Scene understanding techniques using a virtual camera

Pierre Barral†, Guillaume Dorme, Dimitri Plemenos

MSI laboratory, University of Limoges, France

Abstract

An automatic computation of the path for a camera around a three-dimensional scene is discussed. Previous work on automatic virtual camera, based on a heuristic evaluation function, is reviewed, and new methods to improve this work are presented. Theses methods consist of additional criteria in the evaluation function as well as new ways to determine the camera's path. Ideas to animate automatically a camera inside a scene are presented, and in the end, results are discussed.

Keywords: scene understanding, automatic virtual camera, good view, in-depth search

1. Introduction

Modeling is a very important stage in computer graphics. When the user needs to deal with some complex objects during the interactive design of a scene, he (she) may need to have a good understanding of the scene he (she) is working on, or at least a better one. It is even more necessary in the case where the designer uses a declarative modeler [3, 5] to generate scenes respecting some properties he (she) defined. That is the same problem when the user gets an object from a third party (i.e. on internet, etc.), or when he (she) wants to choose one in a library, and has not yet any knowledge about it.

In such cases, we want to provide the user with a tool which would compute good view directions. The machine has more information about the scene than the user, therefore it seems natural that in some way it is more able to determine a better view than he (she) is.

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. This is already true when a human has to choose a good view by himself, but it becomes obvious when a machine has to do so without any semantic knowledge about the different parts of the scene (e.g. which direction stands for the floor and which one for the ceiling, what is the typical view people generally associate with this object, etc.).

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

This paper will present the problem and some solutions according to the following plan. In Section 2, a definition of our goal is given. Then a presentation of previous work is done in Section 3. The new techniques are presented in Section 4. Section 5 discusses of a method to produce paths inside a scene. And the first results obtained from this work are shown and described in Section 6.

2. What is the problem?

We want to provide the user with a tool which gives him (her) an automatic animation of a camera. Its movement is expected to help him (her) to understand the geometry of the scene.

The simple fact to move the camera around the objects that compose the scene allows the user to know more accu-

† in alphabetical order

rately about their depth. In addition, it seems obvious that it is necessary to represent the objects under different points of view in order to show the user their complete shape. With trivial and regular objects, it can be important for the user to be sure that there are no hidden features behind the first picture he (she) was presented.

We would like to build a technique which could automatically find a way to animate the camera so that the scene is well understood from a minimal number of displacements. We also want to perform this computation in real-time, in case the user would like to view many scenes in a row, and no time to spare (e.g. case of the solutions given by a declarative modeler).

The main method presented in this paper consists in moving the camera around the scene. A new method is briefly discussed upon which we hope to obtain better results for complex scenes.

3. Virtual camera's movement around a scene

In [1], a method to compute automatically a path for a virtual camera around a scene is proposed. In this section this method is recalled. In a first step, the scene is surrounded by a sphere. Its surface represents the set of the possible positions of the camera (see figure 1).

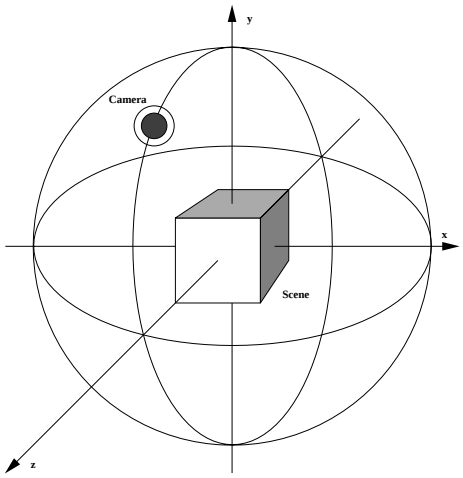

Figure 1: *the scene and its surrounding sphere*

A good point of view is computed according to the method proposed in [3, 5, 6]. The evaluation function of a point of view is based on two criteria: the number of visible faces on the screen, and the surface area that they take up in percentage of lighted pixels. These criteria can be quickly computed by an analysis of a picture rendered through a Z-buffer, and by assigning each face a unique color. Thus, it is possible to benefit of the available hardware accelerations.

