MEMORY SAVING TECHNIQUE FOR SPACE SUBDIVISION TECHNIQUE

Václav Skala
Computer Graphics and Scientific Visualization Laboratory
Department of Informatics and Computer Science
University of West Bohemia, Plzen, Czech Republic

Abstract. The ray tracing technique is very often used for image synthesis because it gives the possibility to render specular objects. Many techniques have been developed for ray tracing acceleration, more or less sofisticated which, are generally speaking, not easy to implement. A simple method how to speed up the primary and secondary ray tracing has been developed. The suggested method based on space subdivision method (not necessarily uniform) is convenient for scenes that consist of many small objects, resp. facets (for experiments only triangles have been used).

Key words: ray tracing, acceleration, data structure, rendering, computer graphics, algorithm complexity

1. Introduction

It is well known that ray tracing is the only one method which is capable of rendering specular effects like reflection through an object and refraction, even some other methods do integrate some features of ray tracing in order to handle these effects.

The bottleneck operation of ray tracing is *the search for the* **first** *object intersected by a given ray.*

There are many techniques based on many sofisticated algorithms [1], like space subdivision methods (uniform, non-uniform, adaptive), octree methods, etc.

The proposed Binary Map of Space Subdivision Method (BMSSM) is based on simple presumptions:

– scene consists of dozens of small objects with regard to the scene volume,

– scene consists of all kinds of objects, like polyhedrons, solids, given by an implicit functions $F(x, y, z) = 0$ or by parametrically defined patches, CSG trees etc.,

2. Principle of the proposed method

Let us suppose the parallel primary ray tracing and that the image resolution is $n \times m$ pixels and that scene consists of p facets, resp. objects.

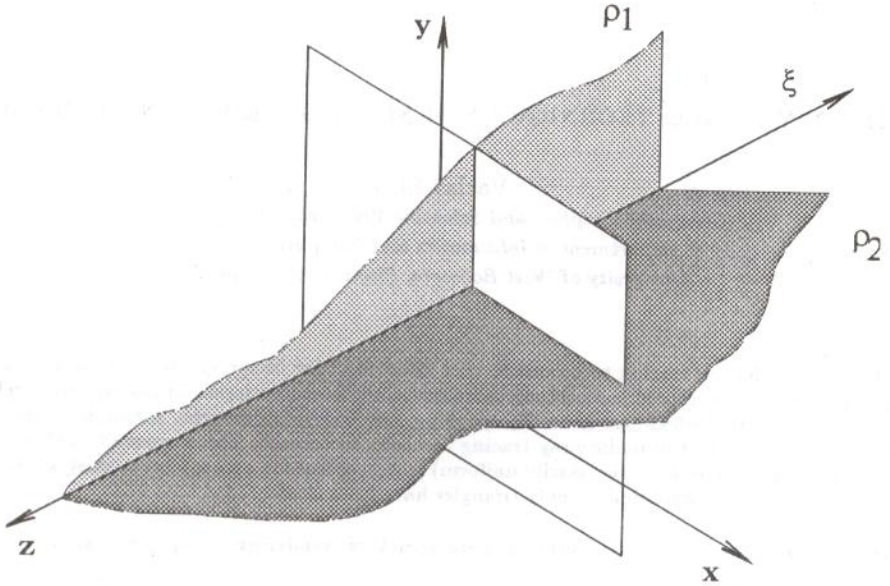

Fig. 1. Perspective primary ray tracing (an observer is in infinity for Parallel Primary Ray Tracing).

The fundamental requirement for ray tracing method is to find a facet, or generally an object and a facet, which is intersected by a given ray and which is the closest facet to the observer, as fast as possible. In general, it means to find all intersection points of the given ray with all facets and select the nearest one. Some bounding volumes like sphere or min-max box volumes are often used to speed up the computation of intersections.

For parallel primary ray tracing the min-max box test

$$x_{min} \leq x \leq x_{max} \quad \text{and} \quad y_{min} \leq y \leq y_{max}$$

or the circle bounding test

$$(x - x_s)^2 + (y - y_s)^2 < r^2$$

will be used for possible intersection detection because those tests seem to be the fastest possible solution.

