Journal of Computer Science 6 (6): 625-633, 2010
ISSN 1549-3636

An Alternative Ray Tracing Algorithm for Understanding Scenes with Embedded Objects

[1]Nancy Dandachy, [1]Dimitri Plemenos, [2]Safwan Chendeb and [3]Bachar El Hassan
[1]XLIM Laboratory UMR CNRS 6172, Department of Computer Science,
University of Limoges, 83 Rue d'Isle, 87000 Limoges, France
[2]PARAGRAPHE/CITU Laboratory, Department of Computer Science,
University of Paris 8, 93526 Saint Denis, France
[3]Departement of Electrical Engineering, Faculty of Engineering, Lebanese University,
Rue AL Arz, Al Kobbeh, Tripoli, Liban

Abstract: Problem statement: In the virtual world domain, the existing techniques of exploration, were not able alone to resolve the understanding problem of scenes with embedded object, also they are time and memory consuming. As a solution, a novel method that enhances the visualization of 3D ray traced scenes with embedded objects by creating a hole proportional to its interior parts was developed and tested. **Approach:** This research presented a novel approach that allows an automatic exploration of 3D scenes with embedded objects. First of all, the apparent contour of the interior object that we want to see, were computed by using the ray tracing algorithm with the selective refinement optimization approach. The second step was to search for pixels which are orthogonal in a certain depth and directed toward the outside of the silhouette in order to create a hole. **Results:** The obtained results were convincing and answering to the goal of this research. The proposed method allows the creation of a hole around an object and can be applied to any type of model. **Conclusion:** This very successful approach for 3D scenes with embedded object exploration is further supported by its ability to give at the same time a global idea about the scene as well as a possibility to explore its interior while saving time and memory.

Key words: Computer graphics, 3D visualization, selective refinement and ray tracing, contour detection

INTRODUCTION

The fast development of the image synthesis domain, the spread of this domain in lot of applications and then because of the development of PC's performance in speed and capacities, the problem of scene understanding and extracting knowledge is becoming more and more pertinent and complicated. The first work in the field of the comprehension of virtual world was published at the end of the eighties and at the beginning of the nineties. There was very little work that faces this problem, because the community of the graphic data processing was not convinced that this field is important for the computer graphics. Only during these last year's people have begun to understand its importance and the necessity to have fast and accurate techniques for good exploration and clear understanding of various virtual worlds.

However, these techniques are not able alone to answer all problems of 3D complex scenes used in various fields and applications. Each scene has its own case that has to be treated separately. For example in the virtual world domain, the case of scenes with embedded objects that we want to have at the same time a global idea about the scene as well as an idea about a part of its interior in order to add some information (Fig. 1).

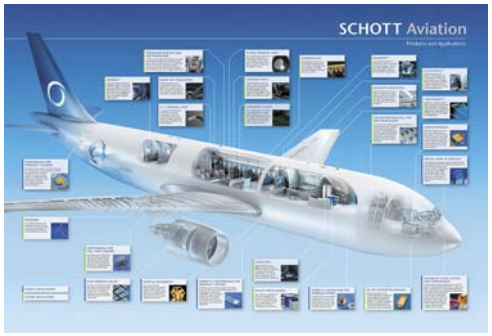

Fig. 1: Scene plane with AR

Corresponding Author: Nancy Dandachy, XLIM Laboratory UMR CNRS 6172, Department of Computer Science, University of Limoges, 83 Rue d'Isle, 87000 Limoges, France

The existing techniques of exploration which based on the computation of a good point of view and/or doing a local or global animation, cost lot of time and memory and do not allow us to have a such result. Even visualization techniques, which are based on the traditional or realistic methods, are not able to resolve this problem. For this reason, we are going to present a new technique that allow the understanding of scenes with embedded objects by creating a hole which is proportional to the interior objects in order to make visible. It is based as a first step on the computation in image space of the apparent contour of the interior objects that we want to see by using a hybrid method that use the selective refinement partition of the ray tracing algorithm combined with the code direction technique. The second step consists on the computation of pixels which are toward the apparent contour in order to create the hole around the interior objects.