When this starting point is found, the evaluation function is applied to 8 adjoining points, which are obtained by moving according to the 8 main directions around this initial position, and by a constant distance (see figure 2). The direction that leads to the best value is kept and an animation of the camera moving between the two positions is performed.

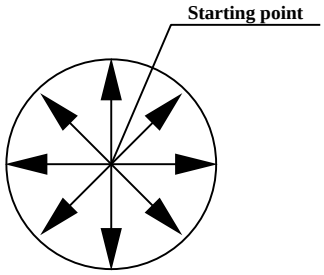

Figure 2: *starting point and the possible directions*

After this first movement, the same process is repeated by taking only 3 possible angles in order to avoid some blunt turning over of the camera (see figure 3).

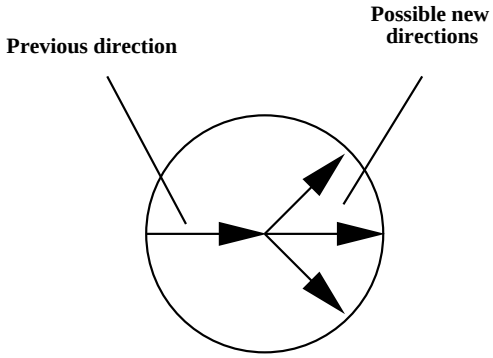

Figure 3: *only 3 directions are considered for a smooth movement of the camera*

To avoid that the camera goes back to the starting point (which has been chosen because it is the best point of view of a region) too rapidly, we proposed to strengthen the evaluation function whith heuristics taking account of the length of the camera's path since the beginning of the animation, and of its distance from the starting point (see figure 4) with equation 1.

$$F(P_c, P_i, P) = \frac{F'(P_c)}{2} \times (1 + \frac{Dist(P_c, P_i)}{Length(P)}) \qquad (1)$$

With: P_c the current point of view, P_i the initial point of view, P the path going from P_i to P_c, F' the basic evaluation function.

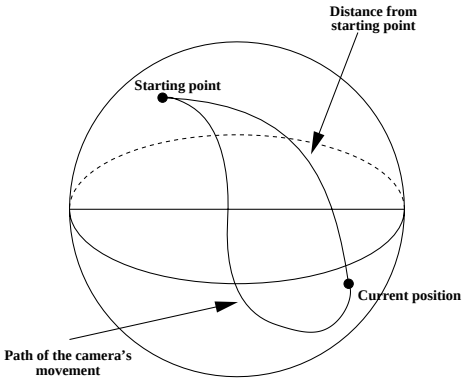

Figure 4: *distinction between the path and the distance between two points*

4. New techniques

Starting from the technique presented in Section 3, some other factors have been tested in the evaluation function. Two variants were also implemented in the way to choose the next position of the camera.

4.1. New heuristics for the good view criterion

New heuristics have been added to those presented in Section 3 to improve the basic evaluation function of a point of view, used before considering the distance to the starting point.

4.1.1. Kamada's coefficient

The Kamada's coefficient (equation 2) is based on propositions made by Kamada in [2] in order to avoid the presence of degenerated edges (i.e. edges that are almost merged when projected on the screen) in the picture.

$$CoeffKamada(DoV,F) = \min_{f \in F}(|DoV \cdot Normal(f)|) \quad (2)$$

With: DoV the direction of view (normalised vector), F the set of faces.

However, the criterion proposed by Kamada was designed to be used for a representation with orthogonal projection, when a perspective projection is used. Because of that, a face which is parallel with the direction of view will be always regarder as bad, whereas if it is far from the center of the screen it will be visible and comprehensible. On the other hand, a face which would be inclined, but badly placed compared to the position of the camera, would look flat, but would be regarded as good.

So it would be more clever to modify Kamada's criterion by replacing the direction of view by the direction of the vector going from the position of the camera to the barycentre of the face.

4.1.2. Balance coefficient

The figure 5 shows a scene made up of two cubes aligned on an axis. In this case it seems right to say that the right picture is better than the left one. Unfortunately, according to the former heuristics, the left one was better rated.

