S* be the apex of the cone. Each of two planes, *P1* and *P2*, can divide the space into two subspaces so that the beam is entirely in one subspace and the cone is entirely in the other subspace. *P1* contains *A*, ***v*** and *Q1S* and *P2* contains *A*, ***v*** and *Q2S*. We use two planes for each edge to determine the position of a cone relative to a beam in a similar way as for a cylinder.

2.4 Colour of pixels

The goal of the previous sections was the detection of jagged edges, small objects, and small shadows corresponding to ambiguous regions. As the adaptive recursive subdivision of the image space progresses, a tree of beams is built. This tree is similar to those used by [*HECK 84*]. It contains full information about the path from the eye to the uniform regions or the smallest non-uniform regions. In particular, it contains the objects intersected by each beam. For each uniform region and smallest ambiguous region, conventional ray-tracing is used to compute the colour of each region. Note that the tree of beams allows us to anticipate objects to be intersected by each ray. Three different types of regions are considered:

- uniform regions greater or equal than a screen pixel;
- uniform regions smaller than a screen pixel;

- ambiguous regions (smaller than a pixel).

In the first case, a uniform region corresponds to one more pixels, and no pixel subdivision is needed. Four rays passing through each pixel corner are traced. We notice that with this method each ray is common to four neighbour pixels and so the method is not actually more expensive than tracing rays through pixel centres. The colour of a pixel is the average of the colours encountered by the four rays.

In the second case, a uniform region corresponds to one or more sub-pixels. Computing the colour of a pixel is very similar to the previous case. Instead of four rays per pixel, there are four rays which pass through the corners of each sub-pixel. The final colour of a screen pixel is the weighted average of the colours of the sub-pixels.

In the third case, a non-uniform region corresponds to one or more sub-pixels. Four rays passing through the corners of each sub-pixel are traced. The first objects intersected by these rays average the colour of the region.

3. Reflection and refraction

So far, we have presented principles of beam construction and adaptive recursive subdivision for opaque scenes without reflections or refractions. The extensions of this beam-tracer to CSG scenes with illumination models incorporating reflection and refraction will now be studied. These two extensions are still under development.

Reflection in a CSG scene is more complex than in a polyhedral scene because the reflecting surfaces are not planar. In this type of scene, concave and convex surfaces can be encountered. The result of a beam reflected on such a surface is not a pyramid, and we must characterize it in another way. The goal is to find a volume that bounds all the rays reflected from a non-planar surface. This is an extension of the bounding-volume concept of refracted rays in the case of a polygonal object, as presented in a previous paper [*GHAZ 92*].

Consider *A*, *B*, *C* and *D*, the intersections of the edges of a vision pyramid with a reflecting primitive object, and the reflected rays *r1*, *r2*, *r3* and *r4* in *A*, *B*, *C* and *D* (*Figure 10*). Intersected

surfaces can be concave or convex; this depends on the CSG operations used. The construction of the reflected beam for these two cases is slightly different.

First consider the case of a convex surface. Each of lines *AB*, *BC*, *CD* and *DA* can be combined with a reflected ray to form a plane. For example, in *Figure 10*, line *BC* and reflected rays *r2* form plane *P1* and line *BC* and reflected ray *r3* form plane *P2*. One of these planes is between *ABCD* and the other plane (*P1* in this example). The outer plane (*P2* in this example) and the three other similar planes are used to construct the reflection volumes.

For a concave surface (*Figure 11*), we can consider four planes for each lines of *AB*, *BC*, *CD* and *DE*. Each of these planes is oriented by one of the reflected ray *r1*, *r2*, *r3* or *r4*. Thus we obtain sixteen planes and take the four outer planes to construct the reflected beam on a concave surface. The base of a reflected beam is defined, as for light beams, in terms of a function of concavity (*Section 2.2*). These *reflection pseudo-semi-pyramids* contain all the reflected rays on a concave surface. Thus, we have defined a bounding volume of all reflected rays.

We can now use previous tests to determine if the region is ambiguous for this level of reflection and if more subdivisions are necessary. All these subdivisions allow us to detect non uniform regions due to the reflection. For each level of reflection, new reflection beams are traced and the tree of beams is updated. The colour of pixel is determine as for the previous case using conventional ray-tracing (*Section 2.4*).