Background at the beginning: One of the most widespread problems was the choice of a bad position of the point of view which misses important details or necessary in formations for the comprehension of the three-dimensional scenes (Fig. 2). Several techniques (Colin, 1988; Barral *et al*., 1999; Dorme, 2001) were implemented within this framework and research continues until now in order to give better results that can save costs in time and memory (Vasquez *et al*., 2002; Vasquez and Sbert, 2003; Sokolov *et al*., 2006; Shinya *et al*., 1987).

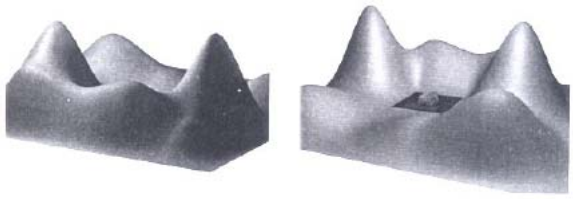

Fig. 2: Left: Bad viewpoint position. Right: Good view point position

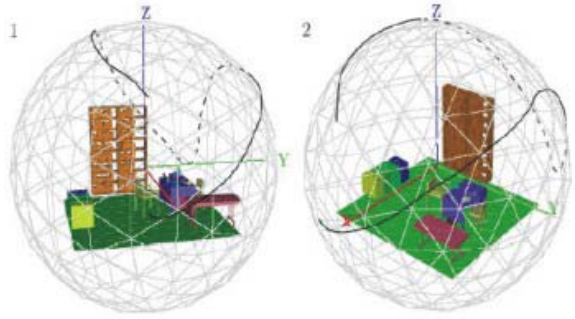

Fig. 3: Global exploration

When the calculation from several good points of view was not sufficient because it does not guarantee the passage of a point to another without making abrupt changes, several authors have proposed methods for global and local exploration through an animation which cross good points of view according to a way which follows heuristic rules avoiding the abrupt changes of the camera (Fig. 3) (Barral *et al*., 1999; 2000; Dorme, 2001; Jaubert *et al*., 2006; Sokolov and Plemenos, 2005; Shinya *et al*., 1987; Vasquez and Sbert, 2003).

However, it is not sufficient to move around the scene. Therefore, we thought about changing the mode of visualization. The existing techniques of visualization are the traditional and the realistic visualization. The first one is divided on two modes:

- The wireframe mode where only contours of objects or facets are posted in the scene (Fig. 4)
- The full mode with an elimination of hidden parts where all objects or facets, considered to be closest to the position of a given point of view, are visualized (Fig. 5)

The problem of the elimination of hidden parts was one of the most important and difficult problem on computer graphics. The first algorithm was developed in the sixties. Its goal was to determine lines, surfaces or volumes which are visible to the observer located in a given point on the 3D space.

Fig. 4: Scene bunny on full mode with elimination of hidden parts

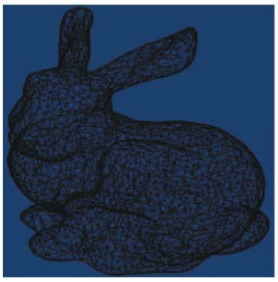

Fig. 5: Scene bunny on wireframe mode

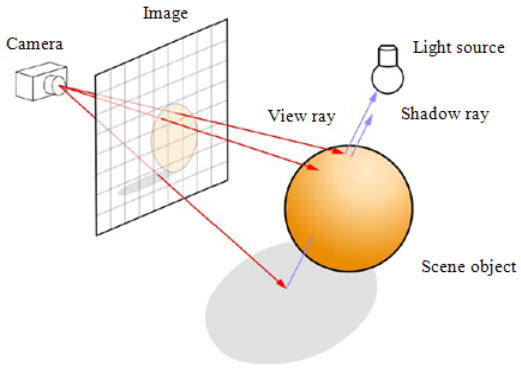

Fig. 6: The ray tracing

The realistic visualization uses the full mode with an elimination of hidden parts that express the photorealism by adding light effects mirrors, shades transparency, textures, reflexions and refractions in order to be the nearest possible to the reality. It is based on the ray tracing algorithm and its varieties: the photon mapping and the radiosity. The photon mapping is founded on the raytracing algorithm and simulates the refraction of light through a transparent substance such as glass or water, diffuse inter-reflexion between illuminated objects, the scattering subsurface of light in translucent materials and some of effects caused by particulate matter such as smoke or water vapor. The radiosity is an application of the finite element method to solving the rendering equation for scenes with purely diffuse surfaces. We will not approach in this study the principles of the radiosity and the photon mapping visualization because they do not enter in the objectives of this study.