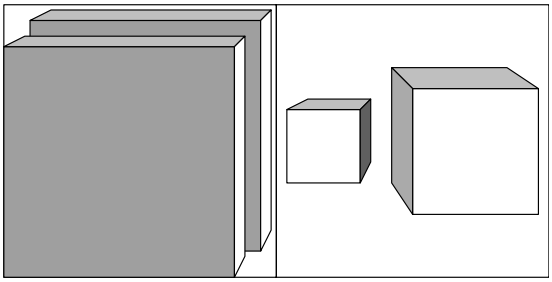

Figure 5: *two boxes almost aligned with the direction of view*

In this case, we think it would be more interesting that the visible faces of the scene take up an area, on the screen, proportional to their real area. In addition, in [7], Fournier proposes to use the ratio of the visible area with its real area, in order to help the little faces to appear, since the user will not miss the very big ones anyway.

To express this, we try to minimize the standard deviation of the series given by the ratio: "visible area (taking into account the visible parts)" by "real area" of the faces (equation 3).

$$CoeffBalance(VF) = -\sqrt{\frac{\sum_{f \in VF}(X_f - XM)^2}{Card(VF)}} \quad (3)$$

With: VF the set of visible faces,
$X_f = \frac{VisibleArea(f)}{RealArea(f)}$,
XM the mean of X_f.

4.1.3. Exploration coefficient

The exploration coefficient tries to help the camera to present new elements of the scene to the user, and is made up of 3 elements which are summed up.

The basic idea is that it seems more natural that a user have a complete knowledge of the scene if he (she) sees all the faces that compose it. So it was decided to mark all the faces that appear on the user screen in a first step.

Then, for a given point of view, one part of the exploration coefficient is the ratio between the number of unmarked faces at this instant, by the total number of faces (equation 4).

$$CoeffExploNbFaces(VFne,F) = \frac{Card(VFne)}{Card(F)} \quad (4)$$

With: $VFne$ the set of visible faces not explored yet, F the set of faces.

A second part is added to this coefficient. Its goal is to facilitate the movement of the camera towards places that would allow to show faces that have not yet been seen.

In a trivial way, the best way to see a face is supposed to be when it is well perpendicular to the direction of view. It was then decided that a point of view would be better than another if it could reduce the angle between the normal of the face and the direction of view. Using the cosinus of the angle as a measure for the angle, we must try to maximize the equation 5.

$$CoeffExploAngles(DoV, VFne) = \frac{\max_{f \in VFne}(DoV \cdot Normal(f)) + 1}{2} \quad (5)$$

With: DoV the direction of view (normalised vector), $VFne$ the set of visible faces not explored yet.

It was also decided to take into account with equation 6 the area on the screen of the faces appearing on screen for the first time.

$$CoeffExploAreas(VFne, Resol) = \frac{\sum_{f \in VFne} VisibleArea(f)}{Resol} \quad (6)$$

With: $VFne$ the set of visible faces not explored yet, $Resol$ the resolution (in pixels) of the picture used to compute visible area of faces.

It was chosen to consider a face as visible (from the exploration's point of view) only when it is represented by a significant number of pixels on the screen and if the angle between its normal and the direction of view is less than 45 degrees, according to the advises given by Kamada.

As one can see in the equation 5 (coefficient based upon the angles of the unmarked faces), the direction of the normal is very important because the dot product can give negative values.
However, this case only happens in a very low number of cases: there must be a visible face oriented towards the back of the screen rather than towards the user. In the case of solid objects (i.e. objects on which backface removal cannot lead visible faces to disappear), this is impossible. In the case of non-solid objects (i.e. objects with gaps), this allows to prefer some parts of the scene to be seen from one side rather than from another. If there was no reason to prefer one side to another, an absolute value could be used around the dot product in the equation.

The three factors are summed to give the exploration coefficient (equation 7).

$$\begin{aligned} CoeffExplo(VFne, F, DoV, Resol) = \\ CoeffExploNbFaces(VFne, F) \\ + CoeffExploAngles(DoV, VFne) \\ + CoeffExploAreas(VFne, Resol) \end{aligned} \quad (7)$$

With: $VFne$ the set of visible faces not explored yet, F the set of faces, DoV the direction of view (normalised vector), $Resol$ the resolution (in pixels) of the picture used to compute visible area of faces.

