11th Conference SIP'2012, pp.121-124, ISBN: 978-1-61804-081-7, St.Malo, WSEAS, France, 2012

An Efficient Space Partitioning Method Using Binary Maps

VACLAV SKALA
Department of Computer Science and Engineering
University of West Bohemia, Faculty of Applied Sciences
Univerzitni 8, CZ 306 14 Plzen
CZECH REPUBLIC
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Abstract: - Space partitioning techniques are well known especially because of their use in computer graphics, especially within ray-tracing acceleration techniques. The primary aim of those techniques is to enable fast test whether a geometric object resides at least partially within the given area. There are many modifications that proved the applicability despite of high memory requirements of complexity $O(M^2)$ for the two dimensional space, resp. $O(M^3)$ for the three dimensional space. The space division technique is used in standard software packages like PovRay etc. A new technique with $O(M)$ memory complexity and its comparison with original space subdivision and residency mask techniques is presented.

Key-Words: - Computer graphics, algorithm acceleration, algorithm complexity, space partitioning, space subdivision, residency mask.

Notifications: p – number of objects, M - number of subdivision in one axis ($M_x = M_x = M_x$ for simplicity)

1 Introduction

The space subdivision is very often used for determination which object from the given data set resides with some part within the given area. Standard techniques of the space subdivision leads to memory requirements of complexity $O(M^2)$ for the two dimensional space, resp. $O(M^3)$ for the three dimensional space, where M is a number of subdivision on one dimension.. In the case of small objects those requirements and not acceptable.

We present a new approach which is of the $O(p\ M)$ memory complexity, where p is a number of objects. Generally, the answer can be given with $O(p)$ complexity. As 64-bit architecture is used nowadays, objects are processed in bulks of 64 objects actually, i.e. additional speed up is obtained.

If tests "which objects interfere with the given area" is used many times then it is convenient to find a technique which could enable to speed up this process. One possibility is to introduce a space subdivision technique and for each k-dimensional interval in the given k-dimensional space keep a list of references of objects that interfere with that interval (area). Several techniques have been developed and used, like uniform, non-uniform, adaptive etc. space subdivisions, usage of trees, quadtrees and octrees etc.[8].

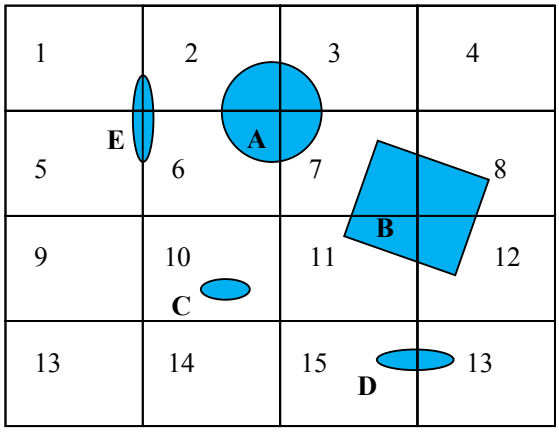

Fig.1 Space partitioning in E^2

2 Space Subdivision Technique

Let us consider space subdivision technique, now. It can be seen that if the object in the scene are "small" enough it is desirable to have the space subdivision as finest as possible, see fig.2.

In the case of the "standard" Space Subdivision (SS) technique, the memory consumption can be approximately estimated as $O(p\ q\ M^k)$, where q is a probability that "an average" object (just one) hits the given interval, generally q is a function defined

as $q = q(M, p, s)$ and s is a size of "an average" object.

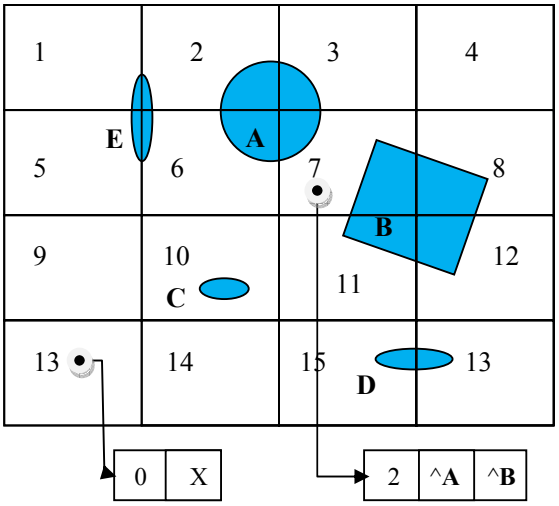

Fig.2

It means that in the case of the three dimensional space, the memory requirements will grow very fast. Let us consider the two dimensional case, i.e. $k = 2$, for more precise estimation. In this case the memory requirements of the SS technique can be estimated as