As we are going to use the ray tracing algorithm with the selective refinement optimization it is important to talk briefly about the related work in this field. The ray tracing algorithm so popular today, date of 1968 and it was initially suggested by Appel (1968). Its first implementation goes back to 1971, in the software of three-dimensional visualization MAGI. It makes it possible to calculate the visibility of the objects at the same time as their illumination. It is able to manage the shades, the transparencies, the plating of textures and the interactions between the objects. Moreover, it is adapted to any type of graphic primitives. This improvement has a cost of course: The computing times are much more important than for the algorithms seen previously.

The principle of the algorithm is the following (Fig. 6):

- One considers a beam of imaginary rays connecting the eye of the observer to the center of each elementary square (pixel) of the space image
- For each ray, its intersections with all surfaces of the scene are calculated, in order to determine the nearest intersection point to the observer
- he luminous intensity of this point of intersection is assigned to the corresponding pixel

Once the visible point by the observer is given, it is necessary to deal with the problem of shadows by sending a ray from the visible point to the source of light. If this ray cuts a surface before the visible point, the visible point will be then considered in shade.

Kay and Greenberg (1979) proposed an extension of the ray tracing algorithm allowing the taking into account of the refraction of the ray when it crosses transparent surfaces.

Whitted (1980) proposed the "backward ray tracing" algorithm which simulate the opposite course of the light towards the eye. It is based on the decomposition of the luminous intensity of a point in a specular component of reflexion and transmission. This decomposition gives a binary tree that the algorithm must traverse each time a ray is sent.

Kay and Kajiya (1986); Muller (1986) and Rubin and Whitted 1980) tried to improve the ray tracing algorithm created by Whitted (1980) in order to reduce the number of sending rays and the computing time while preserving an acceptable quality of the images obtained. Roth (1982) used bounding boxes for the Constructive Solid Geometry (CSG) trees.

Kaplan (1985) uses the Binary Space Partition (BSP) for a recursive subdivision of the image space into 3D voxels. There is also the beam tracing algorithm which classifies propagation paths from a source (Arvo and Kirk, 1987; Muller, 1986; Shinya *et al*., 1987) by tracing recursively pyramidal (Ghazanfarpour and Hasenfratz, 1998) or conical (Amanatides, 1984) beams of rays (set of rays).

Works in (Cook *et al*., 1984; Green and Paddon, 1989; Keates and Hubbold, 1995) use parallel machines in order to save time and which is based on a parallelization using a processor by pixel or group of pixels, by voxel or by object. A relatively complete table which summarizes the accelerating techniques of ray tracing appears in (Arvo and Kirk, 1987; Amanatides, 1984; Ghazanfarpour and Hasenfratz, 1998; Glassner, 1984; Kay and Kajiya, 1986; Shinya *et al*., 1987). Recently, many accelerating algorithm (Cassagnabere *et al*., 2006) which based on the GPUs algorithm in order to optimize the ray triangle intersection calculation.

In addition to these techniques, there exist two rather important techniques where the goal is to reduce the computing time: Selective refinement (Dandachy,

2007) and the selective expansion (Plemenos and Sellinger, 1998). We will study in detail the algorithm of the selective refinement which will be used in our technique.

MATERIALS AND METHODS

The selective refinement algorithm: Selective refinement is one of the most elegant approaches which tend to decrease the computing time by limiting the number of sending rays. It was initially used by Warnock (1969) in its hidden parts elimination algorithm by recursive subdivisions of the image space, then by Catmull (1974) using recursive subdivisions of surfaces until obtaining the simple situations where the problem of the visibility could be solved easily. Jansen and Van Wijk (1984); Plemenos and Sellinger (1998); Dandachy (2007) and Dandachy *et al*. (2007) proposed similar methods applied to the ray tracing algorithm for the elimination of hidden parts.

The algorithm has 2 version where the principle idea is to divide the image space into a set of macro pixels. Each one is a $2^n \times 2^n$ pixels.

