CubicalPath –
Dynamic Potential Fields for Guided Exploration in Virtual Environments

Steffi Beckhaus
Institute for Media Communication
GMD - IMK.VE
53754 Sankt Augustin, Germany
steffi.beckhaus@gmd.de

Felix Ritter, Thomas Strothotte
Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg
Magdeburg, Germany
{fritter,tstr}@isg.cs.uni-magdeburg.de

Abstract

Exploring unknown models or scenes is a highly interactive and dynamic process. Systems for automatic presentation of models or scenes either require cinematographic rules, direct human interaction, framesets or precalculation of paths to a known goal. We are looking for a system which can deal with rapidly changing user interest in objects of a scene or model as well as with dynamic models and changes of the camera position introduced interactively by the user or through cuts. In this paper we describe CubicalPath, a new potential field-based camera control system that helps with the exploration of virtual environments.

1. Introduction

Over the past years virtual environments for education, medicine, engineering and entertainment have become more and more popular. Often these applications either use predefined or precalculated paths for the animated presentation of scenes, or they leave the camera control entirely up to the user.

We are looking for methods and tools which support the co-existence of user-controlled movement and animation for the purpose of presenting objects or actions in a virtual environment. It should also be possible to deal with rapidly changing interests in different objects of the scene, and with fuzzy, graded goal definitions which furthermore can be used within dynamic environments.

The application environments we are addressing are known to the application but unknown to the user. That implies the ability of the application to produce a list of goals – objects that should be visualised – depending on the current needs or requests. In search engines, for example, a query may result in a list of objects matching the question in decreasing order of relevance – with a factor, stating the importance of the matching. The results are visualised in a 2D representation such as a table or a picture or, in virtual environments, by presenting the resulting objects in their environment at their usual position and context. In 3D scenes this generally requires moving through the space to view the resulting objects.

Our emphasis is placed on supplying an independent server that instantly and constantly generates data for the camera if requested. We rely on applications generating attraction values and giving further specifications if required.

Section 1.1 introduces the phrase guided exploration. Section 1.2 presents requirements for guided exploration. In Section 2 we discuss related work. Section 3 describes our CubicalPath approach and system in detail. We present an application using the CubicalPath in Section 4. Finally, we discuss the approach, the results and present ideas for future work in Sections 5 and 6.

1.1. Guided Exploration

User controlled movement in Virtual Worlds can roughly be classified into *navigation* and *exploration* by investigating the task or purpose which makes the movement necessary and the precision of the goal definition. If a goal object (also referred to as *goal*) and its position is clearly defined and the way to this goal is known to the user, the user *navigates* to the goal. If the process of finding the goal object is the task itself or if a goal is defined as a concept or question – where it is important/required that the user finds a solution or completes a task – then no predefined spatial goal exists whose position is known to the user. Rather he or she has to *explore* the space to find one or more objects to help solve a task.

A task might simply be to gain knowledge about the (spatial) composition of a complex system, or it can require doing a simulation process, such as the assembly of parts of a composite object [12]. These concepts of solving tasks and doing simulations play an important role in teaching and learning theories. As stated by Bruner in [2], for effective learning the independent acquisition of knowledge and the solving of problems is most important. Learners should solve given tasks by themselves but they may also be given help by the teacher. By this, they enquire into the subject and also train the necessary techniques. They *explore* the subject by themselves, which also teaches independence and self-learning abilities, but in case of problems they may be *guided* by a teacher. This leads to the concept of *guided exploration*.

1.2. Requirements for Guided Exploration

In this section, we discuss the requirements for a support system with the help of some applications and then define a list of requirements.

If virtual worlds are used for setting up models or experiments, then support of exploration has, among other things, to deal with the automatic presentation of parts of a model or a scene to the user. In many cases help only makes sense when the object in question is in sight. For this the viewpoint of the user or the camera has to be moved to goals while avoiding obstacles.

In *computer-based training*, simulations and virtual models, like the setup of an experiment or the visualisation of concepts, can be enhanced by the extension with guided exploration.

An example is a *Virtual Museum*, which allows to view expositions in the context of the museum for the purpose of comparing or finding the best place for an exhibit [5]. This application will be connected to a knowledge base with information about the pictures. With a speech interface it is even more likely that the user articulates frequent ideas about what he or she wants to see now, because of its easy and direct usage compared to a menu.

Another example is a *medical training system*, where the user has to fit bones and muscles together to gain knowledge about anatomy as in [12]. This application is discussed in more detail in Section 4.

