Visual Understanding of a Scene by Automatic Movement of a Camera

Pierre BARRAL, Guillaume DORME, Dimitri PLEMENOS
(in alphabetic order)
barral@novinfo1.unilim.fr, dorme@alphainfo.unilim.fr, plemenos@unilim.fr
MSI laboratory, University of Limoges, France.

Abstract

A method for visual understanding of a scene by efficient automatic movement of a camera is presented in this paper. The purpose of this method is to choose a trajectory for a virtual camera, allowing the user to have a good knowledge of the scene at the end of a minimal flying over. The method is based on techniques for computing a good point of view, used together with various heuristics, in order to have smooth movement of the camera and to avoid fast returns to the starting point, due to local maxima. So, starting from a good view point, the virtual camera moves on the surface of a sphere surrounding the scene, combining good view, smooth camera movement and distance from the starting point based heuristics.

Keywords: *Scene understanding, Good view direction, Virtual camera, Heuristic search.*

1. INTRODUCTION

Modelling is a very important step in computer graphics. During the interactive design of a scene, the user would like to control results of his (her) work. In order to understand a scene it is important to choose a view direction which shows its most important features. Such a view direction is very difficult to find interactively because the scene is generally 3-dimensional while the screen is 2-dimensional. Thus, it is very important that a scene modeller offers an automated computing of a good view direction. Indeed, the modeller has much more information about the scene than the user and could use this information to automaticaly compute a good view direction. This is especially true for declarative modeling [3, 5], where the designer has insufficient knowledge of the scene during the designing process. However, very few works exist in this domain.

It must be clear that the notion of *good view direction* is a subjective concept. So an algorithm for computing such directions do not give a general solution but a solution good enough in many cases.

Unfortunately, the use of a single view direction is often insufficient to well understand a scene, especially in the case of a complex scene. Thus, in order to give the user sufficient knowledge of a scene, it should be better to compute more than one view directions, revealing the main properties of this scene. The problem is that, even if the solution of computing more than one view directions is satisfactory from a theoretical point of view, from the user's point of view it is not a solution at all because blunt changes of view direction are rather confusing for him (her) and the expected result (better knowledge of the scene) is not reached.

After a brief presentation of the main work on methods for automatically computing a good view in section 2, we will explain, in section 3, why the knowledge of one or more view directions is not always enough to understand well a scene and we will give the general idea of our approach. A general presentation of the proposed method, based on smooth movement of a virtual camera, will be done in section 4. In section 5 a hardware based technique for fast computing good view directions will be proposed. Heuristics used to guide the camera's movement will be presented in section 6. In section 7 advantages and drawbacks of the proposed method will be discussed and some results will be presented. Finally, section 8 will permit to conclude.

2. COMPUTING GOOD VIEWS

Very few works exist in the domain of automatic determination of a good view direction. Colin [1] proposed an algorithm for scenes composed of octree models. This algorithm assigns a value to each view direction, depending on the type of the cubes found in this direction.

Kamada et al. [2] consider a direction as a good one if it minimizes the number of degenerated images when the scene is projected orthogonally. Figure 1 shows a good view and two degenerated ones of a cube.

The method avoids the directions parallel to planes defined by pairs of edges of the scene.

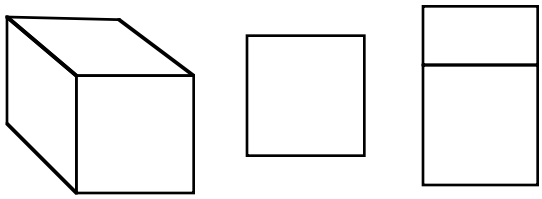


Figure 1: good and degenerated views

If L is the set of edges of the scene and T the set of unit normal vectors to planes defined by two edges, let's call $\vec{P}$ the unit vector of the view direction to compute.