1st version:

- To each macro pixel, a ray is sent from the observer to its high left HL pixel (Fig. 7)
- If the visible surface obtained is different from those seen by the HL pixels of its neighbor macros pixels, the current macro pixel is subdivided in four sub macros pixels and the process starts again for each one of them
- If not, there is a great probability that all the pixels of the current macro pixel see the same visible surface. An approximate luminous intensity will be then attributed. It can be obtained by making a linear interpolation between the luminous intensities of the HL pixels of the current macro pixel and those of its neighbor ones
- The process of subdivision stops as soon as a threshold of subdivision, defined by the user, is reached

2nd version:

- To each macro pixel, a set of rays is sent by the observer to a set pixels called by its "guide-pixels" whose number and position are fixed and chosen before starting by the user. To obtain convincing results, the number of guide-pixels must be at least equal to 3 (Fig. 8)
- If visible surfaces obtained from these "guide-pixels" are different, the running macro pixel is subdivided in four sub macros pixels and the process start again for each one of them

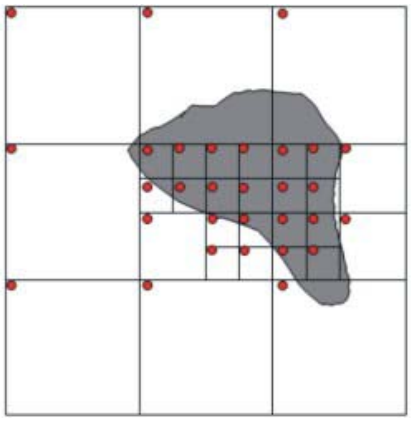

Fig. 7: The 1st version of the selective refinement algorithm

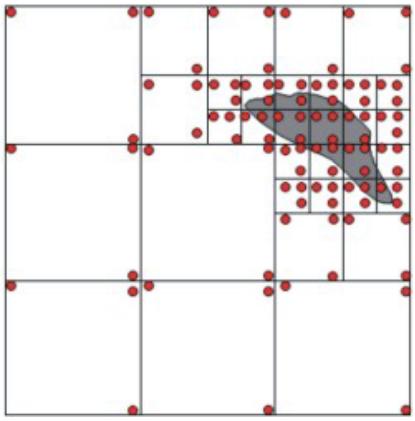

Fig. 8: The 2nd version of the selective refinement algorithm

- If not, it means that the same face is visible by all the pixels of the running macro pixel. An approximate luminous intensity will be then calculated
- The process of subdivision stops as soon as a threshold of subdivision, defined by the user, is reached

In addition to selective refinement, there exist also the techniques of the selective expansion applied to the ray tracing algorithm. Its goal is to reduce the number of sending rays in order to avoid the rays which do not cut the scene which lead to reduce the computing time of the scene visible parts. However this technique depends on a preliminary sampling, intended to determine the basic list of useful macros pixels. Thus, an insufficient sampling might remove certain small objects.

Scene exploration by creating a hole around the interior objects: Our approach is divided in two parts:

- The apparent contour detection part of the interior objects
- The computation of orthogonal pixels directed toward the outside of the apparent contour in order to create the hole

Apparent contour detection of the interior objects: This part is divided on two steps:

- The step 1 uses the optimized ray tracing technique based on the selective refinement algorithm in order to search, for each interior object that we want to make visible, for an initial contour pixel (Fig. 9)
- The step 2 uses the initial contour pixels obtained from step 1 and uses the code direction technique (Dandachy *et al*., 2007) in order to search for the complete contour pixels (Fig. 10)