4.1.4. The new evaluation function

The evaluation function will be a sum of all the coefficients discussed in this section. Each one of them was selected in order to find a value in the same interval. A priori, this property gives to each one an importance equivalent compared to the others. However, if the user wishes to privilege certain coefficients compared to others, it is possible to carry out a weighted sum of it (equation 8). We did not really succeeded in determining the influence of slight modifications of the weights on the general behavior of the camera, so, for a normal use, one can be satisfied to take the same values.

$$\begin{aligned} F' = & FormerF' \\ & + W_K.CoeffKamada(DoV, F) \\ & + W_B.CoeffBalance(VF) \\ & + W_E.CoeffExplo(VFne, F, DoV, Resol) \end{aligned} \quad (8)$$

With: $VFne$ the set of visible faces not explored yet, F the set of faces, DoV the direction of view (normalised vector), $Resol$ the resolution (in pixels) of the picture used to compute visible area of faces, VF the set of visible faces.

4.2. Determining the path of the camera

4.2.1. A variable size displacement

In this variant, our will is both to increase the number of candidate points of view, and to provide a new information based upon the speed of the motion. The position of the camera is computed by performing a second test while studying one branch. Once the position of a possible point of view is computed, 8 close points surrounding it are examined, in the same way the best direction was chosen to start the animation at the very beginning of the method (see figure 6).

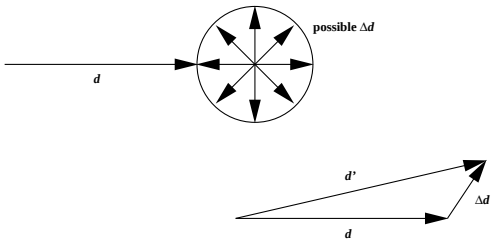

Figure 6: *research in two movements*

Among the 8 possible points is kept the one which gives the best result with the evaluation function. Since this is done for each of the three possible branches, it produces 24 positions. All those positions are not equally distant from the point which is the root of the research tree, and if the animation is performed by interpolating the position of the camera between the two extremities with a constant number of points of view, the difference between a far and a close position will be translated for the user by a modification of the rotation speed. Thus, a fast rotation can help him (her) to "feel" that between two points of view there is no significant elements to see, and a slow rotation will mean that the computer thinks that there are more elements to pay attention to.

This functionality could be disturbed by slowdowns of calculations, but we did not notice artifacts of this nature. One can explain this apparent stability by the fact that the share of calculations resting on the visibility of the scene is not very significant, and that a part of these calculations have to be done independently of visibility. Moreover, the computation of visibility itself is dealt with by techniques of Z-buffer accelerated by the hardware. Since no backface removal is performed, the computing speed is especially conditionned by the area occupied by the faces on the screen. With the constraint of a camera fixed on a bounding sphere, one can think that the variations of this parameter are not very large, although we did not make statistical calculations to affirm it.

4.2.2. In-depth evaluation

This variant consists in reproducing an Artificial Intelligence method used in strategy games for a single player.

This time, when one of the usual candidate points of view is considered, the searching process is re-iterated. It can be performed recursively until a chosen depth (see figure 7).

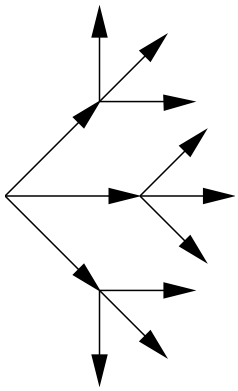

Figure 7: *recursive search*

It is important to note that the complexity of this method is rather heavy. When the basic process evaluates N points of view at each step, and when a depth of D steps is used, the total number of evaluations is : N^D. Another flaw is that according to the angle of each step, and N and D, it can happen more or less often (and more often than less), that some of the points of view are very close. If the evaluation function is supposed not to include major discontinuities, this means that there will be no great differences in the results of computing this function for close positions of the camera.

Since we were not really sure of what was best between a path that leads to a better point of view after D steps, and a path that leads to a better point of view among this same number of steps, we have implemented two variants of this method. The first one only considers the positions of the camera at the leaves of the routes tree, and the other one also takes into consideration the positions at its nodes.