To minimize the number of degenerated images, angles of the vector $\vec{P}$ with the faces of the scene must be as great as possible or angles of the vector $\vec{P}$ with the elements of T must be as small as possible. The evaluation function is the following :

$$f(\vec{P}) = \mathrm{Min}_{\vec{t} \in T}(|\vec{P} \cdot \vec{t}|)$$

Another method [3, 4, 6] uses a heuristic search of a good view direction. From a criterion of *good view* it usually applies a heuristic based on the evaluation of some view directions from which it computes other directions assumed better. These new directions are inferred from the hypothesis that a direction near a good direction is also probably a good direction.

In a first step, the scene is surrounded by a sphere. Its surface represents the set of possible positions of the observer. The sphere is subdivided by the 3 reference coordinate planes whose origine is the center of the sphere, defining 8 regions that in turn define 8 spherical triangles in the surface (figure 2). For each vertex of the spherical triangles the number of (partially or totally) visible surfaces is computed. From this information the most "interesting" triangle is derived; i.e. the one whose vertices have more visible surfaces.

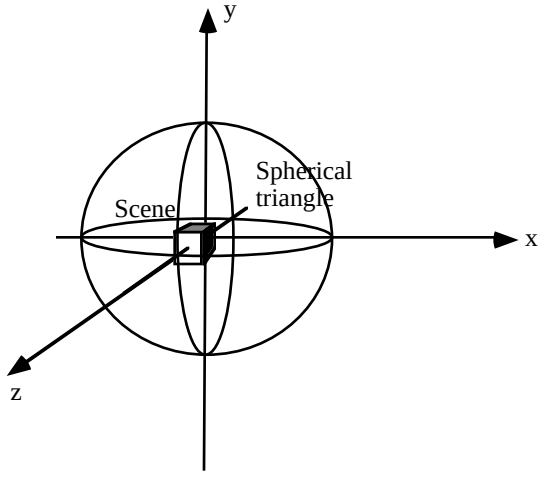
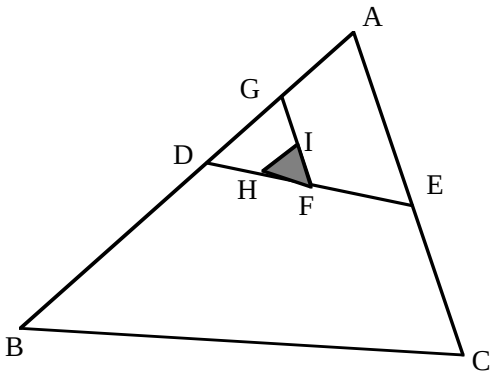


Figure 2: Division of a sphere in spherical triangles

The computation of the view point is now limited to this triangle which is recursively subdivided according to the number of visible surfaces on its vertices as shown in figure 3, where nvs (A) is the number of visible surfaces from point A.



1. Triangle ABC : nvs(A)>nvs(B), nvf(A)>nvs(C)
2. Triangle ADE : nvs(D)>nvs(A), nvs(D)>nvs(E)
3. Triangle DFG: nvs(F)>nvs(D), nvs(F) >nvs(G)
4. Triangle FIH: nvs(F)>nvs(I), nvs(F)>nvs(H)

Maximum number of steps : 3.
Thus, F determines view direction
allowing to see a maximum of surfaces.

Figure 3: Subdivision of a spherical triangle

This method was improved by introducing an additional "good view" criterion, based on the area of the projected total visible part of the scene.This new criterion is combined with the criterion of number of visible surfaces. An example of obtained results with a simple scene is shown in figure 4.

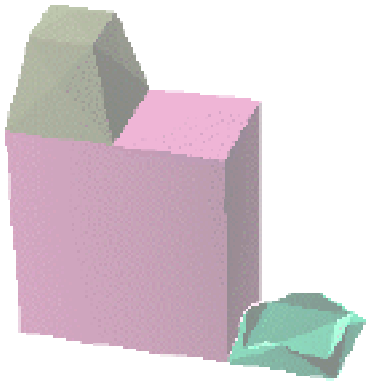


Figure 4: Good view direction for a scene composed of three objects

3. WHAT IS THE PROBLEM

Computing a good view direction is important for the user of a modeller, in order to understand a scene he (she) created or he (she) discovered in a scene library. However, the computation of a single good direction is not sufficient, in many cases, to have a good knowledge of a scene. For some scenes, several views are necessary to well understand their properties. The work of computing these views can be left to the modeller. The problem is that changing a view direction for another one can be confusing for the user, especially if the new view direction is completely different from the previous one.