This type of applications requires

- variable, not predefined goals, since the user frequently might modify tasks or form new questions while moving in the "help" or "animation" mode;
- automatic presentation which can be started and stopped any time by the user or the application;
- independence of the start position of the animation. Because the user may interactively intervene, at time t of the start of support, the camera position can be anywhere in the scene;
- the ability to weight goals, as a knowledge base may deliver fuzzy results or on purpose defines primary and secondary results.

A large area of interest for co-existing user control and animation is in the field of *game programming*. In current games the user is no longer requested to follow a certain path through a game, and even more important, the setup of the scene may vary, thus not be known in advance. The user moves freely through the scene while at some point the system takes control to present objects or events which are considered to be important. This presentation has to ensure that the user does not lose the spatial context. Nonetheless it might also introduce cuts for dramaturgical reasons.

This requires

- change of presentation priorities at any time;
- instant switch into the animation mode when important things happen in the game, and the capacity to present these interesting objects or actions.

In summary, the technical requirements can be defined as follows:

R1. independence of starting point of animation mode

R2. multiple goals which do not have to be specified in advance

R3. importance of goals can be distinguished

R4. support of frequent changes of goals

R5. just in time generation of camera data in animation mode

2. Related Work

In this chapter we present work that has been done in the field of camera path planning.

Latombe [9] presents three general concepts for robot motion planning if a start configuration and a goal is given and a path to the goal is calculated while avoiding obstacles. The presented methods are called roadmap, cell decomposition and potential fields.

Roadmap methods capture the connectivity of the free space in a network of one-dimensional curves called roadmap R. For path planning the initial and goal configurations are connected to R, and R is searched for a path between these points. The visibility graph and Voronoi diagrams are examples of roadmaps.

Cell decomposition methods decompose the free space into cells in such a way that a path between any two configurations can be easily generated. A connectivity graph is generated and with this a continuous free path is computed.

Potential field methods discretise the free space into a fine rectangular grid. A particle moves through the grid under the influence of attractive forces – introduced by goals –, and repulsive forces – introduced by obstacles – thus generating a path. Compared with roadmap and cell decomposition, potential fields are more efficient but may not always find a solution and may get stuck in local minima.

Considerable work has been carried out on methods for finding a free path between points *x* and *y* in a scene. For example, Lavalle [10] uses a game-theoretic framework for path planning. Bandi and Thalmann [1] divide the space into a 3D grid of uniform cells. Then with the A* algorithm, which is a roadmap approach, the shortest path from *x* to *y* is computed.

Another field of research deals with the presentation of predefined goals under consideration of rules for good camera views and constraints stating which object has to be viewed in which manner. Drucker and Zeltzer [4] model the methods used by a film director: logic-based constraints are defined which govern good views on the scene. With this information optimal camera positions are calculated, and with the A* algorithm a path is generated. In [6] idioms and in [8] a precalculated action plan for camera control are given. With this, a number of director's instructions are defined, which are solved with a hierarchical finite state machine.

All this research is involved with prior specified goals, and therefore does not fulfill the requirements mentioned above. In these cases a start position and one or more goal positions are predefined and the calculation of the path is done in advance.

A different problem is addressed by Hong et al. in [7]. In their Virtual Voyage the exploration of CT data of the human colon is assisted by an automated ride through the middle of the colon. From the CT data the wall of the colon is extracted and skeletonised. From this skeleton the middle of the colon is calculated. The start and end of the automatic ride is predefined through the start and end of the colon. The movement is computed through a potential field with the middle of the colon encoded as a path into the field. The user can manipulate the path with a mouse by clicking on a region of interest. A temporary goal is then set at this point and the camera is attracted to this region. In this application the possible path through the colon is clearly defined after a preparation step, and the deviation introduced by the user is minimal. This application meets requirement R5 but because of its very long initialisation step and since the overall start and goal is fixed it does not meet requirement R1 or R4.