5. Camera's movement inside the scene

The techniques consisting in moving a camera around the center of a scene are better designed for small scenes, composed of few elements, and ideal for a single-object scene.

When dealing with a closed scene, such as the inside of a building, or really big scenes made up of many objects not linked together, or at least with some spaces in-between so that a camera could go through them, this method is no longer sufficient. In addition, with scenes composed of objects that are not nested together, and since the camera stays at a constant distance from the center of the scene, this means that, most of the time, many of the objects will be invisible due to their too small representation on the screen.

Because of that, we need to provide the user a new method that will compute routes for a camera inside any kind of scene. Many path planning algorithms are studied in the world, and in many approaches. The problem solved by such methods is usually a robot (or anything that is able to move) which has to start from one point and end to another point. Then these algorithms try to find the best way inside the scene to connect the two points.

In [9], Jardillier & Languenou present a tool that computes camera's movements according to constraints provided by a user. The goal of this work is to provide artists in computer generated animation a way to obtain paths of a camera. The constraints are image-driven: the user informs the program about elements he (she) wants to see in the end of the process, such as an approximate place where an object must be projected... This work is based on a declarative modeling approach, so a certain number of solutions are generated.

In [8], a genetic algorithm enhanced for route planning is presented. Though, this algorithm needs additional information about the scene. The possible places where the robot can go through must be divided into cells, which must have intersection areas with their neighbours. This process is currently done by a human, so we can't yet rely on this algorithm, although it should be good if we could in some way generate

these cells automatically. In addition, the cells must be rated in order to say wether the robot is encouraged to visit them, or repelled by them.

Our problem here is that we have a priori no information about where it would be good for the camera pass. And even more: if we had decided of a route inside the scene going from P_1 to P_n through intermediate points ($P_2 \dots P_{n-1}$), then what is the best so that the user gets the most information? That the camera uses the path from P_1 to P_n, or that it takes it from P_n to P_1?

For the moment, we wish to find a method general enough, based upon a quality criterion of the point of view, like for the observation method of objects by rotation around their centers, and try to improve the used heuristic.

In the method currently considered, a discretisation of the space is performed. The matrix obtained is composed of voxels. Each voxel carries a number of information which allows to estimate the interest of letting the camera passing through its center.

A first information known for a voxel is its distance to the closest face in the scene. This criterion is important to avoid collisions between the camera and the objects of the scene (in case of null distance). This information could also be useful to keep the camera at a good distance (arbitrarily chosen, or deduced from heuristic rules) from which its vision could catch a satisfactory number of elements of the scene.

If the user knows points in the space where he (she) would like the camera to pass, it is possible to add in each voxel its distance to the closest of these points, and by the shortest path.

In this way, guiding the camera through these values, we could create, a priori, interesting paths. As we said above, these paths are not oriented, but applying an evaluation function, like the one we are currently using, at different positions of the route, and by summing its results, we could choose the orientation of the path that gives the highest value.

6. First results

The different heuristics presented have been tested. The one based upon the Kamada's technique and the one trying to balance the visible surfaces have given significant enhancements.

In figure 8, one can see that without Kamada and balance coefficients (on the first picture), the camera shows the teapot only from upward, whereas with them (on the second picture), the camera starts from under (bottom left on the figure), go above very rapidly, and then turn around a vertical axis, still from above. On this example, it is important that the camera goes under the object because this teapot has the feature of having no bottom.