A way to avoid brutal changes of view direction is to simulate a virtual camera moving smoothly around the scene. This approach generates other problems because the determination of good views must be fast enough if we want to give the user an impression of continuous movement of the camera. Moreover, sudden changes of the camera's trajectory should be avoided in order to have a smooth movement of the camera, and heuristics should be provided to avoid attraction forces in the neighbouring of a good view direction.

L. Fournier [7] gives a theoretical study of automatic smooth movement of a camera around a scene, based on reinforcement learning. J.-P. Mounier [8] uses genetic algorithms for planing paths for a camera walking inside a scene. In [9] a solver using interval arithmetic is used to obtain camera's movements satisfying user-defined constraints.

There are two techniques for understanding a scene by means of a virtual camera. In the first one the camera can **visit** the interior of the scene in order to have a closer view of the elements composing the scene. In the second technique the camera and the scene have two distinct nonintersecting universes. The camera **flies over** the scene choosing its trajectory in order to have good knowledge of the scene's properties with a minimum of displacement. In the following sections, a heuristic method to guide the movement of a virtual camera, based on the second technique, is presented.

4. A HEURISTIC METHOD TO GUIDE THE MOVEMENT OF A CAMERA AROUND A SCENE

The camera moves on the surface of a sphere surrounding the scene (figure 5). The starting point of the camera is computed using the method of calculation of a good view direction described above and in [6]. In order to have faster calculation of a good view direction, an improvement based on the hardware capabilities is applied. This improvement will be presented in section 5. In future implementations we think that combination of the used good view criterion with the Kamada's one [2] would improve the quality of the computed view direction.

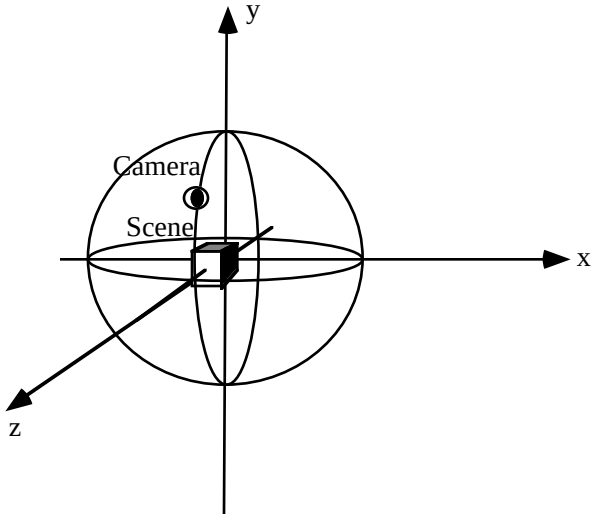


Figure 5: The camera moves on the surface of a sphere

Once a good view direction has been computed and the corresponding view point has been determined on the surface of the sphere, the camera is set in this point and the movement of the camera starts by a research of the camera's initial movement direction. Initially, all directions are plausible and 8 displacement directions are considered (figure 6).

For each possible new position of the camera on the surface of the scene, corresponding to a displacement direction, the view direction from this point is evaluated and the chosen position is the one corresponding to the best view direction value.

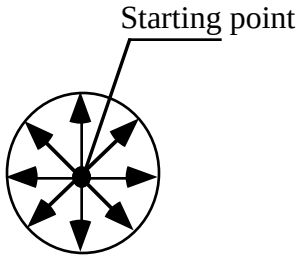


Figure 6: Starting point and possible directions

After the first displacement of the camera, a movement direction is defined by the previous and the current position of the camera. As blunt changes of movement direction have to be avoided, in order to obtain a smooth movement

of the camera, the number of possible new directions of the camera is reduced and only 3 directions are possible for each new displacement of the camera (figure 7).