3. The CubicalPath Method

We shall now present the key concepts of a new method of camera control, which builds on past systems but manages to satisfy all 5 requirements listed in Section 1.2. The

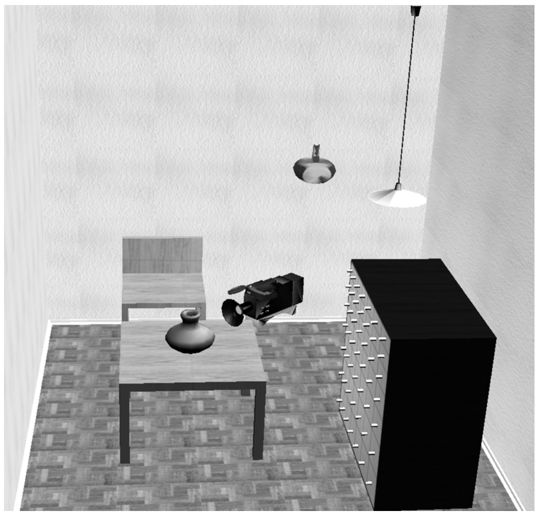

Figure 1. Example scene: room

CubicalPath system is an independent module which can be attached to a system for the purpose of camera control; it constantly generates data in form of a position and a direction vector. As input it receives the geometry information of the scene, a list of interesting objects or views, and the current camera data of the application. The camera data are the position, the direction of view, the angle of view and the up-vector. The key concept of our algorithm is that the geometry of the scene to be visualised is transformed into a coarse voxel- or *cube space*. The space spanning the model or scene is divided into a rectangular grid. Blocks of geometry data forming an object are given IDs for easy manipulation. Possible manipulations of objects are transformation, deletion or change of their attraction. Geometry can be added or manipulated at any time. The number of cubes generated for each object depends on the size of the application space and the resolution of the cube space. The camera is influenced by making objects more "attractive", thereby pulling the camera forward. This is done by raising the "attraction" value of the cubes via the objects.

For example, Figure 1 shows the model of a room. Figure 2 illustrates the representation of this scene in the cube space. Cubes with a low attraction value are visualised with a high transparency. In the cube space the attraction value of all furniture is set to a middle value, the vase on the table has a high attraction as well as the "UFO" placed to the left of the lamp. The camera is added for illustration purposes.

With this setup the camera now moves according to the rules of a potential field. All geometry is seen as obstacles, which generate a repulsive force when the camera gets close. All interesting objects generate an attractive force, its strength depending on the set value and the distance to the camera. When we speak of *geometry* in the potential field, we mean all those single cubes that are occupied by the ge-

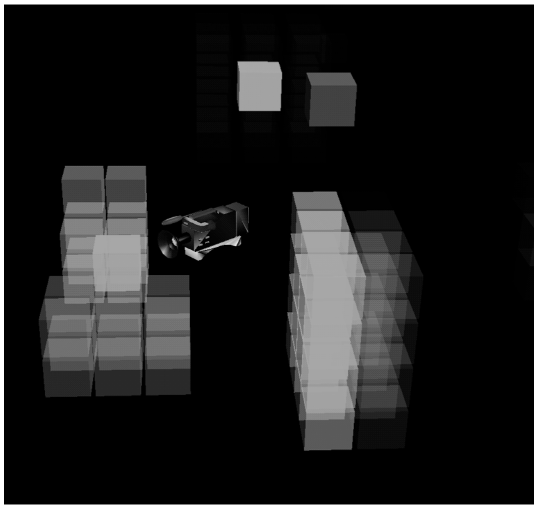

Figure 2. Room scene in cube space

ometry in cubes space. The camera itself is a particle under the influence of these forces. As the camera automatically moves through the cube space it will approach and view attractive and interesting objects. The key point therefore in our algorithm is that after an object has been in the field of view of the camera for a while, the attraction value is reduced, in effect "releasing" the camera so it can move to other objects. This behaviour results in a dynamic potential field and a camera that visits all interesting objects, one by one.

The elegant effect of our method is that changes to the camera position introduced by the user who desired to take control of the camera or through modification of the camera from a director's cut can be made at any time. The potential field is independent from the camera position and therefore does not have to be recalculated.

3.1. Potential Fields

3.1.1 The Basic Method

Potential fields are thoroughly discussed in [9]. Here we shall just give a brief introduction and subsequently extend and build on the technique.

In this method, a point q in configuration space moves as a particle under the influence of an artificial potential field U. This point may be a robot, or as in our case, a camera. The field function U can be defined over the free space as the sum of an attractive potential, pulling the particle to the goal, and a repulsive potential pushing the particle away from obstacles. At each iteration, the artificial force

$$\overrightarrow{F}(q) = -\overrightarrow{\nabla} U(q) \qquad (1)$$

is the direction of movement.

In the presence of one goal, the attractive potential field $U_{att}(q)$ can for example be defined as a conic well:

$$U_{att}(q) = \xi \rho_{goal}(q) \qquad (2)$$

where ξ is a scaling factor and ρ_{goal} denotes the Euclidean distance to the goal. The repulsive force of an obstacle should affect the particle only if getting close to some threshold. The repulsive potential function can be defined as

$$U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \rho(q) \le \rho_0 \\ 0 & \rho(q) > \rho_0 \end{cases} \qquad (3)$$

with η being a positive scaling factor, $\rho(q)$ the distance to the obstacle and ρ_0 the distance of influence.