$$M_{SS} = M^k(4 + 4pq + 4) = 4M^k(pq + 2)$$

where: 4 Bytes are allocated for a pointer representation,
4*q Bytes are allocated for a list of pointers with the probability q
4 Bytes are allocated for the length of the list

3 Residency Mask

The Residency Mask (RM) technique [Cych] uses a bit vector in which each bit is used for the subinterval within the partitioned space for each object in the k-dimensional space, fig.3.

$$\boldsymbol{Q} = \begin{bmatrix} 0000 & 0000 & 0110 & 0110 \\ 0000 & 0000 & 1100 & 0000 \\ 0000 & 0010 & 0000 & 0000 \\ 1100 & 0000 & 0000 & 0000 \\ 0000 & 0000 & 0011 & 0011 \\ 15 & bits & \dots\dots & 0 \end{bmatrix} = \begin{bmatrix} \text{Object } \mathbf{A} \\ \text{Object } \mathbf{B} \\ \text{Object } \mathbf{C} \\ \text{Object } \mathbf{D} \\ \text{Object } \mathbf{E} \\ \end{bmatrix}$$

Fig.3.

Generally an appropriate bit is set to "1" if the object interferes with the given interval in the k-dimensional space. It can be seen that he memory requirements of the Residency Mask are reduced and can be estimated as

$$M_{RM} = pM^k[bits] = pM^k/8\ [Bytes]$$

This technique allows detecting a potential interference of the object using ***land*** operation. A possible interference of objects **C** and **D** represented by row_3 and row_4 in the matrix $\boldsymbol{Q}$ can be expressed as a condition

$$\boldsymbol{Q}[3,*]\ \boldsymbol{land}\ \boldsymbol{Q}[4,*] \neq [0,\dots,0]$$

If the condition is true then that objects **C** and **D** could intersect as their intervals intersect each other. If the value is false then no intersection is possible. It is necessary to point out that finding which object interferes with the given interval is of $O(p\,M^k)$ computational complexity as all objects must be tested using by algorithm 1.

for all j∈{1,…,M^k} **do**
  for all i∈{1,…,p} **do** { for all objects }
    if Q[i,*] **land** MASK[j] **then** DET_TEST

where: MASK[j]=2^j for all j

Algorithm 1

It means that the time complexity can be estimated as $O(pM^k)$ and the processing time is comparable to the uniform space subdivision method. If the number of subdivisions M is higher and objects are small then very long binary vectors are obtained containing almost zeros. In this case it is convenient to represent them as a list of pointers that might be significantly less memory consuming.

4 Binary Masks

Using Binary Mask (BR) technique for determining whether the given object interferes with a subinterval enables to speed up some tests significantly. The RM method is very good for answering the question:

Q1: Find all subintervals which interfere with the given object

But in the most cases we need to answer a little bit different question:

Q2: Find all objects that interfere with the given interval.

Analyzing those tow questions it can be seen that they are complementary, i.e. "inverse" in some sense. We can simply imagine that the RM could be

implemented as a binary matrix, see fig.3. Rows give an answer to the question **Q1**, while columns give an answer to the question **Q2**. It means that for the given j^{th} interval we have to make *p* binary test, see alg.2.

for j^{th} subinterval **do**
  for all $i \in \{1,\ldots,p\}$ **do** { for all objects }
    if **Q**[i,j] ≠ 0 {bit test only} **then** DET_TEST
    # *O(p)* computational complexity only #

Algorithm 2

It means that the processing time is only *O(p)* for determination which object interfere with the j^{th} interval. Nevertheless it is necessary to think how the matrix **Q** could be represented in more efficient way, because for small objects it becomes quite sparse and very large (considering $p \geq 10^3$ and $M \geq 10^2$).

