Un nouvel algorithme de subdivision spatiale adapte a la méthode du lancer de rayon et utilisant un modèle CSG. A new algorithm of space tracing using a CSG model.(en anglais)

Kadi Bouatouch, Mohamed Ouali Madani, Thierry Priol, Bruno Arnaldi

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 612

UN NOUVEL ALGORITHME DE SUBDIVISION SPATIALE ADAPTÉ À LA MÉTHODE DU LANCER DE RAYON ET UTILISANT UN MODÈLE CSG

Kadi BOUATOUCH
Mohamed Ouali MADANI
Thierry PRIOL
Bruno ARNALDI

Février 1987

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

UN NOUVEL ALGORITHME DE SUBDIVISION SPATIALE ADAPTE A LA METHODE DU LANCER DE RAYON ET UTILISANT UN MODELE CSG

A NEW ALGORITHM OF SPACE TRACING USING A CSG MODEL

Kadi **BOUATOUCH**
Mohamed Ouali **MADANI**
Thierry **PRIOL**
Bruno **ARNALDI**

Publication Interne n° 335
Janvier 1987
16 pages

Abstract : This paper describes a new algorithm of space tracing. Scenes are modeled by a CSG tree. Space is subdivided regularly into 3D regions called boxes. With each box is associated a subtree which is the restriction of the whole scene CSG tree to primitives belonging to this box. A 3D grid is used to access boxes.

Résumé : Cet article décrit un nouvel algorithme de lancer de rayon adapté à une nouvelle technique de subdivision spatiale. Le modèle de description de scènes est du type CSG. L'espace est subdivisé en régions 3D appelées boîtes. A chaque boîte est associé un petit sous-arbre qui est la restriction de l'arbre CSG décrivant toute la scène aux primitives géométriques contenues dans cette boîte. Une grille 3D est utilisée pour accéder aux boîtes se trouvant dans la direction d'un rayon.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

A NEW ALGORITHM OF SPACE TRACING USING A CSG MODEL

1 Introduction

It is acknowledged that ray tracing is a powerful technique for rendering the most realistic images by computer. The drawback of this method is the CPU time it needs for creating even one image of moderate complexity. To reduce this CPU time, different methods have been proposed in the litterature and all use either 2D or 3D subdivision.

2D subdivision

Bronswoort et al (2) propose a method which refines the image by subdivision in macropixels, thereby avoiding explicit computation of the intensities of all the pixels of the image. This method presents the disadvantage of losing slivers of the image. Sears et al (9) algorithm consists in using the rectanglar enclosures of the primitive objects (which result from the projection of their parallelepipedic enclosures on the screen plane) to subdivide the screen into regions. A subtree is thereby associated with each region. This technique minimizes only ¡ nary rays intersections. Like Sears et al, Coquillart (3) proposes to project the parallelepipedic enclosures of primitives on a plane which may be different from the screen plane, and subdivides this plane into regions. The difference with the previous method is that the regions may overlap. Two graphs of connexity and overlapping are deduced. This technique works for both primary and secondary rays. Since this method uses a plane projection, the drawback is that the spatial coherence is

not fully exploited and intersection computation is performed for all the primitives whose rectangular enclosures lie along the ray path.

3D subdivision

The parallelepipedic enclosure of the scene (whose faces are perpendicular to the view coordinate system axes) is subdivided into 3D regions containing a small number of objects. A primary or secondary ray which enters a region, intersects only those objects lying in this region. If no intersection is found or the objects intersected are all transparent (in the case of rays directed towards the light sources), a computation of the next region traversed by the ray, is performed. It is shown that space subdivision reduces considerably the amount of intersection computations.

Many authors have proposed different techniques of space subdivision reducing the synthesis time of complex images. However all these methods lead us to make the following remarks :

- Glassner (6) subdivides space in octree and uses a hashing table to access a leaf (voxel). The determination of the next region traversed by a ray, requires the traversal of all the octree structure and comparisons with voxel faces. This search is expensive in time. The kind of modelling system used , is not specified.