In the presence of multiple obstacles and goals, the artificial forces have to be summed up.

3.1.2 Discussion

Compared to other methods, potential fields can be very efficient because they do not have an initialisation step and they can be implemented to work locally.

We chose this method because it is able to deal with multiple and varying goals by simply setting their attraction values appropriately high. Also, it has some implicit heuristics of dealing with graded attraction values. The attraction of a cube can not only be on or off, but can be defined over a range. The important attractive objects will have a higher attraction value than secondary objects that may or may not be interesting in a certain context. This will result in a camera visiting secondary objects on the way to prime objects, if these are close. Here the implicit rules are "view object with high attraction" and "view attractive objects which are close".

This method can also deal with dynamic environments, as new or moved geometry only has to be applied to the cube space. No further calculation is necessary to continue to generate a path as in other methods.

The existence of local minima is often stated as the major drawback of the potential field method. When the algorithm runs into a local minimum, it cannot move any further without intervention. Our approach to deal with most of the local minima issues lies in our method itself; indeed we consider local minima to be a feature rather than a bug. As we change the potential field from time to time by adjusting attraction values, the field is dynamic. Therefore in nearly all cases the camera will only temporarily stay in a minimum. The existence of a temporary local minimum is even desired when the camera should stay at some point for a while.

3.1.3 Enhancements for Camera Control

The potential field approach in itself is not sufficient for camera control. It only considers the position of the camera but not the camera view. Also, the problem of local minima has to be dealt with as discussed above. And as this method works locally, a complicated path, for example like in a maze, cannot be followed. Therefore we introduce some extensions to make the potential field method work even with applications that request a specific path or view:

- We need a *separate treatment of position and direction*. For this we specify different fields for the position and the direction. The position is calculated according to the rules of the potential field method. For the direction we will not sum up the attractive configurations but point to the highest attractive visible configuration after applying a function. This will be discussed further in Section 3.6.

- With *dynamic attraction values* we ensure that multiple goals are visited sequentially. Furthermore, the intensity of the attraction correlates with the time or duration an object is visited. The method for adjusting the attraction values is described in Section 3.7.

- *Navigation objects* will help in specifying a certain path or may prevent moving into dead ends. These objects represent spatial high-level information about the scene and are used to force a certain order. They make it possible to specify views for dramaturgical reasons. This is further discussed in Section 3.3.

3.2. Setup of the 3D Potential Field

To control the potential field we have three attributes for the objects in the CubicalPath method:

- obstacle,

- goal position, and

- goal view.

The goal position value is passed on to the cubes which are occupied by the object. When calculating the new position of the camera, this value is the attraction value for these cubes. All objects that are set to be obstacles will produce a repulsive force, when the camera gets close.

In the same way, the goal view value is applied when the new view of the camera is calculated. Here obstacles are used to check for visibility. They do not generate a force.

Any object has at least one of these attributes but may in fact have all of them. These are important attributes for camera instructions and navigation objects which are discussed in the next section.

3.3. Camera Instructions and Manipulations

To make the potential field approach work in environments, which consist of different interconnected rooms or which have special requirements concerning the path or the view, several concepts have to be introduced.

Views to objects can be defined to enable director's instructions or to provide a certain view onto an object. These views have to be defined in advance. This is done through independent areas for camera position and view, which belong to the object. If an object is attractive then these areas are made attractive for the position and the view potential field. It is possible to define many *position/view pairs* for the camera and activate them as needed. This means that the definition of goals can also be independent of objects.

We will demonstrate this with the three examples illustrated in Figure 3 and, at the same time, discuss ways of controlling the CubicalPath method:

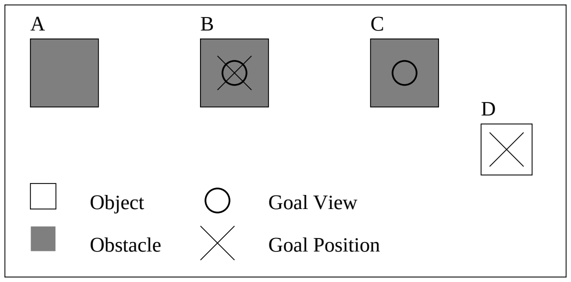

Figure 3. Obstacle, goal position and goal view combinations

All geometric objects are treated as obstacles – like object *A* – i.e. the camera may not move through objects. The distance the camera is keeping to the object is defined through the repulsive function (see function (3) in Section 3.1.1) and their values of η and ρ_0.