The proposed BMSSM technique is a method which is based on a principle that any ray ξ can be given as an intersection of two non-collinear planes ρ_1 and ρ_2. In the case of parallel primary ray tracing we can choose planes so that ρ_1 is collinear to $z - x$ plane and ρ_2 is collinear to $y - z$ plane, see fig. 1.

Because the required resolution of the final image (which can be higher than the actual display resolution) is known, it is possible to define a ray at position (i, j) as the

intersection of ${}^{i}\rho_1$, ${}^{j}\rho_2$ planes, where i, j means i-th row and j-th column (it is obvious that the plane equations must be expressed in world coordinates).

Now it is possible to define for each row an ordered p-tuple ${}^{i}R$ of binary values so that for i-th row is

$$ {}^{i}R = [{}^{i}r_1, {}^{i}r_2, \ldots, {}^{i}r_k, \ldots, {}^{i}r_p]^T, \quad 1 \leq k \leq p \quad 1 \leq i \leq n\,, $$

where ${}^{i}r_k$ indicates that the facet(k) is intersected by the plane ${}^{i}\rho_1$, p is a number of facets in the given scene, n is a number of rows.

Then it is possible to define an ordered n-tuple of binary maps $\mathcal{R}$ as

$$ \mathcal{R} = [{}^{1}R, {}^{2}R, \ldots, {}^{n}R]^T\,. $$

Similarly for j-th row

$$ {}^{j}S = [{}^{j}s_1, {}^{j}s_2, \ldots, {}^{j}s_k, \ldots, {}^{j}s_p]^T \quad 1 \leq k \leq p \quad 1 \leq j \leq m\,, $$

where ${}^{j}s_k$ indicates that the facet(k) is intersected by the plane ${}^{j}\rho_2$, m is a number of columns.

Then it is possible to define an ordered m-tuple of binary maps $\mathcal{S}$ as

$$ \mathcal{S} = [{}^{1}S, {}^{2}S, \ldots, {}^{m}S]^T\,. $$

${}^{i}R$ are p-tuples which contain only logical values ${}^{i}r_k$, so that:

${}^{i}r_k = 0$ means that the given plane ${}^{i}\rho_1$ does not intersect a given facet(k),

${}^{i}r_k = 14$ means that the given plane ${}^{i}\rho_1$ intersects a given facet(k).

Similarly for p-tuples ${}^{j}S$.

It can be observed that p-tuples ${}^{i}R$ and ${}^{j}S$ have the same cardinality, e.g. number of columns for all possible primary rays,

$$ \text{card}({}^{i}R) = \text{card}({}^{j}S) = p \text{ for all } i, j\,, $$

while

$$ \text{card}(\mathcal{R}) = n\text{card}(\mathcal{S}) = m\,. $$

It is possible to define generalized bitwise operations + and & with p-tuples as

$$ {}^{i}R + {}^{j}S = [r_1 + s_1, r_2 + s_2, \ldots, r_p + s_p]^T\,, $$

$$ {}^{i}R \& {}^{j}S = [r_1 \& s_1, r_2 \& s_2, \ldots, r_p \& s_p]^T\,, $$

where &, resp.+, means boolean multiplication, resp. addition.

The ray at the position (i,j) is given as an intersection of ${}^{i}\rho_1$ and ${}^{j}\rho_2$ planes. It is possible to define a p-tuple ${}^{ij}Q$ which is defined as

$$ {}^{ij}Q = {}^{i}R \& {}^{j}S = [{}^{ij}q_1, {}^{ij}q_2, \ldots, {}^{ij}q_p]^T\,. $$

It is obvious that:

– if ${}^{ij}qk = 0$ then there is no intersection of the ray at position (i,j) with the facet(k),

– if ${}^{ij}qk = 1$ then the given ray at position (i,j) do intersect *minimal rectangular bounding box* of the facet(k) and it is necessary to use a more detailed test for facet-ray intersection computation.