- Kaplan (7) uses the objects enclosures to subdivide regularly the parallelepipedic bounding volume of the scene by planes perpendicular to the coordinate system axes, according to the BSP technique (Binary Space Partioning) (4). The data structure obtained is a binary tree whose leaves are 3D regions. Each region contains a minimal number of objects. The traversal of all the tree stucture is needed to determine the region containing a point. For a scene of many tens or hundreds of objects, this traversal becomes very time consuming. For a CSG model, Kaplan proposes to compute the intersection of a ray with the primitive objects lying in all the regions traversed by the ray. We show later, that it is not necessary to consider all these regions for computing the nearest intersection point.

- Fujimoto et al (5) use a spatial enumeration structure called SEADS and a 3D differential analyser called 3DDDA, to determine the next region in the direction of a ray. They give many details about the traversal of the octree structure by means of the 3DDDA and unfortunately no precisions about the SEADS structure they indeed use.

- Wyvill and Kunii (11) choose a DAG (Direct Acyclic Graph) to model a scene. The result of their spatial subdivision is an octree. Finding a voxel containing a point, requires the traversal of all the octree structure. The authors impose a restriction : two primitive objects must not overlap. That limits the creation of realistic and complex objects.

We propose in this paper a new technique of space tracing using a CSG model with no restriction about the overlapping of primitive objects. With each region resulting from the subdivision is associated a subtree. The access to a region does not require the traversal of a data structure like octree or BSP tree.

In section two we give an overview of our method and detail the subdivision process in section three. The next section concerns the ray tracing technique using our spatial subdivision. Section five contains results and discussion. The last section is reserved to the conclusion.

2- Overview of the algorithm

The main characteristics of the algorithm are the following :

- Scenes are modeled by a CSG tree. Primitive objects may be solids or surfaces and the boolean operators are union, intersection and difference [1].

- For each primitive, a minimal bounding volume is computed.

- Like in Kaplan's method, the parallelepipedic enclosure of all the scene is subdivided recursively by planes aligned with the eye coordinates system axes. Each slicing plane divides a space into two equal subspaces [figure 3]. The result is not a BSP tree but only its leaves which represent 3D regions that we call boxes. A box is either empty or contains a subtree which is the projection of all the scene. A number is associated with each box.

-A spatial index (that we call SI) which is a 3D grid, is used to determine rapidly the box containing a point in the subdivided space. Each component SI[i,j,k] of the grid is an integer representing the number of a box.

- Given a ray, the box including its origin is found and intersection with the subtree associated is performed.

- If a ray fails to hit any primitive in a box, it must move to the next box lying in its direction. The algorithm of finding the next box is nearly similar to Glassner's one.

3 Space subdivision

3-1 Subdivision step

Before describing the spatial subdivision process, we explain the concept of minimum primitive enclosure we use to find the list of primitives included in a box. A minimum enclosure bounds the part of the primitive which is effectively used. Let us consider figure 1 where two primitives P1 and P2 are combined by an intersection operator.

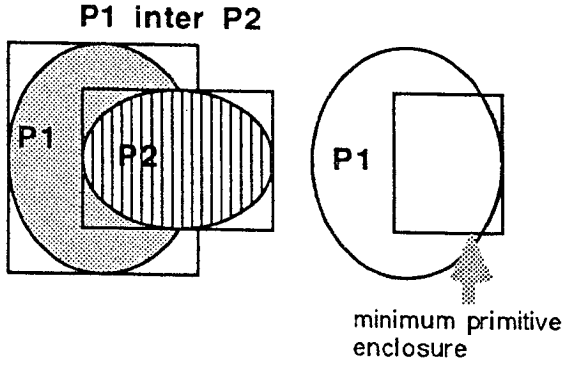
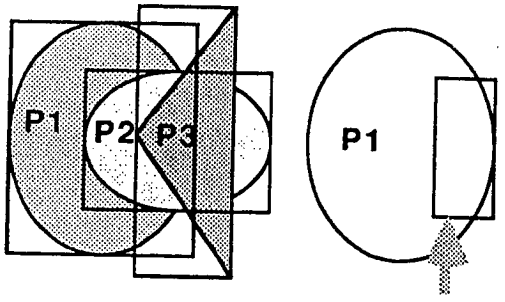


figure 1. minimum enclosure of primitive P1.
Case of two primitives.