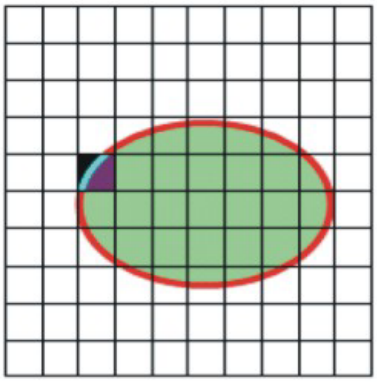

Fig. 9: Search for each interior object for an initial contour pixel

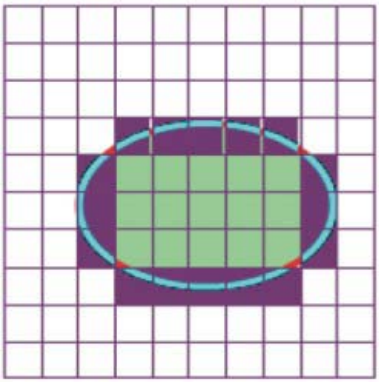

Fig. 10: Apparent contour pixels

In order to get a departure contour point, we divide the image space into a set of macro pixels (each macro pixel is about 8×8 pixels). For each macro pixel, we send rays to the Up Right (UR), Up Left (UL) Down Right (DR) and Down Left (DL) pixels to detect for each ray the Id of the closest object (Fig. 11). We associate each returned Id to its correspondent pixel. The macro pixels which represent different intersections most contain a contour pixel. They are considered as our useful macro pixels which are subdivided into 4 sub macro pixels.

The same process is applied to each sub macro pixel until we obtain a block of 2×2 pixels. The block of 2×2 pixels that has intersection with different objects, contains certainly at least a one contour point. More we have different intersections in the block, more we have initial contour pixels. To avoid having more than one initial contour pixel for the same object, since we get the first contour pixel of an object, we neglect all other pixels that have the same ID.

Step 2: Searching for the complete contour: Before talking about this step, we define first, for each pixel, its 8 neighbors. Each pixel in the neighborhood has a previous and a following pixel, respecting the order indexed from 0-7 (Fig. 12).

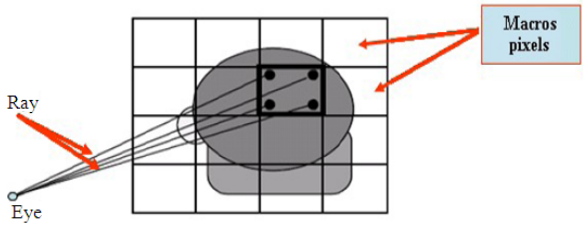

Fig. 11: Send rays to the UR, UL, DR and DL pixels of a micro pixel

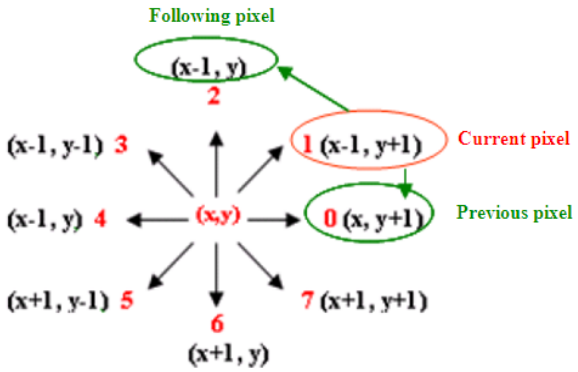

Fig. 12: Defining a pixel neighbors

It is obvious that all pixels defining the contour of the object should have the same Id of the departure one. As the contour point should be the separated point between tow different zone, its following neighbor pixel and its previous one should have different Ids. Otherwise, the current pixel should necessary be an interior point of the object.

Since we get from step 1, for each interior object that we want to make visible, a departure contour point, we uses the code direction algorithm which starts with an initial contour pixel and follows, at each time, a certain direction that conducts us to the following contour point until we obtain the complete contour of an object. At each time, the direction should be one of the 8 directions that conduct to the 8 neighbors defined on Fig. 12.

In order to choose the departure direction that conduct us from the departure contour point to the second contour point, we send a ray to each neighbor of the departure point. The first one that has the same Id of the departure point and its previous and following pixels have different Ids will be our second contour pixel. If all the neighbors were tested and none of them was a contour pixel, we stop the research.

Since we get the second contour pixel, we apply the same process to find the following direction by considering the obtained contour pixel as a starting one. In order to avoid a return to a chosen contour point, we only test between the 8 neighbors which are not tested yet. To do so, we associate to each pixel a buffer that we call the v-buffer that is updated each time the pixel is tested. The algorithm will stop when we fall in one of these tow cases:

- We return to the initial departure point (closed contour)
- None of the neighbors of the current pixel is a contour point (opened contour) Fig. 13

Fig. 13: The apparent contour of bunny scene

Orthogonal pixels detection toward the apparent contour: While getting by the code direction technique, pixels that form the apparent contour of the interior objects, the algorithm search for pixels toward the silhouette using tangent and normal directions (Fig. 14).