If an object is set to be attractive by the application, an attraction value is assigned to this object and both goal attributes are set to a value > 0 as illustrated in object *B*.

If a special view onto an object *C* is requested then a second object *D* can be introduced which forms the goal for the camera position while the goal for the camera view is still the object *C* itself.

In the case of *A* the camera avoids the object when it gets close. In the case of *B* the camera position can be at any point around the object in the distance defined by the repulsion function. In combined cases of *C* and *D* the camera moves to *D* and then will view *C*. As *D* is not an obstacle, the camera can move into the space occupied by *D*.

Navigation objects are introduced to force the camera to use a certain path or to move into a separate space. These are activated and have to be reached before the object itself is activated. They are only used if specific subgoals are to be reached along a path where there is no trivial way to the goals or where the pathway is small. One example is illustrated in Figure 4.

In illustration *A* the camera is attracted directly by the goal. It will move to the wall, which will repulse the camera. In *B* the goal's attraction is preliminarily turned off while the camera is drawn to the pathway by a navigation object. Once this is reached (see *C*) the navigation object is discarded and the goal is set attractive again.

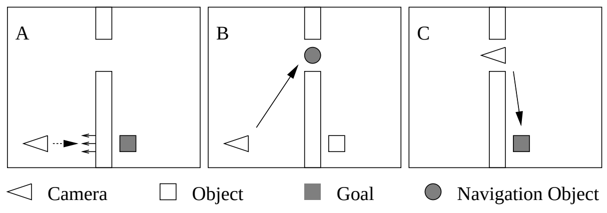

Figure 4. Usage of a navigation object

Predefined paths can be implemented through

a) defining an interesting line, or through

b) passing on of values.

Once a cube was visited, it loses its attraction, and at the same time another cube is set attractive. One cube passes its value onto the next one in the list.

3.4. System design

Based on the CubicalPath method we have developed a system designed to work as a server. The server itself is called CPServer. We use a CORBA Interface [11] that provides the necessary controlling functions for the client. This has the advantage that application and server can talk via a network and that different implementation languages and operating systems can be used. The CPAnalysisServer is a second server, which is used by the CPServer for analysing the current view (see Section 3.7). The communication between the servers and the client application is illustrated in Figure 5.

The client application controls the CPServer through several functions. It sends geometry data, attraction values for objects, sets repulsion and attraction functions, and modifies controlling parameters, for example, for the adjustment of the attraction values. Because default values are set for all functions and parameters, the application is only required to provide geometry data, attraction values and control the start and stop of camera data generation.

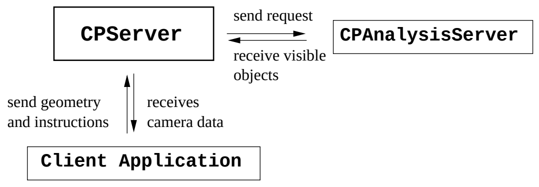

Figure 5. CubicalPath program structure

3.5. Voxelisation

The scene's geometry which is supplied as a set of triangles has to be voxelised to build the associated cube space. The same applies when geometry is moved or scaled. For the voxelisation the 3D Line Voxelisation Algorithm by Cohen [3], which is known to guarantee to visit exactly all the voxels along a 3D continuous line, is extended to deal with triangles. Consider a triangle with edges *A*, *B* and *C*, illustrated in Figure 6. From the centres D_i of all the cubes that were found on the line between *A* and *B*, a line is constructed to *C* and analysed in the same way. As these lines $\overrightarrow{DC}$ converge to each other in direction of *C*, it is guaranteed that with the above algorithm, all cubes crossing the area of the triangle are found. With $n+1$ cubes visited along $\overrightarrow{AB}$, D_0 and D_n are set to be the exact points *A* and *B*.

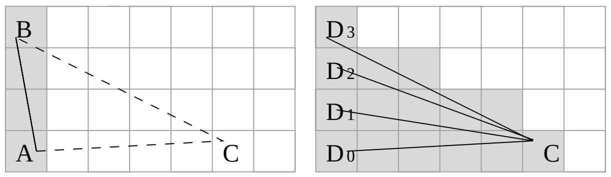

Figure 6. Voxelisation of a triangle

3.6. Calculation of the Camera View

A potential field can be considered as a fuzzy system in the presence of multiple goals or obstacles. A goal must not necessarily be reached because it may be an obstacle itself. In the case of choosing a viewing direction, it has to be ensured that goals are directly viewed. It does not make any sense to just miss the object in the viewing frustrum. With potential fields it is possible that with two equally attractive goals in the same distance the resulting vector points in the middle of these goals and none of them is seen. This is good for the camera position but inappropriate for the camera view. Therefore we chose a different approach here.