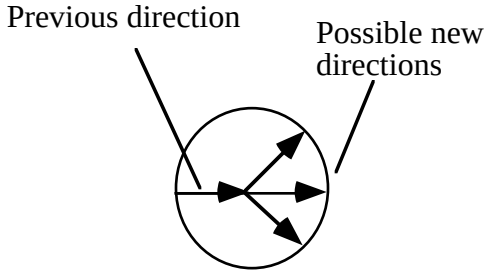


Figure 7: Only 3 directions are considered for a smooth movement of the camera

One of the three possible directions is chosen using heuristic rules taking into account not only the view direction value of a point but also other parameters permitting to avoid attraction points and cycles in the camera's movement. These heuristic rules will be described in section 6.

In the current implementation, the displacement step of the camera is constant. We think that results should be better if a variable size step is used. To do this, each camera's displacement will be the sum of two vectors :

- A variable size vector $\vec{d}$, starting at the current camera's position and whose size and direction are the previous displacement step size and direction.
- A small vector $\overrightarrow{\Delta d}$, with $\left|\left|\overrightarrow{\Delta d}\right|\right| \leq \left|\left|\vec{d}\right|\right|$, whose direction is one of the 8 possible directions (figure 8), according to the evaluation of the corresponding position using the good view direction criterion.

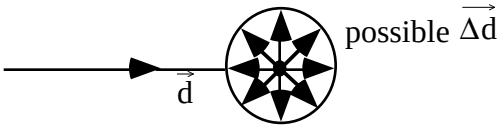


Figure 8: Possible values of $\overrightarrow{\Delta d}$

The new position of the camera will be defined by a displacement from its old position according to the vector $\vec{d'} = \vec{d} + \overrightarrow{\Delta d}$ (figure 9).

With this method, the movement of the virtual camera would be slower for regions containing several good view directions, because the displacement step should be smaller.

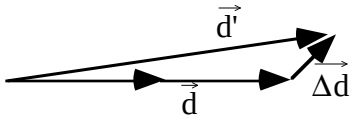


Figure 9: Computing new direction $\vec{d'}$

5. IMPLEMENTING FAST COMPUTATION OF GOOD VIEW DIRECTIONS

The problem with automatic computation of good view directions is that it is a time consuming process, hardly compatible with a real time smooth movement of a camera. So, in order to reduce the time cost of this task, we apply a computation technique using image analysis. Based on the use of the OpenGL graphical library and its integrated z-buffer, the technique used is described in this section.

If a distinct colour is given to each surface of the scene, displaying the scene using OpenGL allows to obtain a histogram (figure 10) which gives information on the number of displayed colours and the ratio of the image space occupied by each color.

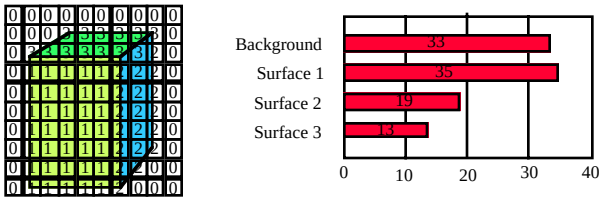


Figure 10: Fast computation of number of visible surfaces and area of projected visual part of the scene by image analysis

As each surface has a distinct colour, the number of displayed colours is the number of visible surfaces of the scene from the current position of the camera. The ratio of the image space occupied by a colour is the area of the projection of the visual part of the corresponding surface. The sum of these ratios is the projected area of the visible part of the scene. With this technique, the two good view criteria are computed directly by means of an integrated fast display method.