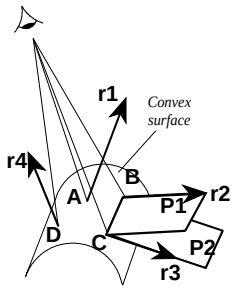
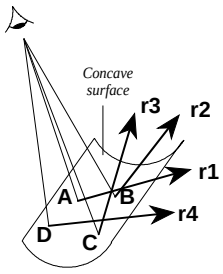


Figure 10: reflection on a convex surface.



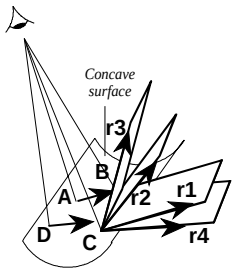


Figure 11: reflection on a concave surface.

The case of the refraction is similar to that of the reflection. The goal is to define a volume bounding all refracted rays. The *refracted pseudo-semi-pyramids* are constructed in the same way as reflected semi-pyramids. Instead of using four reflected rays, we use four refracted rays. To determine if the region is ambiguous, the tests developed for a light beam are used. If the region is ambiguous, subdivisions are necessary and only refracted semi-pyramids are subdivided in four. If the region is uniform, the colour of pixels is computed using the updated tree of beams.

5 Conclusion

In contrast to conventional ray-tracing, our beam-tracing resolves the aliasing problem of small objects and shadows. *Image 3* shows a CSG scene with no reflection and refraction rendered by ray-tracing (top) and with our beam-tracer (bottom). The two zoomed views show that beam-tracing solves the problems of small objects and small shadows. This beam-ray-tracing method can be considered as a first step to an efficient antialiasing technique for CSG scenes. Antialiasing is general and, in particular, small objects and small shadows are detected precisely.

A pyramidal beam is used as a bounding volume to detect uniform and ambiguous regions. No explicit beam-object intersections are necessary, this allows us to use CSG scenes in contrary of the other beam-tracers. Theoretical problems of reflection and refraction are solved and the implementations of these two cases are now under development.

References

[AMAN 84] J. AMANATIDES, “Ray Tracing with Cones”, Computer Graphics, vol. 18, No. 3, July 1984, pp. 129-145.

[COOK 86] R. COOK, “Stochastic Sampling in Computer Graphics”, ACM Transactions on Graphics, vol. 5, No. 1, January 1986, pp. 51-72.

[DIPP 85] M. DIPPE and E. WOLD, "*Antialiasing Through Stochastic Sampling*", Computer Graphics, vol. 19, No. 3, July 1985, pp. 69-78.

[GHAZ 92] D. GHAZANFARPOUR, "*Visualisation réaliste par lancer de pyramides et subdivision adaptative*", proceedings of MICAD 92, PARIS, February 1992, pp.167-181 ;

[HECK 84] P. HECKBERT and P. HANRAHAN, "*Beam Tracing Polygonal Objects*", Computer Graphics, vol. 18, No. 3, July 1984, pp. 119-127.

[KIRK 87] D. KIRK, "*The Simulation of Natural Features Using Cone Tracing*", The Visual Computer, Springer-Verlag, vol. 3, No. 2, 1987, pp. 63-71.

[SHIN 87] M. SHINYA, T. TAKAHASHI and S. NATIO, "*Principals and Applications of Pencil Tracing*", Computer Graphics, vol. 21, No. 4, July 1987, pp. 45-54.

[WHIT 80] T. WHITTED, "*An Improved Illumination Model for Shaded Display*", Communication of the ACM, vol. 23, No. 6, June 1980, pp. 343-349.

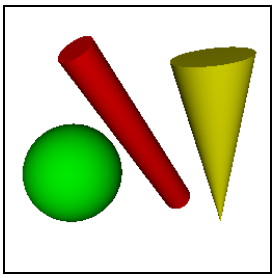


Image 1: Three primitive objects.

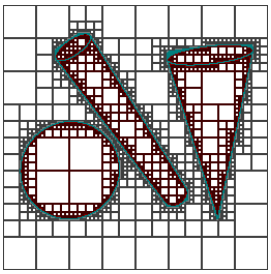


Image 2: Adaptative subdivision. All squares correspond to a uniform region.

Sorry, colour image is not available!

Image 3: A simple CSG model containing small objects and shadows illuminated without reflection or refraction. **Top**: ray-tracing without antialiasing, and zoomed view. **Bottom**: beam-tracing with antialiasing, and zoomed view.