A list of view attracting cubes is generated by searching the cube space. The attraction value at each of the cubes is processed with a function taking the distance to the current camera position into consideration. The camera then points to the most attractive, visible cube. The visibility of goals

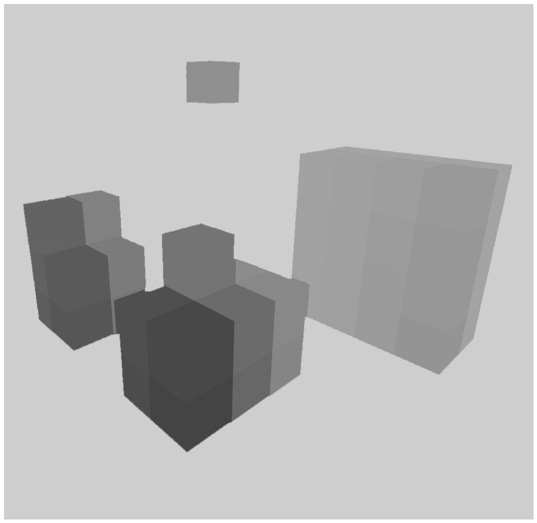

Figure 7. Output of the CPAnalysisServer

is ensured by rendering of all obstacles and goals for views and analysing the resulting images, if the requested cube is visible. This is done by the CPAnalysisServer which is described in the following section. If cubes are found to be invisible, then other cubes from the list have to be examined.

3.7. Adjustment of Attraction Values – CPAnalysisServer

A separate server, the CPAnalysisServer, receives all cubes which are obstacles or goal views and the current camera position with a list of interesting goals.

This server is used for

- finding a goal which is visible from the current camera position, and
- analysing how much of which goal is seen in the resulting view. This is an important factor for adjusting the appropriate values.

The analysis is done by rendering the cubes with the current camera settings and then analysing the resulting image. Every rendered cube gets a distinct colour as shown in Figure 7 for the "room" scenario. The colour of the cubes identifies the original position of the cube in 3D space. During the analysis, a histogram of the cubes is built. Then a list of seen objects and goals is built together with the percentual amount of these objects covering the screen. This is then used to decrease the attraction values of all goals seen depending on their visibility in the view.

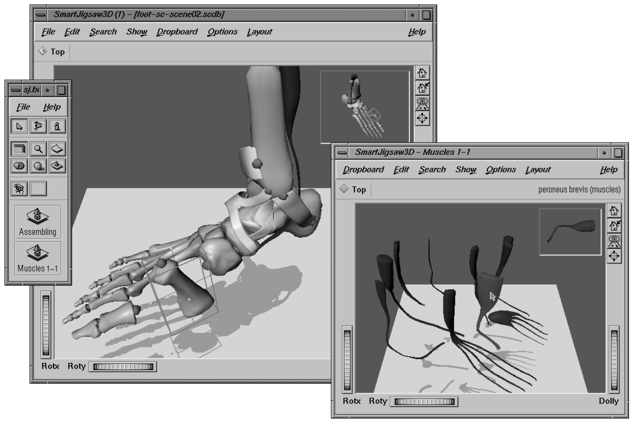
SmartJigsaw3D (1) – [foot-sc-scene02.scdb]
File Edit Search Show Dropboard Options Layout Help
Top
sj.b
File Help
Assembling
Muscles 1–1
Rotx Roty
SmartJigsaw3D – Muscles 1–1
Dropboard Edit Search Show Options Layout Help
Top
peroneus brevis (muscles)
Rotx Roty Dolly

Figure 8. 3D puzzle application showing a scenario in anatomic education

4 Applications

We applied the CubicalPath system to a virtual 3D puzzle (see Figure 8) that has been developed to improve the understanding of spatial phenomena within a complex 3D model [13]. By enabling the users to assemble a geometric model themselves we motivate them to explore the spatial relations and at the same time give them a goal to achieve while learning takes place. For this purpose, a 3D model of the subject at hand is enriched with docking positions which allow objects to be connected. Since complex 3D interactions are required to compose 3D objects, sophisticated 3D visualisation and interaction techniques are included.