The importance of a view point is now computed by the following formula:

$$I(V) = \frac{\sum_{i=1}^{n} \left[\frac{P_i(V)}{P_i(V)+1}\right]}{n} + \frac{\sum_{i=1}^{n} P_i(V)}{r}$$

where: I(V) is the importance of the view point V,
$P_i(V)$ is the number of pixels corresponding to the polygon number i in the image obtained from the view point V,
r is the total number of pixels of the image (resolution of the image),
n is the total number of polygons of the scene.

In this formula, [a] denotes the smallest integer, greater than or equal to a.

The main advantages of this technique are the following:

- Approximated computing of the number of visible surfaces and of the projected area of each visible surface by image analysis is very easy. The total time cost of the technique is O(d) + O(m+n), where O(d) is the image computing cost, m is the number of pixels of the image and n the number of polygons (surfaces) of the scene.

- The display cost with a hardware acceleration based z-buffer is not important and a big number of polygons can be displayed very quickly.

The main drawback of the used acceleration technique is that, with the used OpenGL mode, the maximum number of colours is 4096. Thus a maximum of only 4095 distinct surfaces can be processed. However, sufficiently complex scenes can be constructed with 4095 surfaces. Moreover, with another colour coding model, this limit should be suppressed and the maximum number of colours should be of the order of millions.

6. CAMERA'S MOVEMENT HEURISTICS

A virtual camera could be defined as a quadruple $(p, \vec{m}, \vec{v}, \vec{b})$ where p is the position of the objective of the camera, $\vec{m}$ the displacement direction, $\vec{v}$ the view direction and $\vec{b}$ the upright direction of a virtual cameraman.

The purpose of the virtual camera's movement around the scene is to give the user a good knowledge of the scene's properties. To do this, a maximum of interesting regions of the scene must be viewed by the camera, with a minimum displacement from the starting point.

In order to avoid a fast return of the camera to the starting point, because of attraction due to the fact that this point determines a good view direction, a weight is assigned to each position of the camera. The weight of a position is proportional to the distance of the virtual camera from the starting point. What is the distance of a point on the surface of a sphere from the starting point? We define this distance as the length of the arc of the circle obtained by the intersection of the sphere by the plane defined by the starting point, the current position of the camera and the centre of the scene.

In fact, we must consider two kinds of distance.

- The path traced by the camera's movement (figure 11a). The length of this path can be computed as the sum of the chords of elementary arcs obtained by decomposition of the path (figure 11b).

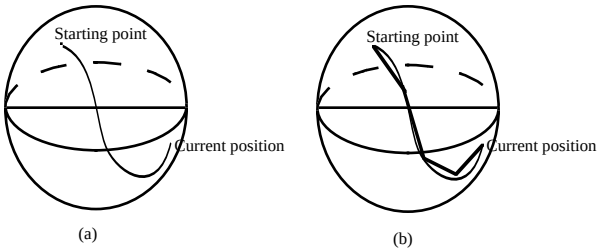


Figure 11: Path traced by the camera and approximated length of the path.

- The minimal length arc between the starting point and the current position of the virtual camera, that is, the distance of the current position from the starting point (figure 12).

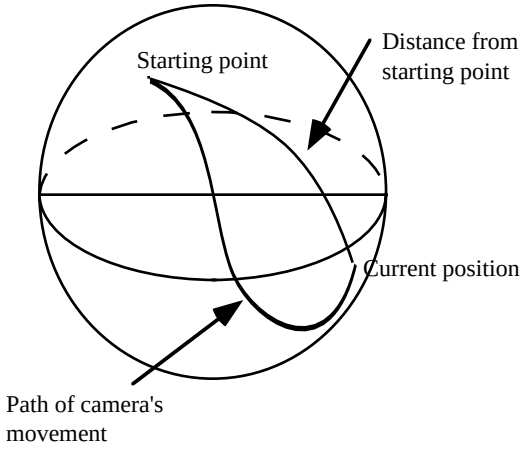


Figure 12: Distance of the current position of the camera from the starting point.

In order to create an heuristic function guiding the movement of the camera, we can observe that the importance of the camera's distance from the starting point is inversely proportional to the length of the path traced by the camera.