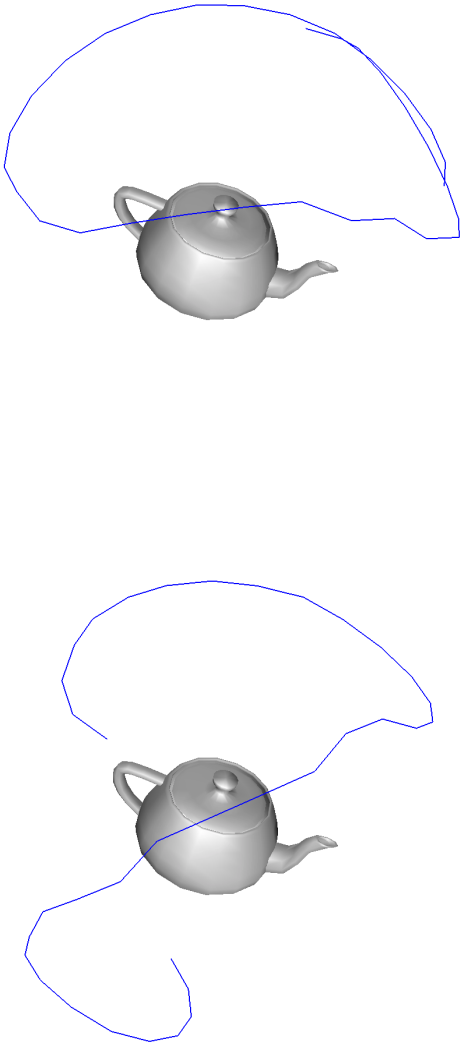

Figure 8: *examples of paths without and with the coefficients of Kamada and balance*

The criteria based upon exploration are sometimes insufficient. First, it must be noted that to mark the visible faces during the displacement of the camera, it is necessary to perform the rendering used for the evaluation process a second time. Even if this rendering is partially done through hardware acceleration, it is still necessary to give the user a good number of steps during the animation, and thus marking process becomes time consuming. Only one image upon two could be marked, but it still would make the animation look slower.

In addition, it was found that on some cases this coeffi-

cient can be awkward. Actually, those problems are essentially due to the exploration coefficient by the angles. It was noted that when the objects are containing holes (like a glass or a goblet), the method suffers of a drawback. When trying to minimize the angle between the direction of view and one of the lateral faces inside the hole, the camera tries to go in front of it. Unfortunately, because of the topological nature of this part of the object, by going in front of this face, some other faces of the hole hide the one that was targeted. We then get infinite loops in the path, when the camera tries to face as well as possible faces that it will never see directly, whereas the only way to see them is to watch them inclined. Figure 9 presents the example of a glass with which the camera doesn't manage to move vertically significantly enough to get out of this loop.

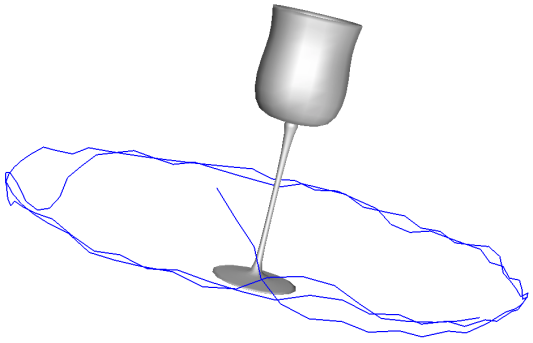

Figure 9: *exploration failure in the case of an object containing a hole*

The new methods to determine the path (Section 4.2) were also tested. It is once again obvious that these methods, using a great amount of additional measures, are much more slower. In such conditions, and with the hardware configuration used (Silicon Graphics O^2), it is not possible anymore to compute the path in real time. It is however one of our goals. So it appears that a compromise must be done between computation complexity and the speed of their process.

It was noted that the variations in speed expected from the first method are not always very easy to explain. This remark can mean that the evaluation function gives results that are not yet in good adequacy with the human estimation.

In figure 10, the dark path is taken when observing the object with a variable speed, and the light path is the usual one. It is difficult to perceive the changes in the length of segments, but they are clearly felt during the animation. They

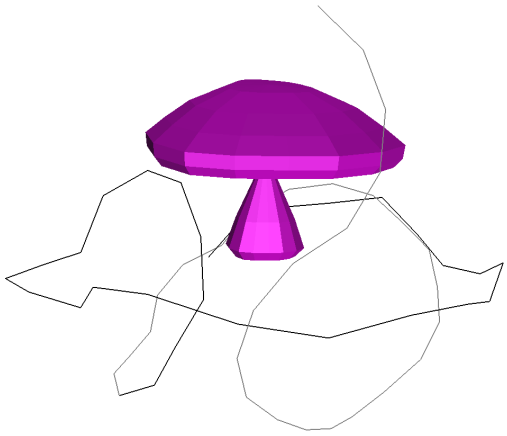

Figure 10: *examples of paths with and without variable speed displacement*

essentially appear in slowing downs of the camera when it is getting close to the base of the mushroom hat.