We see that it is not necessary to keep the whole enclosure of P1 since its minimum enclosure is sufficient to test if it intersects a box. Let us add a third primitive to the previous example, as it is shown in figure 2. Three primitives P1, P2 and P3 are combined by means of two intersection operators. We obtain a minimum enclosure of P1, smaller than this of figure 1.



P1 inter P2 inter P3 minimum primitive enclosure

figure 2. minimum enclosure of primitive A.
Case of three primitives.

The algorithm of computing the minimum enclosure of all the primitives of the CSG tree, is the following:

for each leaf P of the CSG tree **do**
begin
    B := **P**
    repeat
        if B enclosure is included in this of the parent node of P
            then do not change it
            else replace it by its intersection with the enclosure of the parent node of P
        replace P by its parent node
    until (P is the root)
end { end of loop }

We obtain then a list of minimum primitive enclosures which are used for space subdivision.

As precised previously, the space to subdivide is the enclosure of the scene which is a parallelepiped whose faces are aligned with the eye coordinate system axes. This space is sliced by planes perpendicular to x,y and z axis. Each slicing plane divides a space (or box) into two subspaces (or boxes) of equal dimensions. The subdivision process stops either when a box is intersected by a minimum number of minimum primitive enclosures, or the maximum level of subdivision is reached with respect to each axis. These two criteria used to stop the subdivision are chosen by the user. This subdivision is similar to the one proposed by Kaplan. The difference is that the BSP tree structure is not saved. We keep only its leaves which constitute a linear array of boxes. With each box is associated a list of primitive objects whose

minimum enclosures intersect it . Figure 3 illustrates a 2D example of subdivision where the scene contains four primitives A,B,C and D.