The tangent vector direction in a current pixel is given by following formula:

Tg_Vector = following_contour_pixel - current_pixel

Each tangent direction has 2 possible normal directions selected among the 8 directions in the neighborhood of the current pixel: The outgoing and the ingoing one. We are interested on the outgoing direction that points to the pixel having a different Id from the current one. The ingoing direction should points to the pixel having the same Id of the current pixel.

By computing the outgoing normal directions with the depth of one pixel we obtain the following result in Fig. 15. It appears to be not efficient in order to suggest the idea of a real hole since the depth of one pixel is not sufficient and the hole is not continuous.

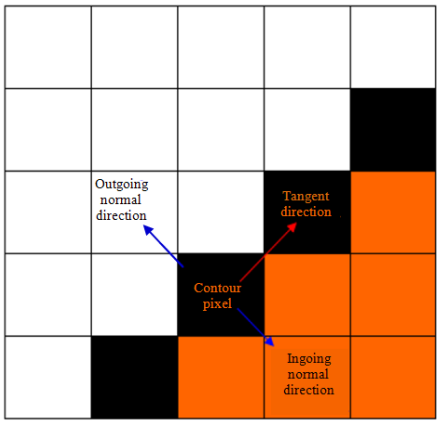

Fig. 14: tangent and normal directions

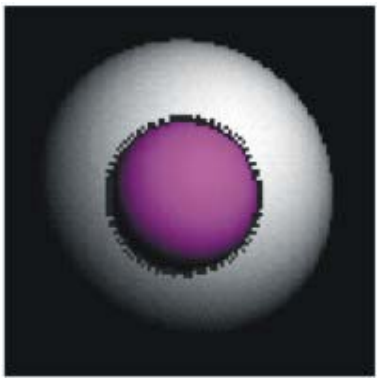

Fig. 15:The hole with a one depth outgoing normal directions

In order to regulate this, we suggest to augment the depth number and to take into account not only outgoing normal directions, but also its the previous and following directions which are illustrated in the Fig. 16.

In order to define a depth, we search also for pixels which are in the k^{th} depth of the outgoing normal direction, its previous and following directions (see Fig. 17). These points are N_k, P_k and F_k which are obtained by using these following formulas:

$$N_k = C + k \times N_0$$
$$P_k = C + k \times P_0$$
$$F_k = C + k \times F_0$$

Where:
C = The current pixel
N_k = The k^{th} pixel in the k^{th} outgoing normal direction
P^k = The k^{th} pixel in the k^{th} previous direction
F_k = The k^{th} pixel in the k^{th} following direction

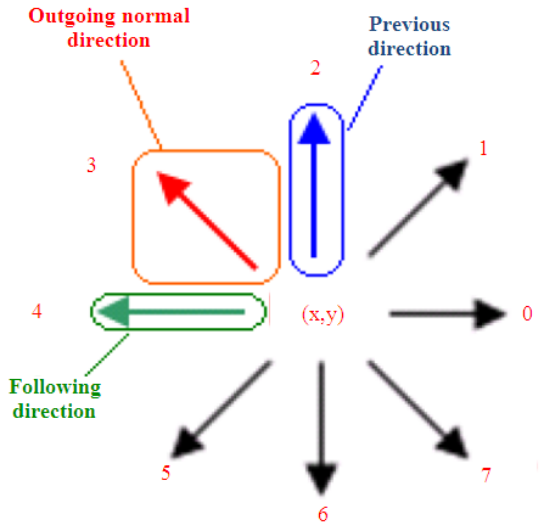

Fig. 16: The previous and following direction to a outgoing normal direction

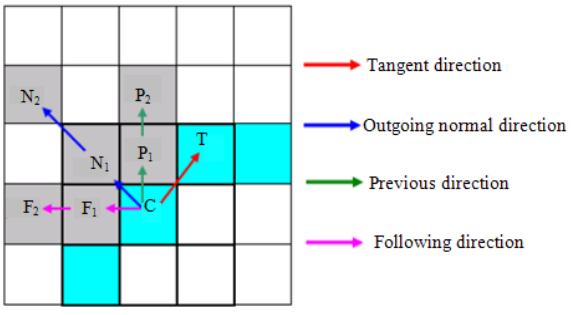

Fig. 17: The K^{th} pixels toward the current pixel C

RESULTS AND DISCUSSION

Generally, a depth of 4 pixels is enough to give good results (Fig. 18 and19).