Some similarities of the BMSSM method can be seen with uniform and adaptive space subdivision acceleration method [1], but used data structures are fairly simple and easy to implement. A_vector of bits can be the simplest way of implementation.

This approach offers one, probably, unusual feature which leads to **better image consistency**. It is well known that small objects may disappear from the final image as an observer moves. The BMSSM method gives at least the basic method how to detect image consistency. The fundamental requirement is that **all** facets must be intersected at least by one ray if we consider each facet alone in the scene.

Let us define p-tuple

$$\mathcal{V} = [v_1, v_2, \ldots, v_p]^T$$

as

$$\mathcal{V} = +_{i=1}^{n}({}^{i}R) = [+_{i=1}^{n}({}^{i}r_1), +_{i=1}^{n}({}^{i}r_2), \ldots, +_{i=1}^{n}({}^{i}r_k), \ldots, +_{i=1}^{n}({}^{i}r_p)]^T .$$

If any k exists so that $v_k = 0$ ($1 \leq k \leq p$), then the facet(k) is not intersected by **any** plane ${}^{i}\rho_1$, for all $i = 1, \ldots, n$.

It means that p-tuple $\mathcal{V}$ must have all items equal to the value 1, e.g.

$$\mathcal{V} + _{i=1}^{n}({}^{i}R) = [1, 1, \ldots, 1]^T .$$

Similarly if $\mathcal{W} = [w_1, w_2, \ldots, w_p]^T$ as

$$\mathcal{W} = +_{i=1}^{m}{}^{j}S = [1, 1, \ldots, 1]^T .$$

If any k exists so that $w_k = 0$ ($1 \leq k \leq p$) then the facet(k) is not intersected by **any** plane ${}^{j}\rho_1$, for all $j = 1, \ldots, m$.

The BMSSM method suppose that all facets are small according to the final image size. Of course if some large objects appear in the scene it is necessary to see that they will be often tested whether they have an intersection point with a given ray. In this case it is recommended to split such an object into small facets, if possible.

It can be easily proved that BMSSM is equivalent to minmax box test as far as the functionality is concerned.

3. Theoretical complexity estimation

There is a small overhead of the BMSSM method because it is necessary to determine the ${}^{i}R$ and ${}^{j}S$ p-tuples and some additional memory is needed to store $\mathbb{R}$ and $\mathbb{S}$ binary maps. It is important to point out that if the image resolution is $n \times m$ pixels and p is a number of facets in the given scene then the complexity of the overhead is given by:
– the complexity of determining ${}^{i}R$ and ${}^{j}S$ is only

$$O(n, m, p) = [n + m]pc_0 ,$$

where c_0 is the cost for determining whether a *plane* do intersect a facet, e.g. using separation test,
– the memory requirements are approximately equal to

$$(n+m)p \quad [bits]\,.$$

Of course, it is not generally possible to avoid the ray tracing complexity for primary rays, which is given by

$$O_1(n,m,p) = nm[c_1 p + c_2 p_1]\,,$$

where: – c_1 is the cost for a *minmax* box or a sphere bounding volume tests, if used,
– c_2 covers the cost of detailed test for ray intersection with a_given facet(k)
– p_1 is a number of minmax boxes intersected by the given ray.

For each ray it is necessary to evaluate **only** boolean expression with bit vectors

$$^{ij}Q = ^{i} R \& ^{j} S\,,$$

which is *faster* than a test which evaluates an intersection with bounding volumes *for all facets.*

The complexity O_2 of primary ray tracing is generally given as

$$O_2(n,m,p) = mn[c_3 p + c_2 p_1]\,,$$

where c_3 covers the cost of $^{ij}q_k = ^{i} r_k \& ^{j} s_k$ computation for a given k and the cost of the test whether $^{ij}q_k \neq 0$ for all $k = 1,\ldots,p$. If we compare complexities O_1(n,m,p) and O_2(n,m,p-) we can see that BMSSM will be faster if and only if