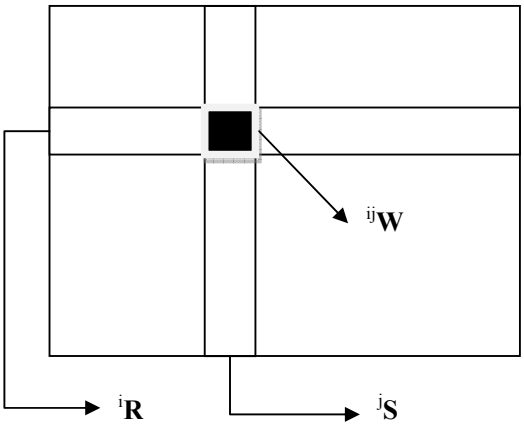

Figure 3.

Let us define $^{i}\mathbf{R}$, resp. $^{j}\mathbf{S}$ sets that contain all objects that interfere with the given i^{th} row slice, resp. j^{th} column slice. Then a set $^{ij}\mathbf{W}$ defined for the two dimensional space as

$$^{ij}\mathbf{W} = {}^{i}\mathbf{R} \cap {}^{j}\mathbf{S}$$

It determines all objects that ***can*** interfere with the subinterval at the *(i , j)* position.

Similarly for the three dimensional case we can write

$$^{ijk}\mathbf{W} = {}^{i}\mathbf{R} \cap {}^{j}\mathbf{S} \cap {}^{k}\mathbf{T}$$

where $^{k}\mathbf{T}$ is a set which contains all objects that interfere with the ^{k}th slice (orthogonal to z-axis).

By now, we haven't dealt with a representation of the set, although the performance of set operations is critical to the BM technique efficiency. Binary vectors can be used as a representation for the sets $^{i}\mathbf{R}$, resp. $^{j}\mathbf{S}$ and $^{k}\mathbf{T}$, assuming that the q^{th} bit expresses whether q^{th} object does interfere with the given slice. In this case we can write for the two dimensional case:

$$^{ij}\mathbf{W} = {}^{i}\mathbf{R} \textbf{ land } {}^{j}\mathbf{S} \qquad \text{for all } i,j \in \{1,\ldots,M\}$$

and for the three dimensional case:

$$^{ijk}\mathbf{W} = {}^{i}\mathbf{R} \textbf{ land } {}^{j}\mathbf{S} \textbf{ land } {}^{k}\mathbf{T}$$

for all $i,j,k \in \{1,\ldots,M\}$.

It can be seen that the memory required for the BM method is given as:

$$M_{BM} = k\,p\,M[bits] = \frac{k\,p\,M}{8}\,[Bytes]$$

But we have to compute an intersection of the given sets. There is no significant difference from the RM and proposed BM methods. Bitwise operations are very fast according to detailed test performed on objects that interfere with the given interval and do not have a significant influence to the computational time.

Another advantage of the proposed technique is that we can easily check the consistency of the given scene. Let us define

$$\boldsymbol{X} = \cup\ {}^{i}\boldsymbol{R} = [x_1, \ldots, x_p]$$

where *p* is a number of objects.
If $\exists i \in \{1, \ldots p\} | x_i = 0$ then the i^{th} object is not "visible" in any subinterval and the preprocessing was not correct. Similarly for other axes:

$$\boldsymbol{Y} = \cup\ {}^{i}\boldsymbol{R} = [x_1, \ldots, x_p] \quad \boldsymbol{Z} = \cup\ {}^{i}\boldsymbol{R} = [x_1, \ldots, x_p]$$

5 Theoretical Comparison

In order to reach some theoretical conclusions it is necessary to compare methods presented above, i.e. Space Subdivision (SS), Residency Masks (RM), Binary Masks (BM) techniques. Let us consider coefficients of efficiency ν_1, ν_2, ν_3 as follows

$$\nu_1 = \frac{M_{SS}}{M_{RM}} \qquad \nu_2 = \frac{M_{SS}}{M_{BM}} \qquad \nu_3 = \frac{M_{RS}}{M_{BM}}$$