More we get pixels; more the hole gives the impression to be deeper. However, it is important to be aware to the fact that the hole should not be so large compared to the outside object volume since it might mask a big part of it.

In order to regulate this problem, we take a depth that do not exceed the half distance d where d is the minimal distance from the all k^{th} pixels of the hole to the projected boundaries edges and it is given by the following formula:

$$d = \min_{p \in P}(dist(p,S))$$

Where:
P = The set of pixels in the depth k of the hole
S = The set of the boundaries edges

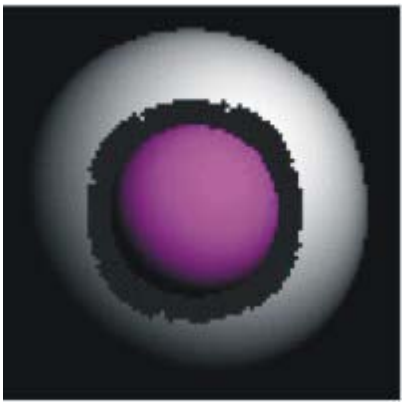

Fig. 18: Scene sphere

Fig. 19: Scene house

Fig. 20: The example of tow internally tangent sphere

We are not interested in cases where the interior object is very closed to the border of the including object, because it will be useless to create a hole which is not able to have a sufficient depth without exceeding the volume of the including object. Figure 20 shows the example of 2 internally tangent sphere.

In order to avoid the obtaining image presented in Fig. 20, it might be possible before starting our algorithm, to use one of techniques based on the calculation of good points of view described in (Sokolov *et al*., 2006).

CONCLUSION

We have presented in this study, a technique which help to improve the visualization of three-dimensional scenes with embedded objects. These methods are easy to implement, rapid and are not expensive in memory. The obtained results are convincing and answer to our goal.

The creation of a hole around the interior objects is an effective idea since it guarantees to the observer to be able to clearly see the including objects as well as some of its interior parts. The proposed method allow the creation of a hole around an object and can be applied to any type of models since it is based on the detection of the interior object's contour pixels and then on the research of the orthogonal pixels toward the interior object's silhouette.

Practically, it does not present any problem except the case where an interior object is very close to the border of an including object which make the algorithm unable to create a hole around the object including without exceeding its border.

In order to have a better result, it might be better in the future to begin with a good point of view calculated by one to the techniques presented in the Introduction of this study.

We have admitted in this study that we know the interior parts that we want to make visible. It would be interesting in the future to develop a new automatic method of a scene exploration technique allowing interaction with the user, where the user can point out which parts of the scene he would like to explore in detail before starting the algorithm of creating holes.

REFERENCES

Amanatides J., 1984. Ray tracing with cones. Comput. Graph., 18: 129-135. DOI: 10.1145/964965.808589

Appel, A., 1968. Some techniques for shading machine renderings of solids. Proceeding of the AFIPS Spring Joint Computer Conference, Apr. 30-May 2, Atlantic City, New Jersey, pp: 37-35.

Arvo, J. and D. Kirk, 1987. Fast ray tracing by ray classification. Proceeding of the International Conference on Computer Graphics and Interactive Techniques, (CGIT'87), ACM Press, New York, USA., pp: 55-64.

Barral, P., G. Dorme and D. Plemenos, 1999. Visual understanding of a scene by automatic movement of a camera. Proceeding of the International Conference GraphiCon'99, Aug. 26-Sept. 3, Moscow (Russia), pp: 59-65.

Cassagnabere, C., F. Rousselle and C. Renaud, 2006. CPU-GPU multithreaded programming model: Application to the path tracing with next event estimation algorithm. Proceeding of the International Symposium on Visual Computer, Nov. 2006, Lake Tahoe, NV, USA., pp: 265-275.

Colin, C., 1988. A system for exploring the universe of polyhedral shapes. Proceeding of the Eurographics'88, Sept. 1988, Nice (France), pp: 209-220.

Cook, R.L., T. Porter and L. Carpenter, 1984. Distributed ray tracing. SIGGRAPH Comput. Graph., 18: 137-145.