$$O_1(n,m,p) > O_2(n,m,p) \quad \text{i.e.} \quad c_1 > c_3$$

Before making any comparisons it is necessary to point out that time needed for each operation $(:=, <, \pm, *, /)$ does differ from computer to computer, see tab. 1.
So, it is possible for parallel primary ray tracing to estimate time T_{minmax} needed for minmax box volume test (float operations considered only with some probability estimations) if the cost 18 of one **for** cycle statement is considered then

$$T_{minmax} = (0, 4 * 23/12, 0, 0, 0) + 18 = 114\,.$$

The time T_{BMSSM} can be estimated as (see appendix)

$$T_{BMSSM} = 90\,.$$

It is possible to estimate the efficiency ν of the proposed BMSSM which can be expressed as (if worst case considered – ray intersects all facets)

$$\nu = \frac{T_{minmax}}{T_{BMSSM}} = \frac{218}{90} = 1.26\,.$$

If a circle bounding test is used, then time T_{circle} needed for this test (cost 18 of one **for** cycle statement must be considered) can be estimated as

$$T_{circle} = (2, 1, 3, 2, 0) + 18 = 222$$

PC 386DX/387 25 MHz

	:=	<	±	*	/
int	14	42	10	40	58
float	204	260	80	82	154

PC 486 33 MHz 64KB cache – **selected for evaluation**

	:=	<	±	*	/
int	5	9	3	26	44
float	33	50	16	20	114

Time is in 1/10 sec. for 5 000 000 operations

Tab. 1. Time needed for operations $(=, <, \pm, *, /)$.

(2 assignments must be taken into account for storing $(x - x_s)$ and $(y - y_s)$)

$$\nu = \frac{T_{circle}}{T_{BMSSM}} = \frac{222}{90} = 2.54\,,$$

$$\nu = \frac{T_{circle}}{T_{minmax}} = \frac{222}{114} = 1.94\,.$$

Those results mean that **even for parallel primary** ray tracing the BMSSM should be faster than the usage of the minmax box test. Generally not every facet is intersected by a ray the expected ratio ν should be higher because some tests in BMSSM will be skipped.

4. Perspective primary ray tracing

The parallel primary ray tracing is a very special case of perspective ray tracing. Therefore it would be desirable to find a modification of BMSSM for perspective primary ray tracing, see fig.1. In this case it is necessary to use a sphere bounding volume or minmax box tests. If the minmax box test is considered the computational cost is significantly higher for perspective primary ray tracing than the sphere bounding volume, because it is necessary to compute the nearest intersection point of the ray with the minmax box.

In case of sphere bounding volume test, see fig. 2, it can be shown that if we consider a bounding sphere in the form

$$(x - x_q)^T(x - x_q) - r^2 = 0\,,$$

and a ray as

$$x(t) = x_A + s_2 t\,.$$

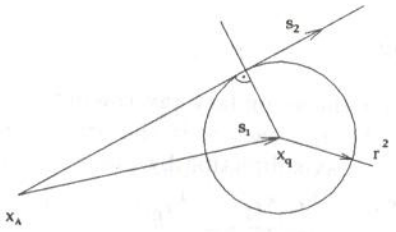

Fig. 2. Sphere bounding volume test.

Then it is possible to write

$$at^2 + bt + c = 0\,,$$

where

$$a = s_2^T s_2 \quad b = -2s_1^T s_2 \quad c = (s_1^T s_1 - r^2)\,,$$

and

$$s_1 = x_q - x_A\,.$$

In this case an intersection of the given ray with a given sphere exists if and only if

$$(s_1^T s_2)^2 - s_2^T s_2 (s_1^T s_1 - r^2) \geq 0\,,$$

where $(s^T{}_1 s_1 - r^2)$ can be precomputed as it does not depend onthe ray because it depends on the sphere bounding volume and observer positions only.

If number of facets or objects is high enough it is convenient to normalize the vector s_2 so that $|s_2| = 1$. The condition can be rewritten as

$$(s_1^T s_2)^2 - q \geq 0\,,$$

where $q = ({}^T_s{}_1 s_1 - r^2)$ is constant for the given facet or object.