Substituting results obtained earlier we get

$$\nu_1 = \frac{4M^k(p\,q + 2)}{M^k\,p/8} = 32(p\,q + 2)$$

As we consider $p \gg 1$ and $p\,q \gg 2$ then

$$\nu_1 = 32\,q$$

For small objects we get in many scenes $q \ll 1$. The RM method is therefore less efficient when objects are getting smaller. Comparing results obtained for the proposed BM method we obtain

$$\nu_2 = \frac{4M^k(p\,q+2)}{k\,M\,p/8} = 32\,M^{k-1}\,(p\,q+2)/p$$

Because $p\,q \gg 2$ then

$$\nu_2 = 32\,q\,M^{k-1}$$

It means that memory efficiency for the three dimensional case can be estimated as

$$\nu_2 = 32\,q\,M^2$$

It is necessary to point out the efficiency ν_2 over the "standard" SS method grows **quadratically**.

Comparing memory requirements for the RM and BM methods we obtain:

$$\nu_3 = \frac{M_{RM}}{M_{BM}} = \frac{M^k q/8}{k\,M\,p/8} = \frac{1}{k}\,M^{k-1}\,\frac{q}{p}$$

and for the three-dimensional case we get

$$\nu_3 = \frac{1}{3}\,M^2\,\frac{q}{p}$$

The results presented above prove significant memory savings if the BM technique is used. The computational time of the intersection detection is small in comparison to the computational time of the detailed test DET_TEST.

6 Conclusion

A new modification for the space subdivision technique, the BM method, was developed with *O(k N)* memory complexity instead of $O(q\,M^k)$. The memory complexity of the proposed method grows with the dimensionality linearly, while the "standard" SS technique has $O(M^k)$ complexity, in general. The efficiency of the proposed method grows with $O(M^{k-1}/k)$.

The experiments proved that the MB technique provides faster solution as well. Nevertheless the proposed method is not convenient for some cases especially in the volume visualization methods, e.g. for CT or MRI data processing.

Conclusion

The author thanks to colleagues and students at the University of West Bohemia, Plzen and VSB Technical University, Ostrava for their comments and suggestions and to anonymous reviewers for constructive comments. The project was supported by the MSMT CR, projects No.LA10035 and ME10060.

References:

[1] Aeraal,L., Programming Principles in Computer Graphics, 2-nd ed. John Wiley&Sons, 1992.

[2] Arnaldi,B., Priol,T., Bouatouch,K., A new space subdivision method for ray tracing CSG modeled scenes, The Visual Computer, Vol.3, pp.98-108, 1987.

[3] Avo,A., Graphics Gems II, Academic Press, 1991.

[4] Bauman,G., Livny,Y., El-Sanna,J., GPU-Based Adaptive Subdivision for View Depended Rendering, WSCG 2009, Union Agency, Science Press, 2009.

[5] Cleary,J.G., Wivill,G., Analysis of an Algorithm for fast ray tracing using uniform space subdivision, The Visual Computer, Vol4., pp.65-83, 1988.

[6] Costa,V., Pereire,J.M., Compact Rectilinear Grids for the Ray Tracing of Irregular Scenes, WSCG 2011 Proceedings, pp.9-16, Union Agency, Science Press, Czech Republic, 2011.

[7] Cychosz,J.M., Use of Residency Mask and Object Space Partitioning to Eliminate Ray Object Intersection Calculation, Graphics Gems III, pp.284-287, 1992.

[8] Glassner,A.(Ed.), Graphics Gems, Academic Press, 1990.

[9] Hapala,M., Karlik,O., Havran,V., When it Makes Sense to Use Uniform Grid for Ray Tracing, WSCG 2011 Communication papers proceedings, pp.193-200, 2011.

[10] James,A., Day,A.M., WEB Clustering: A New Approach to Space Partitioning, WSCG99 Proceedings, Plzen, pp.165-170, 1999.

[11] Kirk,D.(Ed.), Graphics Gems III, Academic Press, 1992.

[12] MacDonald,D.J., Booth,K.S., Heuristics for Ray Tracing Using Space Subdivision. *Visual Computer*, 6(3):153-166, 1990.

[13] Skala,V., Ray Tracing Methods and Its Complexity (in Czech), Algoritmy 93 Conference Proceedings, Slovakia,1993.

[14] Skala,V., Memory Saving Technique for S pace Subdivision Technique, Machine Graphics and Vision, Poland Academy of Sciences, Vol.2., No.3., pp.237-250, 1993.