Here the routes are starting under the foot of the mushroom (bottom left on the figure). As we begin to see it at this step of the animation, the normal path will lead the camera above the hat of the mushroom after it has turned around the foot, whereas the path with variable speed will never show this part of the object.

The speed variations are interesting, but they are not always easy to understand, and the routes obtained with them can sometimes be much less interesting in the end.

Table 1 shows, on the example of the panel body of a car (figure 11), some statistics about the results given by the evaluation function, on the set of points of view selected during animation. The rows describe the results obtained by using different levels of in-depth research, whereas the columns present the results at different keyframes of the animation: the very beginning (steps 5 and 10), after a few moments (step 50) and a significant number of displacements of the camera (step 200). For each combination of level and keyframe, the first number in the cell shows the average value of the results given by the evaluation function on the points of view chosen since the beginning of the animation. The second number shows the standard deviation of the results. This measure gives an idea of the regularity of the quality of the path.

This table shows that increasing the depth does not necessarily improve the quality of the chosen points of view. For example, with a depth of 3, one can see at step 10 a better mean than with normal path, but the rest of the time the re-

Depth / Nb steps	5	10	50	200
1	0.707 0.062	0.574 0.142	0.646 0.105	0.614 0.109
2	0.692 0.070	0.638 0.077	0.621 0.082	0.643 0.077
3	0.688 0.057	0.609 0.092	0.593 0.050	0.592 0.037
4	0.715 0.022	0.674 0.044	0.690 0.044	0.702 0.041

Table 1: *examples of paths with various in-depth search levels*

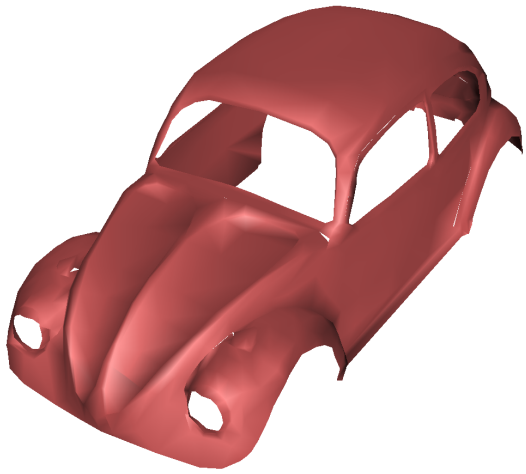

Figure 11: *the panel body of a car, used to compare different in-depth search levels*