Thus, our heuristic function computing the weight of a position for the camera on the surface of the sphere must take into account :

- The global view direction note of the camera's position (n_c).
- The path traced by the camera from the starting point to the current position (p_c).
- The distance of the current position from the starting point (d_c).

Finally, the function we have chosen is the following :

$$w_c = \frac{n_c}{2}\left(1+\frac{d_c}{p_c}\right)$$

where w denotes the weight and c the current position of the camera.

7. DISCUSSION AND RESULTS

The method of automatic computing of a virtual camera's movement presented in this paper has been implemented and the obtained results prove the effectiveness of this method. Good view directions are computed very quickly and the camera's movement is smooth enough in its trajectory around the scene. With the majority of tested scenes, a good knowledge of the scene is reached with a minimum displacement of the camera. In a small number of cases, the initial determination of a good view direction fails but the movement of the camera permits to find quickly a good trajectory.

In figure 13 one can see the trajectory of a virtual camera around a symmetric scene. The camera's movement is smooth enough and the computed initial view direction (right end of the trajectory) is a good view direction. Moreover, all intermediate positions of the camera determine good view directions because they never lie to one of the three main axes defined by the scene.

The trajectory of the virtual camera around a pear in figure 14 is globally well chosen. The initial view direction (right end of the trajectory) is a rather good view direction.

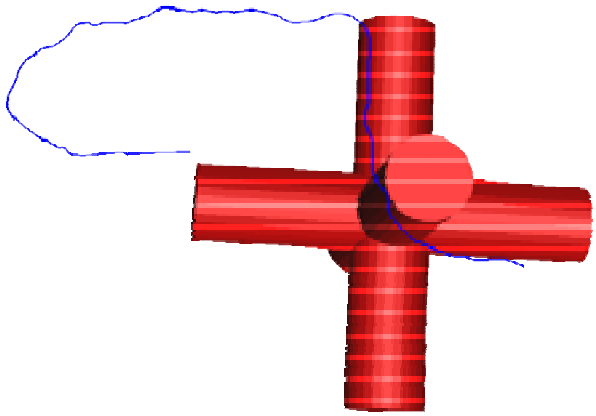


Figure 13: Smooth movement around a symmetric structure

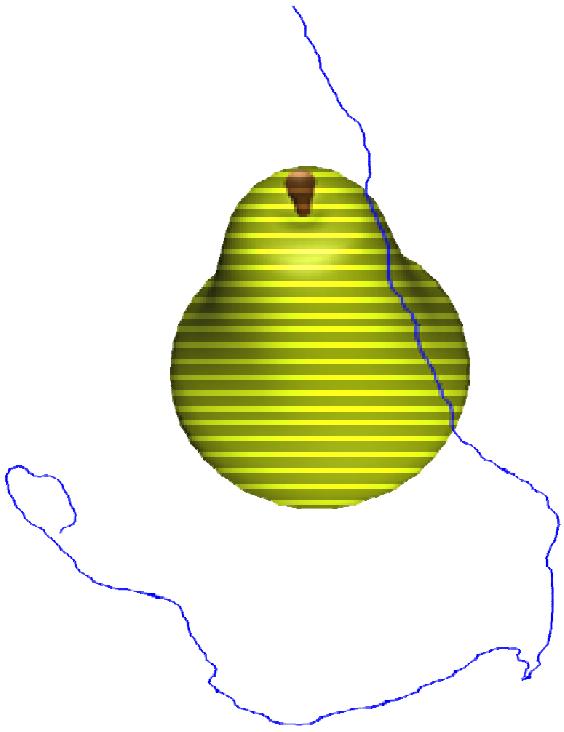


Figure 14: Camera's trajectory around a pear

Figure 15 shows the movement of the virtual camera around a complex scene. The computed initial view direction on the right end of the trajectory is a good one and the camera's trajectory is well chosen for its major part. However, the trajectory is deteriorated at the end of the movement.