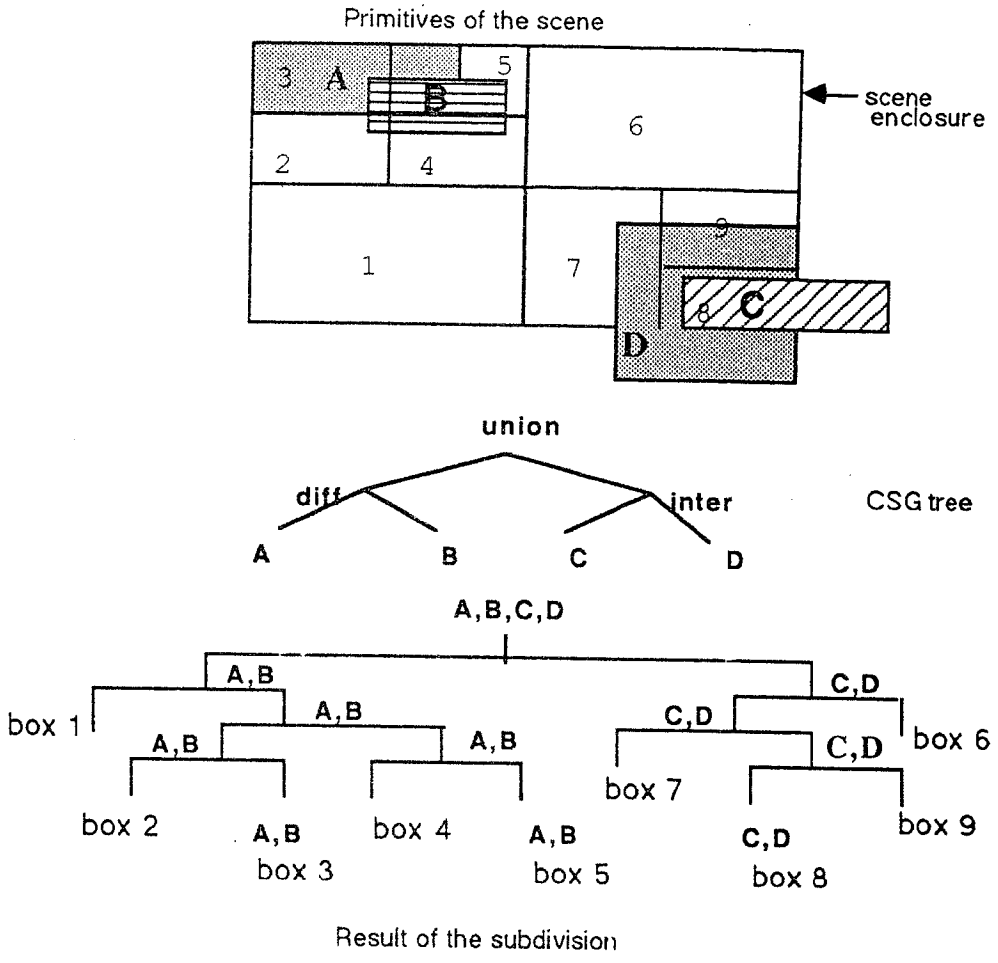


Figure 3. subdivision process, nine boxes are found

3-2 Projection of the CSG tree representing all the scene

Given a list of minimum enclosures, the purpose is to project the scene CSG tree on each box; that is to restrict it to the primitive objects of the box. The result of this projection is a subtree. This restriction is made by means of simple rules which are as follows .

Consider a node N and the two restricted subtrees NL and NR associated with N and combined by the boolean operator ope. NL and NR are respectively the left and right subtrees. Two cases are considered :

- the node is a primitive P (leaf) : if P belongs to the box then the node projection is P, else it is null(null means that the box does not contain the object).

- otherwise :

if NL and NR are not null then projection(node) = NL ope NR

if NL and NR are null then projection(node) = null

if NL is null and NR not null then there are three cases :

if ope = union then projection(node) = NR

if ope = difference then projection(node) = null

if ope = intersection then projection(node) = null

if NL is not null and NR is null then there are three cases :

if ope = union then projection(node) = NL

if ope = difference then projection(node) = NL

if ope = intersection then projection(node) = null

A recursive procedure traverses the whole scene CSG tree to apply these restriction rules.

3-3 Data structure of a box

A box is represented by a structure constituted by the following fields :

- **address** : a three integer components vector necessary to compute the 3D grid SI.
- **level** : a three integer components vector indicating the subdivision level reached according to the three coordinates axes.
- **size** : a six components vector representing the position of the box in space : xmax, xmin, ymax, ymin, zmax, zmin.
- **ptrtree** : a pointer to a subtree.
- **penclosure** : parallelepipedic enclosure of this CSG tree.
- **senclosure** : spherical enclosure of this CSG tree.
- **empty** : boolean taking the true value if there are no primitives in the box.

Before the subdivison process, the address of all the space to subdivide is [1,1,1]. Let C be the point of a box having the smallest coordinates . During subdivision step, each address component named address.w corresponding to the slicing plane perpendicular to w axis(where w is either x or y or z) progresses as follows:

- address.w of the sub-box containing C=(address.w of parent box)*2
- address.w of the other sub-box=(address.w of parent box)*2+1