While the main goal of the 3D puzzle is to directly interact with the objects – the puzzle pieces – of the 3D model, there is also a need to provide help if the user gets lost. Among other things, such as the display of textual information to a selected item, the system is able to guide the user to a specific object or object group he or she is looking for. Furthermore, it may place a chosen object at the right position onto the partly composed model or just show related objects.

Here, the CubicalPath system shows its strength by considering not only the spatial context but also by providing a means to incorporate the semantic context of the users interaction. Each object in the puzzle maintains a value indicating its current importance to the user (degree of interest – DOI). Based on the observation that most of the objects are related to each other in some way, they can pass or propagate some of their DOI to related entities. This semantic network is realised as a separate knowledge server tightly coupled with the 3D puzzle and the CubicalPath system (see Figure 9).

User interaction in the puzzle causes a shift in this interest structure which in turn is transferred to the Cubical-

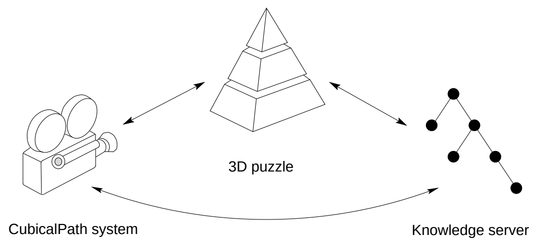

Figure 9. Cooperation between 3D puzzle and CubicalPath system

Path system according to the goal to accomplish. To give an example, if the user requests help in form of guidance to an object that has been selected from a list, this object gets a high DOI. Other objects in the current task context, e.g. to place missing bones on a partly composed skeleton, receive some of this interest, e.g. other bones that are scattered in the immediate vicinity and which have to be docked on the skeleton nearby. Therefore, the CubicalPath system also presents these objects while guiding the camera – thus the user – to the requested object.

The ability of CubicalPath to control the DOI by itself leads to another useful feature. Since objects that have been shown to the user for some time lose some of its interest (the DOI decreases), the camera automatically pans to the next interesting objects if the user does not interact with the 3D puzzle. Hence, the system generates a dynamic animation reflecting the user's learning context.

4.1 Results

The 3D puzzle uses the complete 3D space. In a museum the user normally moves through the space at fixed height and, therefore the camera movement generally is reduced to 2D. Because we move in 3D space here, we occasionally had situations where the camera approached an object from the top. This was prevented by the use of a navigation object.

In a first trial we found the 3D Puzzle and the CubicalPath working together well. The presentation of objects of interest succeeded well in the complex scene of the foot (recall Figure 8). The application also generated view/position pairs for goals to enforce a special view on objects or sets of objects. It showed that with the CubicalPath the objects could be presented in the desired way.

We tested the CPServer on an SGI Octane EMXI with 2xR12000 and 300MHz. The models of the foot we used consisted of 11,244 polygons for model *A* and 45,952 polygons for model *B* in form of triangle strips.

The resolution of the cube space is defined by the number of cubes *x*, *y*, and *z* in these axes. The time *v* needed for the conversion of the triangles into the cubes space depends on the size of the cube space (see Figure 10). This is the time to convert the complete model before the camera data can be calculated by CubicalPath. As can be seen, the voxelisation is time consuming in the presence of a large cube space. However, in the progress of guided exploration in dynamic models it can be expected that only relatively few objects will be modified at a time. This will dramatically reduce the conversion time even in large resolutions of the cubes space, when objects are modified or added.

model	*resolution/cubes*	*v/s*	*i/s*
A	10^3	0.32	0.005
	25^3	0.39	0.03
	50^3	0.50	0.27
	100^3	0.88	2.13
B	10^3	1.18	0.005
	25^3	1.22	0.03
	50^3	1.42	0.27
	100^3	2.62	2.12

Figure 10. Speed of voxelisation v and iteration i, time measurements in seconds. Model A consists of 11,244 polygons, model B consists of 45,952 polygons

The calculation of one iteration *i* of the CPServer finished within milliseconds in all but the last of these cases. This is sufficiently fast and results in a smooth camera path, as in our example the model was properly resolved by a resolution of 50^3 cubes.

5. Discussion

We have presented a new method, called CubicalPath, which provides continuously generated camera data on request. It can deal with rapid changing interest in objects, multiple goals with different degrees of interest, and it works independently of the start position of the camera. This is achieved through the use of dynamic potential fields. The potential field adjusts itself when objects of interest are viewed by the camera. Therefore, objects lose their attraction and after a while the camera moves to the next interesting object. Precomputation only involves the voxelisation of geometry. The path itself is calculated on the fly. This results in the nice feature that no unnecessary calculations are done in the case of changing goals. We showed that with some extensions the advantages of potential field methods can be combined with the advantages of other methods.