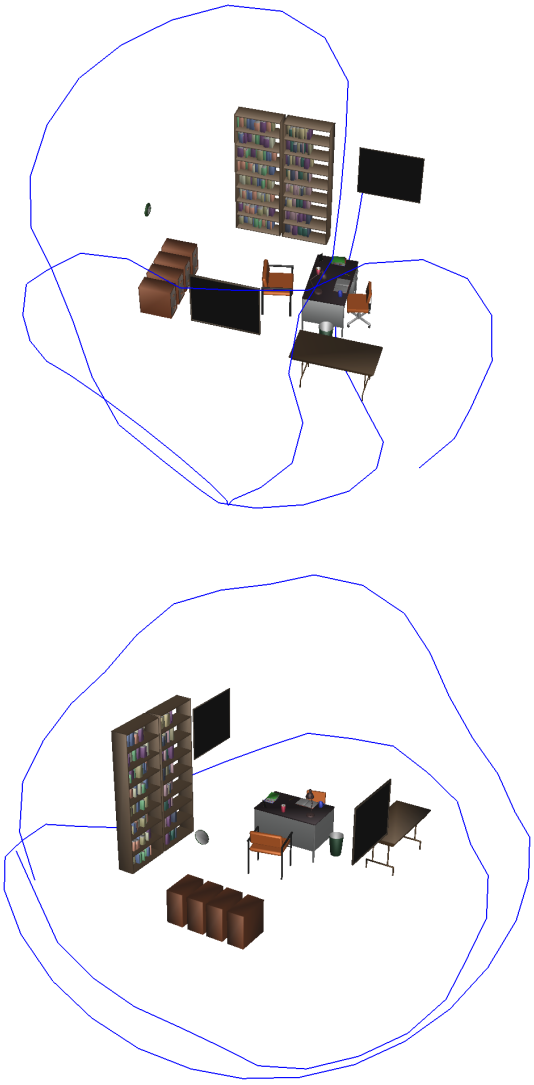

Figure 12: *comparison between the old results and the new ones on a complex scene*

sults are less interesting. At the other hand, with a depth of four, the results are always better.

On the contrary, at the exception of one particular case, the standard deviations are always better, which means that the results are more regular.

As a synthesis, it can be said that the use of the new criteria of good point of view gives a good improvement of the behavior of the camera in most of the cases. But the methods proposed to determine the path do not give such significant variations. This is even more true for the in-depth search, that does not give improvements a human can perceive.

In figure 12, we can see that the expected enhancements brought by the new criteria are not always visible.

The first image presents the path of the camera obtained by the method presented in section 3. The camera starts from behind the board next to the library, does one and a half turn around a horizontal axis, from bellow, and then performs a round around a vertical axis.

The second image presents the path obtained by the new criteria of good point of view. This route starts behind the library (left hand in the picture), goes under the floor level, performs one and a half turn around a vertical axis, staying below the floor level, and then goes above the ceiling to go back to the starting point.

It is very difficult in this case to say if one result is better

than the other, so we consider there is no enhancement on this scene.

The method proposed to explore a scene with internal paths is currently being experimented, therefore no results about it can be presented yet.

7. Conclusion and future work

A method that allows to obtain a rather good discovery of a scene by rotating the camera around its center was studied. It gives satisfactory results on many scenes, although some of the techniques mentioned did not give conclusive enhancements. Now, we would like to find a criterion to stop the animation. A completion percentage of the overlap of discretisations of the surrounding sphere at different levels of detail could be a good measure. In addition this tool could certainly be used to re-design the criterion of exploration since its implementation is not yet really relevant.

The problem of the automatic walkthrough inside a scene is now being investigated, in order to improve results on complex scenes made up of several objects.

References

1. P. Barral, G. Dorme, D. Plemenos. Visual Understanding of a Scene by Automatic Movement of a Camera, *GraphiCon'99*, Moscow (Russia), September 1999.
2. T. Kamada, S. Kawai. A Simple Method for Computing General Position in Displaying Three-dimensional Objects, *Computer Vision, Graphics and Image Processing*, 41 (1988).
3. D. Plemenos. Contribution to the study and development of techniques of modeling, generation and display of scenes. The MultiFormes project. Professorial dissertation, Nantes (France), November 1991 (in French).
4. D. Plemenos, X. Pueyo. Heuristic Sampling Techniques for Shooting Rays in Radiosity Algorithms, *3IA'96 International Conference*, Limoges (France), April 1996.
5. D. Plemenos. Declarative modeling by hierarchical decomposition. The actual state of the MultiFormes project. *GraphiCon'95*, Saint Petersburg (Russia), July 1995.
6. M. Benayada, D. Plemenos. Intelligent display in scene modeling. New techniques to automatically compute good views. *GraphiCon'96*, Saint Petersburg (Russia), July 1996.
7. L. Fournier. Automotive vehicule and environment modeling. Machine learning algorithms for automatic transmission control. Ph.D. Thesis, Limoges (France), January 1996 (in French).
8. J.-P. Mounier. The use of genetic algorithms for path planning in the field of declarative modeling, *International Conference 3IA'98*, Limoges (France), April 1998.
9. F. Jardillier, E. Languenou. Screen-space Constraints for Camera Movements: the Virtual Cameraman, *Computer Graphics Forum Volume 17*, 1998.

Authors

Pierre Barral, Guillaume Dorme, Dimitri Plemenos
(in alphabetical order)

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@msi.unilim.fr, barral@msi.unilim.fr