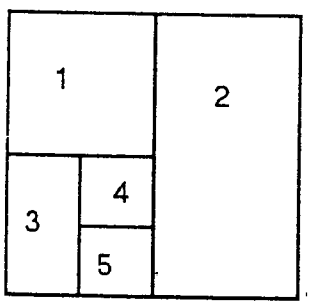
3-4 Construction of the 3D grid SI

Once the space subdivision in boxes is accomplished, the construction of the 3D grid SI is performed according to the following algorithm written in pseudopascal:

```
algorithm
{ let i,j and k be the three indices of the grid and B a box }
for each box B do
begin
    if B.level = max { the box has a minimum size }
        then  begin
                i := B.address.x +1
```

```
            j : = B.address.y + 1
            k : = B.address.z + 1
            Sl[i,j,k] := box number associated with B
          end
        else { the size of B is not minimum }
            subdivide B into minimum size boxes
            {each of such boxes corresponds to an element of the grid }
            {compute the corresponding indices as follows }
          for xx := 0 to 2^(levelx-B.level.x)-1  do
            begin
              i : = B.address.x *(2^(levelx-B.level.x))+xx+1
              for yy := 0 to 2^(levely-B.level.y)-1 do
                  begin
                    j : = B.address.y*( 2^(levely-B.level.y))+yy+1
                    for zz :=0 to 2^(levelz-B.level.z)-1 do
                      begin
                        k := B.address.z*( 2^(levelz-B.level.z))+zz+1
                        Sl[i,j,k] := box number associated with B
          { levelx,levely and levelz are  the maximum levels of subdivision with respect to
            x,y and z axis }
                      end
                  end
              end { of loop xx }
end {of loop  box }
```

A 2D example of construction of the grid is illustrated by figure 4.



Result of subdivision

1	1	2	2
1	1	2	2
3	4	2	2
3	5	2	2

Associated grid

figure 4. 2D construction of a grid

4 Ray tracing

Given a ray, the search of the box containing its origin is first performed. If the ray fails to intersect

primitive objects of the box, it moves to the next box lying in its direction. This process is reiterated until no intersection is found or the ray leaves the scene.

4-1 search of the box containing a point P(x,y,z)

Let vxmax, vxmin, vymax, vymin, vzmax, vzmin, be the six planes bounding the scene. The contents of element grid SI[i,j,k] represent the number of the box containing P(x,y,z). The three indices i, j and k are computed as follows :

i=trunc((x-vxmin)/(vxmax-vxmin)*2^levelx)+1

j=trunc((y-vymin)/(vymax-vymin)*2^levely)+1

k=trunc((z-vzmin)/(vzmax-vzmin)*2^levelz)+1

If the observer does not belong to the subdivided space, it is easy to find the first box traversed by a primary ray. In this case the third index k of the 3D grid is zero.

4-2 search of the next box

A ray is defined by its following parametric equations :

$x=t*dx+x_0$

$y=t*dy+y_0$

$z=t*dz+z_0$

Where t is the scalar parameter, $O(x_0, y_0, z_0)$ is the ray origin and (dx, dy, dz) its direction vector.

The search algorithm is nearly the same as Glassner's one except that only three ray-face intersections are needed instead of six :

- According to the sign of dx, dy and dz, we compute the intersection of the ray with the three faces of box B containing its origin : xmin or xmax, and ymin or ymax, and zmin or zmax. We obtain then three values of t.

- The smallest value of t corresponds to the point P where the ray leaves box B. We compute then its coordinates X,Y,Z.

- If P belongs to the face of B perpendicular to x or y or z axis, we add or substract to x, y or z component of P, respectively a value deltax, deltay or deltaz which is equal to half the length of x,y or z side of a minimum size box . We obtain then a point P' and the box containing it. P' is only used to find the next box in the ray path. The mechanism consits in fact in pushing a point P in the direction of a face normal.

- If P is on an edge or a vertex, we push it in the direction of the normals of faces sharing it.

Figure 5 illustrates this search algorithm.

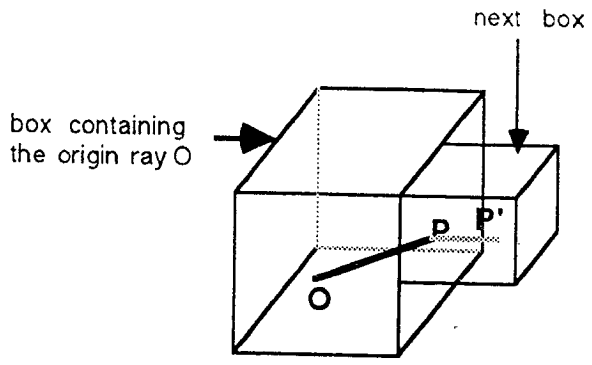


figure 5. research of the next box

4-3 Intersection

Two bounding volumes are associated with each box : a rectangle for filtering primary rays and a sphere to minimize secondary rays intersections. The bounding volumes enclose the subtree contained in the box. Intersection calculation is similar to the one proposed by Roth [8]. The difference is that the subtree is small and only its root has a bounding volume.