The approach of using potential fields has some further advantages. As the geometry is completely transformed into cubes, collision detection, if required, is much easier. A big advantage in scenes with a large amount of geometry is that the speed of calculation is independent of the complexity of the geometry. The CubicalPath method operates only on the number of cubes that define the cube space. It uses an "abstract" and simplified version of the geometry data through its cubes.

Certain issues using potential fields still have to be addressed. One is that the camera could end up alternating between two objects A and B having the same attraction, instead of first viewing A for a while and then proceed to B.

The case of unwanted local minima can still occur. Any goal may produce a desired minimum. In the presence of attractive, visible objects, the dynamic nature of the potential field will eventually move the camera out of minima. But if the camera moves into a local minimum and the algorithm will not change any values in the field, the camera gets stuck. A situation like this has to be detected and dealt with, for example, through a randomly chosen way out of the minimum or through flattening out the potential at this position [9].

Nevertheless, these problems only tend to occur in scenes with a lot of geometry and in the presence of a large number of goals. Usually the algorithm deals well with these local problems through the dynamic nature of the cubic space.

A problem with potential fields in 3D is the "prettiness" of the path. As there is no real planning process involved, the camera might choose ways which lead to approaches of a goal from the top, or the use of a long way, instead of a short one. This is possible when the direct connection to the goal leads into a dead end. A prior search could prevent this. The CubicalPath system solves this problem by introducing camera instructions, since the concept of building a path to a goal does not exist in the CubicalPath system. These instructions have to be supplied by the application. Any system generating a "high-quality" camera path can pass its results into the CubicalPath system and therefore extend its own features by a system which allows guided exploration and free movement of the user.

6. Future Work

We plan to extend the system to use a hierarchical representation of the cube space to speed up the calculation. Also, we plan to automatically generate navigation objects if required.

The CubicalPath approach will be tested with different scenarios like 3D Games and in the Virtual Museum [5]. For the virtual museum an art database and a user profile will provide input for the attraction of art objects.

A formal user study has to verify whether the generated path meets the expectations of users. In virtual environments "seasickness" can easily occur and the spatial context can be lost if the camera moves too fast or uses unpleasant paths. Plus the question of how the users react if they and the system both work on the camera has to be examined. The application has to ensure that only one of them modifies the camera at a time and that the user can stop the camera movement any time.

A further user study should evaluate the concept of guided exploration in virtual environments by comparing the success of solving tasks with and without the help of guided exploration.

References

[1] S. Bandi and D. Thalmann. Space discretization for efficient human navigation. In *Proc. of Eurographics*, volume 17, pages 195–206, 3 1998.

[2] J. S. Bruner. *The Process of Education*. Harvard University Press, 1976.

[3] D. Cohen. Voxel traversal along a 3d line. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 366–369. Academic Press, Inc., 1994.

[4] S. M. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Proc. of Graphics Interface*, pages 190–199, Banff, Alberta, Canada, 1994. Canadian Information Processing Society.

[5] G. Eckel. A virtual exhibition design environment, 1999. http://viswiz.gmd.de/∼eckel/publications/eckel99f/.

[6] L.-W. He, M. F. Cohen, and D. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proc. of ACM Siggraph*, pages 217–224, 1996.

[7] L. Hong, S. Muraki, A. Kaufmann, D. Bartz, and T. He. Virtual voyage: Interactive navigation in the human colon. In *Proc. of ACM Siggraph*, pages 27–34, 1997.

[8] P. Karp and S. Feiner. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proc. of Graphics Interface*, pages 118–127, 1993.

[9] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[10] S. M. Lavalle. *A Game-Theoretic Framework for Robot Motion Planning*. PhD thesis, University of Illinois at Urbana-Champaign, Illinois, 1995.

[11] OMG. CORBA specification. http://www.omg.org.

[12] B. Preim, F. Ritter, and O. Deussen. A 3d-puzzle for learning anatomy. In *Proc. of 2nd International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 750–757, Cambridge, UK, 1999.

[13] F. Ritter, B. Preim, O. Deussen, and T. Strothotte. Using a 3d puzzle as a metaphor for learning spatial relations. In *Proc. of Graphics Interface*, pages 171–178, Montréal, Que., Canada, 2000.