Dandachy, N., 2007. Alternative visualization techniques for understanding three dimensional scenes. Ph.D. Thesis, University of Limoges, France.

Dandachy, N., Plemenos D. and B. El Hassan, 2007. Scene understanding by apparent contour extraction. Proceeding of the International Conference of 3IA'07, May 2007, Athens (Greece), pp: 85-96.

Dorme, G., 2001. Study and realization of techniques for understanding three dimensional scenes. Ph.D. Thesis, University of Limoges, France.

Catmull, E., 1974. A subdivision algorithm for computer display of curved surfaces. Ph.D. Thesis, University of Utah.

Ghazanfarpour, D. and M. Hasenfratz, 1998. A beam tracing with precise antialiasing for polyhedral scenes. Comput. Graph. J., 22: 103-115. DOI: 10.1016/S0097-8493(97)00086-1

Glassner, A.S., 1984. Space Subdivision for fast ray tracing. IEEE Comput. Graph. Appli., 4: 15-22.

Green S.A. and D.J. Paddon, 1989. Exploiting coherence for multiprocessor ray tracing. IEEE Comput. Graph. Appli., 9: 12-26.

Jansen, F.W. and J.J. Van Wijk, 1984. Previewing techniques in raster graphics. Comput. Graph., 8: 149-161.

Jaubert, B., D. Plemenos and K. Tamine, 2006. Techniques for off-line scene exploration using a virtual camera. Proceeding of the International Conference of 3IA'2006, May 23-24, Limoges (France), pp: 31-44.

Kaplan, M.R., 1985. Space tracing, a constant time ray tracer. SIGGRAPH Course Notes, 18: 149-158.

Kay T.L. and J. Kajiya, 1986. Ray tracing complex scenes. Comput. Graph., 20: 269-278.

Kay, D.S. and D. Greenberg, 1979. Transparency for computer synthesized images. ACM SIGGRAPH Comput. Graph., 13: 158-164.

Keates, M. and R.J. Hubbold, 1995. Interactive ray tracing on a virtual shared-memory. Comput. Graph. Forum, 14: 189-202. DOI: 10.1111/1467-8659.1440189

Muller, H., 1986. Image generation by space sweep. Comput. Graph. Forum, 5: 189-196. DOI: 10.1111/j.1467-8659.1986.tb00297.x

Plemenos, D. and D. Sellinger, 1998. Declarative modeling by iterative refinement. Proceeding of the International Conference of 3IA'98, Apr. 1998, Limoges (France), pp: 67-79.

Roth, S.D., 1982. Ray casting for modeling solids. J. Comput. Graph. Image Process., 18: 109-144. DOI: 10.1016/0146-664X(82)90169-1

Rubin, S. and T. Whitted, 1980. A three dimensional representation for fast rendering of complex scenes. Proceeding of the International Conference on Computer Graphics and Interactive Techniques, July 14-18, AM Press, Seattle, Washington, United States, pp: 110-116. http://portal.acm.org/citation.cfm?id=800250.807479

Shinya M., T. Takahashi and S. Naito, 1987. Principles and applications of pencil tracing. ACM SIGGRAPH, 21: 45-54.

Sokolov, D., D. Plemenos and K. Tamine, 2006. Methods and data structures for virtual world exploration. Visual Comput., 22: 506-516 .

Sokolov, D. and D. Plemenos, 2005. Viewpoint quality and global scene understanding. Proceeding of the 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage, Nov. 2005, The Eurographics Association, Pisa (Italy), pp: 67-73. http://www.gametools.org/archives/publications/LIM_vast2005.pdf

Vasquez, P.P and Sbert M., 2003. Automatic indoor scene exploration. Proceeding of the International Conference of 3IA'03, May 2003, Limoges (France), pp: 13-24.

Vasquez, P.P., M. Sbert, M. Feixas and W. Heidrich, 2002. Image-based modeling using viewpoint entropy. Proceeding of the Computer Graphics International, July 2002, pp: 267-279.

Warnock, C.S., 1969. A hidden surface algorithm for computer generated half-tone pictures. Department of Computer Science, University of Utah.

Whitted, T., 1980. An improved illumination model for shaded display. Commun. ACM, 23: 343-349.