A primitive object may be shared by several boxes as shows figure 6. To avoid computing repeatedly the same intersection of a ray with this kind of primitive, a mail box is associated with each primitive of the scene CSG tree and an unique number is associated with each ray. When an intersection with a primitive is performed, we put the ray number and the intersection result in its mail box. Then if another intersection with this same primitive has to be performed, we test if the current ray has the same number as this being in the mail box. If it is the case, we do not need to perform intersection since result is already in the mail box. Otherwise intersection is performed and result is put in the primitive mail box.

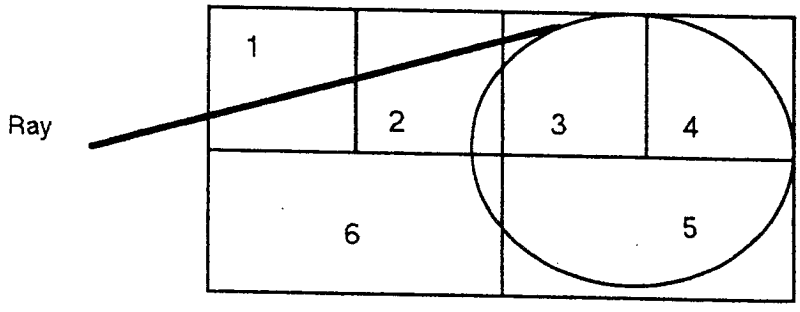


figure 6. example of a ray intersecting boxes containing the same primitive.

5 Results and discussion

The space tracing algorithm presented in this paper, has been implemented in Pascal on a VAX 11/750 computer. Tests have been performed for three kinds of scenes called IMA1, IMA2 and IMA3. These scenes are described in table 1 and represented by figures 7,8 and 9.

image	IMA1	IMA2	IMA3
number of primitives	32	72	12
number of light sources	3	1	2
depth of rays tree	1	1	5
types of object	sphere cylinder cube cone	sphere cylinder	sphere cylinder cube

table 1 : characteristics of test images

Table 2 compares the traditional ray tracing to the new space tracing algorithm. The computation time ratio is 4 for a 12 primitives scene and 25 for a 72 primitives scene. This gain may be more important for more complex images. Another interesting result given by table 2 is that preprocessing time (space subdivision) is very small.

image	IMA1		IMA2		IMA3	
resolution	512*512		512*512		256*256	
spatial subdivision	no	yes	no	yes	no	yes
maximum sudivision level		4		4		3
maximum number of primitives per box		2		2		3
number of boxes		251		549		34
subdivision time		6s		17s		1s
synthesis time	13h 54mn	47mn	13h 44	47mn43	7h 38mn	2h
gain factor		17.74		25.65		3.78

table 2 : comparison of the traditional and new ray tracing algoritmms

Table 3 presents the results of statistical tests performed on the three kinds of images. the mail boxes seem very efficient since they contain 40 to 50 per cent of the intersection results. Thereby a lot of intersection computations are avoided. The mean number of calls to the next box procedure seems small for both primary and secondary rays but a little larger for rays directed towards light sources, in case of scenes containing transparent objects. In order to prove that with our algorithm, synthesis time does not depend on primitive objects number but on the type of primitives, we have chosen scene IMA2 to which we have removed 36, 48, 60 and 64 primitives to get scenes having a variable number of primitives of the

same kind.