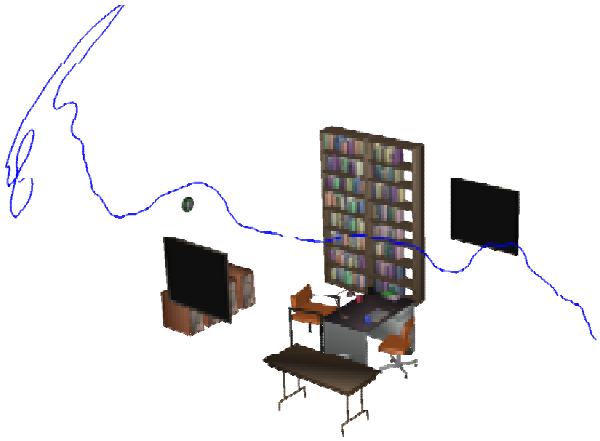


Figure 15: Deterioration of the trajectory at the end of the movement

Globally, the obtained results are very satisfactory, even if we think that it is possible to improve them by implementing techniques presented in above sections but which are not yet implemented.

8. CONCLUSION

In this paper, we have presented a new method for automatic computing of the trajectory of a virtual camera, moving around a scene. The purpose of this method is the understanding of the processed scene with a minimum displacement of the camera. Understanding a scene is very important for scenes obtained by declarative modelling techniques or scenes found by the user and on which he (she) has insufficient knowledge.

The obtained results are very satisfactory, even if, in some cases, the movement of the virtual camera is longer than necessary. We do think that the use of additional heuristics, described in the above sections and not yet integrated in the method presented in this paper, as partial use of the Kamada's technique for computing good view directions and use of variable length steps for the camera's movement would improve the efficiency of the whole process.

The method described in this paper could be improved by allowing user's interactive intervention in order to correct the camera's movement. We think that a learning mechanism could be integrated to the camera's movement, permitting to improve its trajectory according to the user's desires.

9. REFERENCES

[1] COLIN C., Automatic computing of good views of a scene, MICAD'90, Paris (France), February 1990 (in French).

[2] KAMADA T., KAWAI S., A Simple Method for Computing General Position in Displaying Three-dimensional Objects, Computer Vision, Graphics and Image Processing, 41 (1988).

[3] PLEMENOS D., Contribution to the study and development of techniques of modelling, generation and display of scenes. The MultiFormes project., Professorial dissertation, Nantes (France), November 1991 (in French).

[4] PLEMENOS D., PUEYO X., Heuristic Sampling Techniques for Shooting Rays in Radiosity Algorithms., 3IA'96 International Conference, Limoges (France), April 3-4, 1996.

[5] PLEMENOS D., Declarative modeling by hierarchical decomposition. The actual state of the MultiFormes project., GraphiCon'95, Saint Petersburg, July 1995.

[6] PLEMENOS D., BENAYADA M., Intelligent display in scene modelling. New techniques to automatically compute good views., GraphiCon'96, Saint Petersburg, July 1996.

[7] FOURNIER L., Automotive vehicle and environment modeling. Machine learning algorithms for automatic transmission control. PhD Thesis, Limoges (France), January 1996 (in French).

[8] MOUNIER J.-P., The use of genetic algorithms for path planning in the field of declarative modeling., International Conference 3IA'98, Limoges (FRANCE), April 28-29 1998.

[9] F. JARDILLIER, E. LANGUENOU, Screen-Space Constraints for Camera Movements: the Virtual Cameraman., Computer Graphics Forum, Volume 17, number 3, 1998.

Authors:

Pierre BARRAL, Guillaume DORME, Dimitri PLEMENOS
University of Limoges
MSI Laboratory
83, rue d'Isle
87000 Limoges
France
Phone: (+33) 555 43 69 74
Fax: (+33) 555 43 69 77
E-mail: plemenos@unilim.fr, dorme@alphainfo.unilim.fr, barral@novinfo1.unilim.fr