If the sphere bounding volume is used then time needed can be estimated as ($w := s_1^T s_2$ (1,0,2,3,0) ; $w * w - q > 0$ (0,1,1,1,0))

$$T_{sphere} = (1, 1, 3, 4, 0) + 18 = 229\,.$$

Then the estimation of the efficiency ν can be expressed as (the cost of the cycle 18 must be considered)

$$\nu = \frac{T_{minmax}}{T_{sphere}} = \frac{218}{229} = 0.95\,.$$

So we have got the theoretical ratio ν that was expected because the minmax box test should be faster for **parallel primary** ray tracing than the sphere bounding volume test for **perspective** case, but the ratio is approximately equal to one.

Now it is possible to compute the expected ratio ν as

$$\nu = \frac{T_{sphere}}{T_{BMSSM}} = \frac{229}{90} = 2.54\,.$$

It means, in the worst expected case, that BMSSM should be **2.54 times faster** than the sphere bounding volume test.

5. Secondary ray tracing

The usage of the BMSSM for the secondary ray tracing is quite simple and straightforward. Let us consider a similar approach as in the primary parallel ray tracing case and divide the given space in the z_axis direction by plane ρ_3, too. In this way binary maps

$$^kT = [^kt_1, ^kt_2, \ldots, ^kt_p]^T \quad 1 \leq k \leq L$$

are obtained with similar properties as binary maps ^iR and ^kS, where L is a number slices in z_direction, see fig. 3.

Similarly we can define $\mathcal{T}$ L-tuple as

$$\mathcal{T} = [T_1, T_2, \ldots, T_L]^T .$$

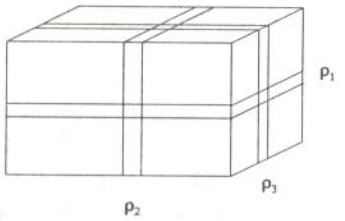

Fig. 3.

The BMSSM method is actually used for detecting whether the given ray can intersect bounding minmax box volume, i.e. supervoxel. Each supervoxel can be imagined as an intersection of three orthogonal slices ^iR, ^jS, ^kT .

The bit map for a supervoxel at the position (i,j,k) can be expressed similarly to the primary ray tracing case as

$$^{ijk}W = {}^iR \,\&\, ^jS \,\&\, ^kT ,$$

where:

$^{ijk}W = [^{ijk}w_1, ^{ijk}w_2, \ldots, ^{ijk}w_p]^T$, and

$^{ijk}w_r = 0$ means that the r-th object does not interfere with the supervoxel (i,j,k),

$^{ijk}w_r = 0$ means that the r-th object does not interfere with the supervoxel (i,j,k).

The proposed BMSSM method is similar to the **Space Subdivision Method (SSM)** that is very often used for substantial ray tracing computation speed up. The SSM method is based on space subdivision into **supervoxels** and **each** supervoxel is associated with information which objects interfere with such a supervoxel, see fig. 4.

This structure is actually an inverted list and can be generally used for interference tests with **non convex** objects, too.

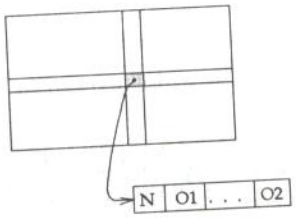

Fig. 4. Space subdivision method – SSM.

Time considerations

In the case of the usage of BMSSM method it is necessary to evaluate expression with bit maps

$$^{ijk}W = {}^{i}R \& {}^{j}S \& {}^{k}T\,,$$

where $^{ijk}W = [^{ijk}w_1, {}^{ijk}w_2, \ldots, {}^{ijk}w_p]^T$. It is not substantial according to the cost of intersection computation of the given ray and selected objects.

In both SSM and BMSSM methods a 3D DDA algorithm can be used for efficient finding of intersection of a supervoxel with the given ray.

Memory considerations

The great disadvantage of the SSM method for large scenes is the memory requirements, that can be approximately expressed as

$$M_{SSM} = 2N^3(q+1+1) \quad [Bytes]\,,$$

where: N^3 is a number of supervoxels for the given scene (N for each direction), q is an average number of objects interfering withthe given supervoxel ($0 < q < p$), if we count 2 Bytes for integer (counter of objects) and pointer (objects identification) implementation.

It is obvious that the amount of the required memory grows extremely fast so the SSM method can be implemented **only for small N**.

Let us consider a scene which is subdivided into $N \times N \times N$ supervoxels. It can be easily shown that the memory requirements for BMSSM can be expressed as

$$M_{BMSSM} = 3Np \quad [bits]\,,$$

or

$$M_{BMSSM} = \frac{3}{8} N\ p\ [Bytes]\,.$$

Then the memory efficiency ν_{mem} can be expressed as

$$\nu_{mem} = \frac{M_{SSM}}{M_{BMSSM}} = \frac{2N^3(q+2)}{\frac{3}{8}N\ p} = \frac{16}{3}\ \frac{q+2}{p}\ N^2\,.$$

There are some special cases that should be mentioned

a) large objects

For large objects can be seen that q will converge to p, so the memory requirements can be expressed ($p \gg 2$) as

$$\nu_{mem} = \frac{M_{SSM}}{M_{BMSSM}} \doteq \frac{16}{3} N^2 ,$$

b) small objects

case when each supervoxel contains one object in average, i.e. $q \doteq 1$; then

$$\nu_{mem} = \frac{M_{SSM}}{M_{BMSSM}} \doteq 16 \frac{N^2}{p} ,$$

c) small objects and extremely sparse

For small objects and extremely sparse $0 \leq q \ll 1$, so

$$\nu_{mem} = \frac{32}{3} \frac{N^2}{p} \doteq 11 \frac{N^2}{p} .$$

All those cases are very special but all show that the proposed BMSSM method **more efficient** according to the SSM method as far as the memory requirements are concerned.

The average number of intersection q can be defined as

$$q = p \frac{r^3}{N^3} ,$$

where r^3 is a average size on an object in supervoxels.

6. Experimental results

Basic experiments have been made on PC 386 for image resolution 256 × 256 pixels.

Parallel primary ray tracing

The following results were obtained for parallel primary ray tracing:

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	10.38	9.52	8.04	6.13	4.74	4.10
630	11.39	10.31	8.55	6.44	4.90	4.22
1000	11.96	10.72	8.83	6.59	4.98	4.28
1600	12.65	11.30	9.22	6.82	5.15	4.44
2500	13.06	11.64	9.48	6.99	5.26	4.55
4000	13.72	12.18	9.86	7.22	5.39	4.51
6300	14.03	12.43	10.03	7.30	5.43	4.65

Tab. 2. Time efficiency ν of the BMSSM against minmax box volume test.

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	26.11	23.40	19.10	13.79	9.86	7.64
630	28.62	25.30	20.29	14.43	10.17	7.82
1000	29.95	26.23	20.88	14.73	10.32	7.92
1600	31.91	27.77	21.87	15.24	10.59	8.11
2500	33.07	28.71	22.53	15.63	10.83	8.29
4000	33.97	29.40	22.96	15.83	10.91	8.35
6300	38.94	33.64	26.24	18.03	12.39	9.48

Tab. 3. Time efficiency ν of the BMSSM against sphere volume test.

From experimental results the time efficiency ν for circle and minmax box tests was $\nu \in < 1.81 , 2.52 >$.

Perspective primary ray tracing results

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	346.57	310.27	253.10	183.01	130.76	101.18
630	7 379.94	335.89	269.40	191.72	135.10	103.83
1000	399.93	350.26	278.72	196.73	137.80	105.68
1600	423.53	368.58	290.28	-	-	-

Tab. 4. Time efficiency ν of the BMSSM against minmax box.

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	33.80	30.48	24.76	17.89	12.83	9.88
630	37.19	32.76	26.32	18.73	13.16	10.15
1000	38.74	33.93	27.01	19.05	13.35	10.24
1600	42.07	36.58	28.86	-	-	-

Tab. 5. Time efficiency ν of the BMSSM against sphere volume.

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	0.72	1.70	3.90	9.44	21.99	50.82
630	1.14	2.68	6.15	14.86	34.69	80.41
1000	1.80	4.25	9.76	23.57	54.82	126.74
1600	2.89	6.81	15.66	37.85	87.94	202.91
2500	4.51	10.65	24.46	59.08	137.34	317.13
4000	7.23	17.06	39.23	94.77	220.22	507.10
6300	11.39	26.89	61.83	149.37	347.27	799.03

Tab. 6. **Average** number of intersection for a ray.

Number of objects	size of objects (one side of the box) 10	16	25	40	63	100
400	6	8	12	21	38	80
630	8	11	17	33	61	121
1000	10	15	23	48	93	192
1600	12	20	32	65	131	304
2500	14	26	45	94	196	436
4000	19	36	67	141	306	689
6300	27	49	98	206	450	1054

Tab. 7. **Maximum** number of intersection for primary rays.

The shown results **do not cover** time for complete detailed tests. Number of the facets used within the tests has been limited due to available memory on PC.

Experiments have shown that it is not convenient for **primary** ray tracing to subdivide space in z-direction.

7. Conclusion

The presented BMSSM method speed up the primary and secondary ray tracing substantially especially if small facets are used in the scene. The advantage of this approach is seemed in a simple data structure which can be used to represent the $\mathbb{R}$ and $\mathbb{S}$, resp. $\mathbb{T}$ tuples. The BMSSM method can be used even for non triangular facets, even for CGS trees and there is a straightforward usage of hierarchical data structures for solids or facets. Tuples $\mathbb{R}$ and $\mathbb{S}$, resp. $\mathbb{T}$ can be pre-computed at the scene definition stage to speed up the ray tracing computation.

The **fundamental advantage** of the proposed method is seen in small memory need with regard to space subdivision technique.

The proposed method gives at least the basic criterion for scene consistency evaluation. From the programmer's point of view, the BMSSM seems to be convenient for application of Object Oriented Programming techniques, too.

Acknowledgments
The author would like to express his thanks to Mr.P.Sebranek for testing BMSSM method and to all who contributed to this work, especially to his students as many suggestions were proposed.

Appendix

The T_{BMSSM} time can be determined from the algorithm shown bellow.

```
/* count is a number of 32 bits words */
count = No objects >> 5;
if ((No objects % 32) ! = 0)count + +;
void COMPUTE (int i, int j, int count)
{ int k;
   long kk;                                          / * podtrzitko je F8 a - */
   unsigned long mask;
   unsigned long huge *mask_x, huge *mask_y;
   mask_y = array_y[i]; /* array stores iS bit maps */
   mask_x = array_x[j]; / * array stores jR bit maps */
   for (k = 0; k < count; k + +)
   { mask = ( *( mask_x + k ) & ( *( mask_y + k ));
    kk = k << 5;
    while ( mask ! = 0L )
    { if (( mask & 1L) ! = 0 ) DETAIL_TEST (kk);
      /* DETAIL_TEST (kk) is a detail test for object kk */
      mask = mask >> 1; kk++;
    }
   }
```

References

1989

[1] Glassner A.S.: An Introduction to Ray Tracing. AP.

[2] Hansmann W., Hopgood F.R.A., Strasser W.: Proc. EUROGRAPHICS'89. North Holland.

1991

[3] Post F.H., Barth W.: Proc. EUROGRAPHICS'91. North Holland.

1992

[4] Speer L.R.: An updated cross indexed guide to the ray tracing literature. ACM SIGGRAPH, 2.

Václav Skala is a Research Professor of Computer Science at the University of West Bohemia in Plzen, previously Institute of Technology. He studied Technical Cybernetics and Computer Science at the Institute of Technology in Plze. In 1975 he took a master degree in Computer Science followed by a PhD degree specializing in Relational Database Systems at the Czech Technical University in Prague. Since 1975 to 1981 he worked as a researcher. In 1978 he studied Computer Science at MEI in Moscow and in the academic year 1983-84 Computer Graphics at Brunel University in London. In 1981 he took up position as a senior lecturer at the Cybernetics Department teaching Programming Languages, Database Systems and Computer Graphics.