image	IMA1		IMA2		IMA3	
resolution	256*256		256*256		256*256	
subdivision level	3	4	3	4	3	4
number of primitives per box	2	2	2	2	2	2
number of boxes	123	251	279	549	86	260
mean number of calls to next box procedure :						
. primary rays	2.49	2.57	3.11	3.21	2.92	3.28
. secondary rays					2.5	2.98
. rays towards light sources	3.14	3.39	2.62	2.8	4.19	5.25
efficiency of mail boxes in per cent	39.6	41.7	39.8	39.97	4.19	50.2
synthesis time	14mn 27s	13mn 24	18mn 26s	16mn		
subdivision time	3s 50	6s 58		19s		

table 3 : statiscal results

Table 4 shows that for these images, the mean time synthesis per ray is relatively constant for space tracing but grows rapidly for traditional ray tracing. Also, it shows that the gain factor in synthesis time grows with primitive objects number. Other tests have shown that sometimes, increasing subdivision level is not necessary since it increases synthesis time instead of decreasing it. This can be explained by the increase of calls number to next box procedure which is due to a larger number of boxes.

image	IMA21	IMA22	IMA23	IMA24	IMA25
primitives number	72	36	24	12	8
mean time synthesis per ray, case of traditional ray tracing	177	93	75	57	52
mean time synthesis per ray, case of space tracing	10.1	7.68	8.1	8.5	8
gain factor	18.7	12.1	9.3	6.75	6.5

table 4 : influence of primitives number

6 Conclusion

With the new space tracing algrithm , computation time is decreased by a factor of 25 for a scene having few primitives. We assume that for more complex scenes of many hundreds of primitives, this factor may be very important. Scenes are represented by a CSG tree which is a simple and efficient fashion to modelize objects exactly. In our algorithm, time consumed by the search of the next box lying along the ray path, is constant whatever the image complexity, whereas it is variable for space subdivision methods needing the traversal of a data structure and thereby it may be very important for complex scenes. Minimum primitive enclosures and mail boxes contribute efficiently to the increase of gain factor. This gain is obtained at the expense of a more important memory occupation due to boxes array and 3D grid creation. A technique to minimize this occupation is to reduce the grid size by using sub-grids as proposed by Tammimen et al(10) for performing boolean operations on polyhedral objects. These sub-grids are determined as follows :

- choose small values for the maximum subdivision levels associated with the three axes.
- during spatial subdivision step, when maximum subdivision levels are reached, a sub-grid is associated with each box containing many primitives.

Thereby, two grid levels are created. An element of the first level grid may point to either a box or a subgrid which points in its turn to a box.

7 Acknowledgements

We are grateful to Christian Bouville and Roger Brusq for some helpful suggestions. We thanks also Patrice Quinton for his reliable advice. This work has been supported by the CCETT under contract no 86CN08.

8 Bibiography

[1] Bouatouch, K. , Arnaldi, B. , Priol, T. , LGRC : un langage pour la synthèse d'image par lancer de rayon, TSI no 6 (1986).

[2] Bronsvoort, F. , Jarke, J. V. , Jansen, F. W. , Two methods for improving the efficiency of ray casting in solid modelling, CAD vol 16 no 1 January (1984).

[3] Coquillart, S. , An improvement of the ray tracing algorithm, Eurographics (1985)

[4] Fuchs, H. , Redem, Z. M. , Naylor, B. F. , On visible surface generation by a priori tree structure, ACM (1980)

[5] Fujimoto, A. , Tanaka, T. , Iawata, K. , ARTS : Accelerated ray tracing system, IEEE CG &A April (1986) 16 - 26.

[6] Glassner, A. S. ,Space subdivision for fast ray tracing, IEE CG&A October (1984) 15 - 22.

[7] Kaplan, M. R. , Space tracing, a constant ray tracer, SIGGRAPH 85 tutorial on the uses of spatial coherence in ray traycing.

[8] Roth, S. D. , Ray casting for modelling solids, Computer Graphics and Image processing no 18 (1982) 109 - 144.

[9] Sears, K. H. , Middleditch, A. E. , Set-Theoretic Volume Model Evaluation and Picture plane Coherence, IEEE CG&A March (1984).

[10] Tammimen, M. , Karonen, O. , Mantyla, M. , Ray casting and block model conversion using spatial index, CAD vol 16 no 4 July (1984) 203 - 208.

[11] Wyvill, G. , Kunii, T. L. , A functional model for constructive solid geometry, The Visual Computer no 1 (1985) 3 - 14.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique