OBJECT-CENTERED NAVIGATION IN VIRTUAL CONSTRUCTION APPLICATIONS

Colette Elcacho, Thomas Dingel, Reinhard Klein

Department of Animation and Image Communication
Fraunhofer Institute for Computer Graphics, Rundeturmstr. 6,
64283 Darmstadt
Germany
{elcacho, tdingel, rklein}@igd.fhg.de www.igd.fhg.de/~{elcacho, tdingel, rklein}

ABSTRACT

In this paper we present a novel concept for navigation in virtual environments. While a variety of navigation metaphors have been proposed for immersive VR, desktop scenarios are typically based on mouse navigation, using a flying, walking or driving metaphor. Steering all six degrees of freedom in this way is a complicated task even for a trained user. We propose an innovative object-centered concept for navigation, which allows exploring an unknown environment by directly going to the objects one desires looking at closely. The user specifies the object by its name, e.g. "table", an aspect, e.g. "front" and a distance level, e.g. "near" in an intuitive fashion and is immediately transferred to the desired look-at point. This is done by pointing to an object with the mouse or using a simple speech recognition approach. The object-centered viewpoints are computed dynamically and need not be predefined. Our new navigation concept has been approved by different users and has been tested in an interactive construction environment.

Keywords: navigation, object-centered, 6DOF, interaction, human computer interaction, construction kits, virtual reality (VR), desktop VR.

1. INTRODUCTION

Moving through a virtual environment in an intuitive fashion is a dream that has not yet turned into reality, a tale told by the many publications related to the subject.
The focus of this work is on the development of new navigation techniques for virtual 3D-construction kits where the user virtually constructs his model from single parts, like in a Lego game.
Special snap points and special snapping behavior on all the parts allow the easy and exact combination of the parts. Each goal snap point matches one origin snap point. The generic definition of the snap behavior and rules of construction parts allows us to easily create different kinds of construction kit behavior.
Although the special snapping mechanism greatly simplifies the construction task, navigation in the already constructed scene is still a very time consuming task even for trained users: Before we can attach a new part to our model for example by dragging the part with the mouse towards the corresponding snap points the user must position her/himself correctly with respect to the part she/he is attaching the new one to.
The same navigation tasks must be performed for inspecting a certain part of a model for investigating how a model is build up from single parts. To free the users hands we aimed for an interaction metaphor that is simply steered by a mouse but can also be driven by speech. Before presenting our new approach we briefly summarize previous work and discuss its deficiencies.
The early and recent publications are generally concerned with the problem of immersive navigation [WaOs90][BBCH00][StCP95]; a few mention the desktop problem [MaCR90].
In an early publication Ware and Osborne analyze the nature of the problem [WaOs90]:

"The task of placing a viewpoint in a virtual environment has basically six degrees of freedom – three for positional placement and three for angular placement …".

Ware and Osborne investigate three different interaction metaphors:
Eyeball in hand: the user navigates the scene by moving a special hand held device as a virtual camera through the scene;
Scene in hand: This metaphor interprets the movement of the interaction device as a movement of the entire scene;
Flying vehicle control: the interaction device is considered a flying vehicle and the image displayed corresponds to the image the user would see if he or she were placed inside the vehicle.
They tested the acceptance of the metaphors on three typical environments and found that the flying vehicle control is least suited for moving around a closed object and best suited when navigating inside an object, such as a maze. The scene in hand metaphor is reported least suited for the navigation through a maze and best suited for the movement around a closed object.
Variations of the flying vehicle metaphor are the car driving and the walking or locomotion metaphor the latter introduced by Brooks et al. [Broo86], cited in [HPCK94]. In 1994 Pausch et al. [HPCK94] introduce a new interaction technique: the ray cast metaphor. Here a ray is cast at an object intended for selection or as a navigation goal.
This theoretical background applies for mouse-based desktop VR applications, such as realized in the standard browser CosmoPlayer [Cosm99]. Here a pragmatic solution to the navigation problem. is provided; constrained to fulfill the ISO/IEC specification [ISO-97], which requires navigation modes for:
Walking: The camera moves through the scene bound to a certain height;
Examine: The motion of the input device is mapped to a motion of the scene, not the camera.
Flying: The motion of the input device is mapped to a motion of the camera.
The implementation uses a dashboard metaphor offering two sets of controls, one for walking and flying and a second one for examining the scene, displayed in figure 1a and 1b respectively. The walk and fly mode, is switched by selecting a gravity on / off button.
Change-controls chooses between the examination and the move mode. The latter presents a combination of the eyeball in hand metaphor (go) and the flying vehicle metaphor (tilt, go, slide); while the first mode can be interpreted as a derivative of the scene in hand metaphor (rotate, pan), where the scene can be rotated and translated. Seek (seek), allows selecting an object and then moving closer; similar to the ray cast metaphor, presented in [HPCK94].
The metaphors are complete in the sense of allowing for all desired navigational motions. However, they are complicated to handle for novice users and even trained persons may find a special navigation task difficult to accomplish. The software documentation explicitly mentions that one:

"… could easily become disoriented by switching from one set of controls to another. …" [Cosm99].

This experience is not limited to a special browser, but found in all similar software today. It might be the reason for a different concept that is also supported: the navigation by viewpoints. A viewpoint is described by a *position*, the *look at direction* and an *angle describing the field of view*. Browsers allow navigating a scene along a number of predefined viewpoints. The browser interface displays a list of available viewpoints and provides a navigation control for "forward" and "backward". The browser supports a direct linear interpolation between these viewpoints. The scene designer must however take care that the path connecting two viewpoints does not intersect any objects.

Free navigation interface
of a standard browser (walk/fly)
Figure 1a

Free navigation interface of a standard browser
(scene manipulation)
Figure 1b

If they intersect an object, the browser will stop at the object, if collision detection is turned on, and move through the object, if it is turned off. The browser does not perform any other than the direct path computation, and the design of the path is in the responsibility of the application designer. The navigation is easy and informative, if the path is well defined. But here a free movement in the scene is not supported, i.e. when the visitor wishes to see something else, than the guided tour path, he or she faces the problem of free navigation using the interaction metaphors described before. The problems reported by Ware and Osborne for immersive environments [WaOs90], are similar for desktop environments [MaCR90], even though here, the visitor of the scene has a better overview being outside the environment. The object-centered

navigation metaphor tackles all these issues and allows directly accessing a desired object in an intuitive fashion.

2. THE CONCEPT OF OBJECT-CENTERED VIEWPOINTS

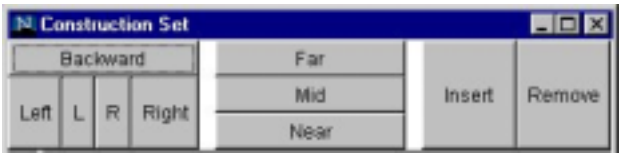
The concept of object-centered viewpoints provides a new metaphor, which allows the user to select the object of interest, directly go for it and then use relative positioning to move around the object. All viewpoints are dynamically computed during the navigation, they need not be predefined.
Each object in the scene is associated with a set of object-centered viewpoints for an orientation or aspect, i.e. a view from:

- Back
- Front
- Left
- Right
- Below
- Above

And for each of these orientations or aspects a level of distance to the object can be specified as:

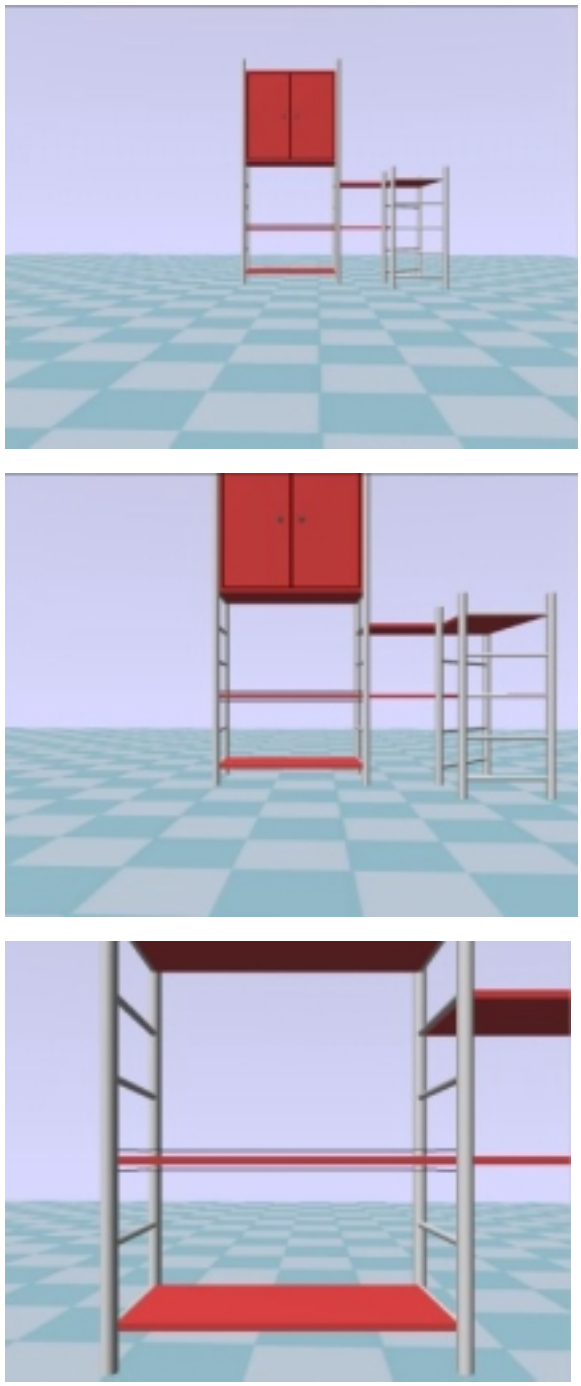
- Near
- Far
- Intermediate

In this way each object provides a total of eighteen object-centered viewpoints.



Navigation controls for object-centered navigation in a construction kit
Figure 2

The object-centered navigation keeps track of the current front position, i.e. which direction the visitor is looking at the scene. If the user moves on to the next object, by selecting it, the current front position is kept, such that the user does see the newly selected object from the object-centered aspect of orientation (back, front, left or right, above or below) that is closest to the current front position.
For navigation of the scene, the visitor can easily go to the position he or she wishes, by directly addressing the object. Figure 2 shows the navigation controls for object-centered navigation. Figure 3 shows an object from three different object centered viewpoint positions.



Object-centered viewpoints
Figure 3

A task such as putting an object somewhere or using a tool, etc. is much easier, when the system supports navigation directly to the point of interest. When moving left or right, the system keeps track of the current look at position as the front position and defines the left, right, and back orientation relative to the current front view, using a counterclockwise numbering of the directions.
For a more fine-grained navigation around an object, in addition to the basic orientations back, front, left, right, which are at the corresponding sides of a

bounding box surrounding the object, intermediate steps are also provided, such as:

- Turn “a little” left
- Turn “a little” right

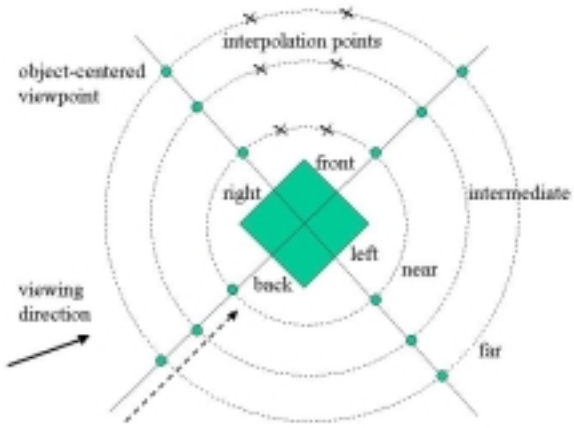
These small step widths are useful, when the visitor of the scene is to complete certain tasks with the objects in an interactive environment. This allows us to easily detect, where the relative front positions of the surrounding objects lie:

- Front -> 0
- Right -> 1
- Back -> 2
- Left -> 3

The currently viewed side of the object is always defined as the relative front face. When navigating in the environment, the user has the option to select a new object for moving there, or examining current object. If the current object is examined more closely, the user can go round the object using the left, right, front and back navigation or the colloquial “turn a little” left / right navigation. Moreover the distance can be varied switching from near to far or intermediate in any desired sequence. Every object may have a predefined front, back, left and right position that can be used for absolute positioning, this is however optional.
When moving from one viewpoint to the next, we use a simple interpolation and camera animation that smoothly takes the visitor to the new look-at position and beforehand we use a simple visibility test.

3. INTERPOLATION BETWEEN VIEW-POINTS AND PROBLEM OF VISIBILITY

In order to allow a smooth motion among the viewpoints, an interpolation must be performed for the animation of the camera. It should move along a natural path; objects that intersect the line between source and target viewpoint must be properly detected and traveled round.
A trivial method for avoiding visibility problems is not to perform an animation. This method is often referred to as teleporting. It is possible with object-centered viewpoints, but generally not satisfactory for the visitor, since it does not explore the full experience of 3D space.
The first step in defining a camera animation is determining the camera path in the scene. The second step is the display of the animation. An optimal algorithm should detect the shortest viable path between the two viewpoints and guide the visitor to the new location.



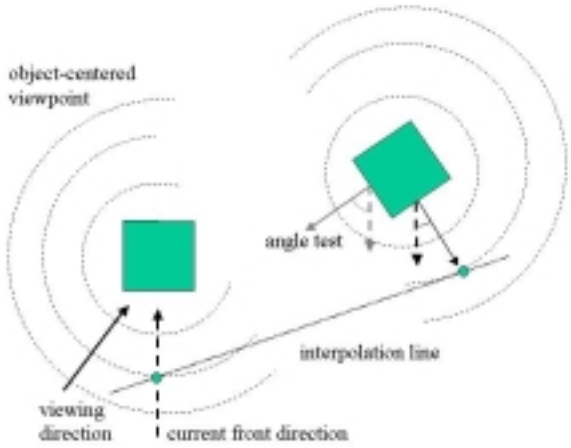
Object-centered viewpoint model
Figure 4

Before computing the camera animation it must be checked that the target viewpoint is not inside an object. For adjacent objects some of the near-viewpoints may intersect a neighboring object, or lie inside it. In these cases the corresponding viewpoint must be locked for navigation.
The object-centered navigation allows moving around an object (left, right, a little left, a little right) and a change of the focused object, i.e. the navigation to a new object. Both navigation tasks take the visitor to a different object-centered viewpoint.

Navigation around an object For the navigation around an object, the object-centered viewpoints: left, right, front, and back are possible targets, and the directions: “a little” left and “a little” right, too. Turning right or left uses an angle of 90 degrees, turning back of 180 degrees. To move “a little” left / right uses an angle of 30 degrees. The object-centered viewpoint positions lie for each of the distances near, intermediate, and far on a circular line with different radius around the object’s bounding box. The interpolation is done in a linear fashion along four interpolation points, the start viewpoint, the end viewpoint and two equidistant intermediate positions, facing the center of the bounding box, see figure 4. If no collision is detected along the interpolation line, which is also the camera path, the path is used for the camera animation. If a collision is detected, the path is computed for the next level of distance of the two viewpoints. I.e. if the current level of distance is “intermediate” the next tested level is the “far” level.
The animation contains then five interpolation points, first a zoom out to the next level of distance viewpoint, then the animation as described above and then a zoom in on the original level of distance for the new viewpoint. If going up the levels of distance does not work either, going down the levels of distance is tried, and if this does not work, the next level of hierarchy of objects is used for the

computation of the viewpoint animation path. The smallest level of hierarchy in the scene structure is a simple atomic object and the highest level of hierarchy corresponds to the entire environment. This method is useful, when object hierarchies are available.
The intersection test is repeated until no intersection is found. The algorithm works well for scenarios, where for each pair of objects, such a condition can be met. Generally this is possible for sparsely populated scenes such as e.g. the interior of a room. Densely populated scenes with many intersecting objects or scenes that form a maze are difficult or even impossible to handle without intersection. The current implementation does ignore an intersection, when it is found on the highest level of distance. Here future implementation must provide an algorithm that allows smoothly moving around objects. For the test scenarios, shown in figure 3, 6, 7 and 8, the algorithm however worked fine, and allowed us to perform the user tests on the navigational ease of use of the object-centered viewpoints and the comparison to the free navigation metaphor.

Navigation to a new object Here the user selects the target of his or her motion and then is taken there. To compute the path for the corresponding camera animation, we use a simple algorithm. First the front object-centered viewpoint of the new object is determined. The angle formed by the normal vector of the object-centered viewpoint and the current front direction is computed and the viewpoint with the smallest angle is selected as the appropriate new front direction, the distance level is adopted corresponding to the current distance level, see figure 5.
For the new object-centered viewpoint and the current object-centered viewpoint, a visibility test is performed. If no other object lies on the interpolation line, the camera path is set and the animation is performed. If an occluding object intersects the camera path, the other distance levels are checked for occlusion. First the up direction, then the down direction is considered, according to the same schema used for the navigation around an object. If no occlusion-free path is found, the next hierarchy level is considered up to the first level of hierarchy that comprises both objects. Then the interpolation strategy changes to the one described before for the navigation around an object, since both objects belong to the same hierarchy group, and are contained in an identical bounding box. Again the algorithm works fine for sparsely populated scenes. In arbitrary scenes it cannot be granted that a valid animation path is found, nor can a decision be made on whether a valid camera animation is possible or not.



Object-centered navigation to a new object.
Figure 5

This restriction must be considered, when the object centered animation is intended for use in an extended fashion, i.e. “blindly”. In the experiments we did, the scene was visible, i.e. the user was navigating among the objects he or she could see. And of course, if there is a line or “curve” of sight for the user between two objects, a camera path does exist.
In “blind” environments, i.e. environments, where the user gets the option to navigate to locations he or she does not immediately see, a different method must be applied for the visibility testing, and an optimal reliable algorithm must be used.
Here our focus was in first place on the evaluation of the object-centered navigation and on the comparison with the free navigation, concerning the ease of use. Therefore we focused on sparse environments, which are sufficient for the test purpose, since according to [WaOs90] even for very basic tasks, such as the navigation around an object or moving between different signs, a free navigation is difficult.
In order to allow for a navigation to all reachable viewpoints, even if no valid camera path is detected, the algorithm in the current implementation ignores the topmost found occlusion and takes the user in an animation to the new viewpoint, regardless of the collision.
The great advantage of object-centered navigation over the six degrees of freedom interfaces is obvious. It can still be improved by adding a new option: speech interaction. This feature frees the users hands for other tasks while exploring the scene.

4. NAVIGATION BY SPEECH INTERACTION

Speech interaction and other natural interaction methods are analyzed in an early publication by Nielson, who surveys non-command user interfaces and comes to the conclusion that even non-command based user interfaces have to face the issue that:

"… there are tasks that are more naturally accomplished by explicit commands …" [Niel93].

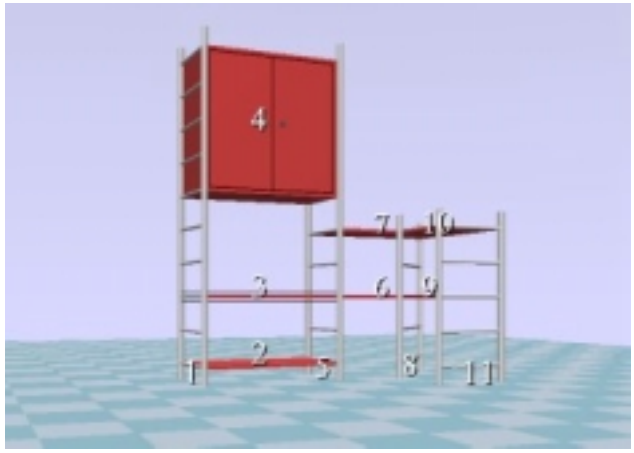
The aim of non-command based interfaces is that of a more natural interaction with the computer. Nielson cites the example of a card table system, where the user is playing a game. An other examples of non command-based interfaces is eye tracking, which Nielson however considers "esoteric", and difficult to realize since:

"… Users do not have full control over their eye movement; and the eyes run all the time, even when the user does not intend to have the computer do anything. …"

The advantage of eye tracking lies in the option freeing the user's hands. This advantage can also be achieved using speech interaction [JKLS00]. Speech recognition is still a difficult and unsolved task for the general case, but systems trained on recognizing special commands in a well-defined context tend to work well.
Issuing spoken navigation commands allows the user to navigate via voice interaction through the scene and keep performing a different task with his or her hands. This is especially useful in applications such as virtual construction manuals or construction demonstrations using 3D models and animation. The user may navigate the manual by voice interaction while assembling or fitting an object, e.g. a piece of furniture.
With the voice interaction he or she can easily move the focus to a special detail of interest, all while continuing the manual work. Voice interaction based navigation is also useful in a large number of other scenarios and applications [JKLS00].
For object-centered navigation the voice commands directly correspond to the interaction commands of the navigation interface depicted in figure 2.
I.e. the speech navigation interface provides the following set of commands for navigation around an object:

- Turn to the Front (say: "front")
- Turn to the Back (say: "back")
- Turn Left (say: "left")
- Turn Right (say: "right")

For navigation to a specific object, the user would click the object in the non speech-based object-centered navigation, described in section 2. Here, each object is given a label, containing a number for identification. To move to an object, the user just says its number. The animation of the camera path is done in the same fashion as for the non voice-based navigation, described in section 3.



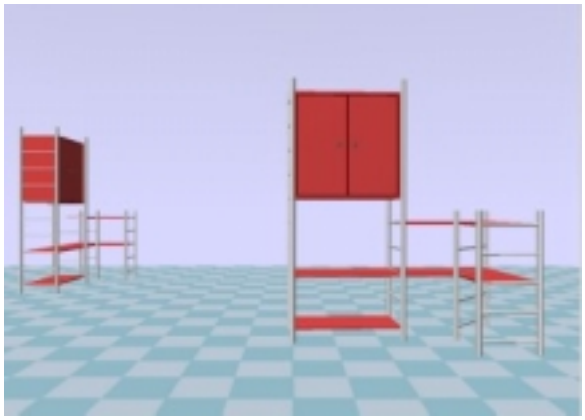
Object-centered speech navigation,
identification by numbers
Figure 6

Technically the voice interaction we used is based on a Java wrapper to a standard speech engine of Microsoft for PC platforms.
We found that the speech interaction works fine and has the same positive effects as the object-centered navigation without speech interaction. There is a slight difficulty in the recognition of the words, if the person using the speech engine does not train the system beforehand. Moreover, the speech engine often recognizes command words from discussions taking place close to the microphone, although no command was spoken. Figure 6 shows a screenshot of the object numbering for easy identification of navigation targets.
In this work, we have implemented an object-centered navigation for object-level navigation, i.e. without considering more than two levels of hierarchy. For a general ease of use, a navigation system could provide more than just object level navigation. It may be useful to select groups of objects and then focus on the group as a whole instead of a single object.

5. PROBLEM OF HIERARCHIES

When exploring an environment one typically has a certain idea of what he or she wishes to see next. In an unknown building, a museum for instance, one may wish to go to a certain exhibit or tableau directly. In these cases the object-level navigation is useful. The tableau or exhibit is an object-level target. For accessing intermediate camera positions between objects or for focusing on more than one object, a different method is required.
Approaches presented in related publications often rely on artificial intelligence based solutions, without managing the complexity of the problem [JKLS00]. The object-centered navigation model can be extended easily using hierarchies to encompass these problems of colloquial motion specification such as expressed by the instruction: move "close to" a

group of objects. Hierarchies allow recognizing objects in groups of interest, such as e.g. a table in a room surrounded by several chairs.
The user may wish to first step close to the group, and then in a second step from a closer position decide again, which chair to choose.

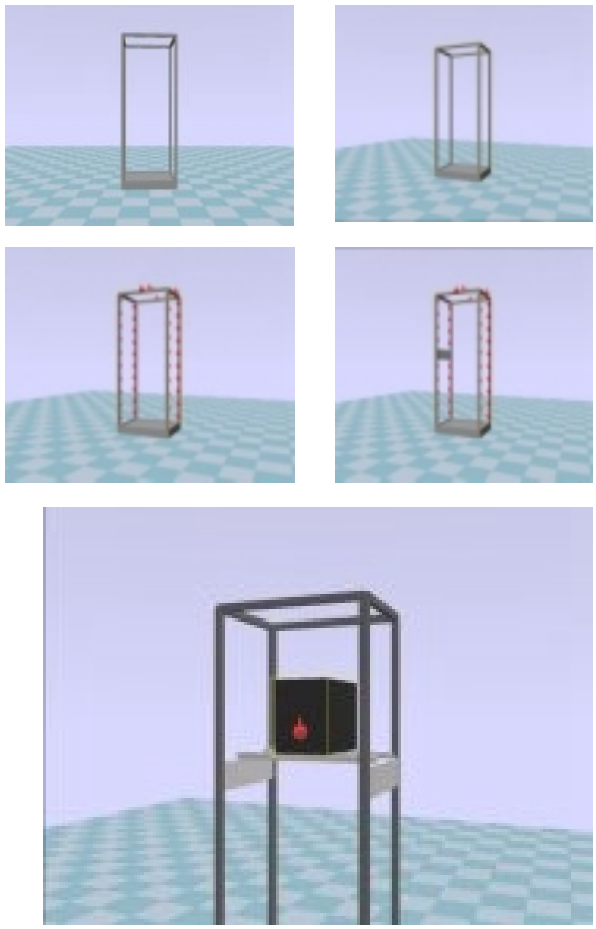


Simple Object Hierarchy
Figure 7

In some of these cases the navigation can be handled using a predefined hierarchical structure of the model. Streets, buildings, and rooms form such a hierarchy, e.g. in geographic information systems. In some cases a predefined hierarchy may be inappropriate. As an example consider again an entrance hall of a large unknown building, where the one wants to visit a corner showing an advertising pillar, a door and an information desk. From there it is easy to decide which of the three objects may be of further interest.
In such a case, a predefined hierarchy will most probably fail offering a navigation target that comprises such an individual group of three objects, since they do not share any logical connection that would justify placing them in a joint hierarchical wrapper. A more flexible method is needed. We suggest a method that allows the user to select a certain number of objects at once. In a test algorithm, we allow the user to select up to three objects as navigation targets. The algorithm computes the bounding box that comprises all three objects and generates the object-centered viewpoints for the joint bounding box. Figure 7 shows an environment, for which a simple two level hierarchy is implemented. Hierarchies are also helpful, when using large and complex models of single real world objects, such as e.g. an aircraft or a car. Buildings and geographic information systems, too, are typical hierarchical structures.

6. RESULTS

We have tested and evaluated the navigation with object-centered viewpoints in a construction kit scenario, which lets the user build new furniture and other models [Ding00]. We have used the same scenario for the test, in one version it contained the object-centered navigation, in the other version it only supported the free navigation controls. The scenario is depicted in figure 8.



Test environment for construction and navigation
Figure 8

The persons testing the applications were instructed to construct twice the same furniture model using the free navigation model first and the object-centered navigation second; they were asked for a comparison concerning the ease of use for both interfaces.
The test persons comprised novice users and experienced game players who were already habituated to free navigation controls.
Both groups did the construction in considerably less time using the object-centered navigation. While the novice users expectedly performed very badly with the free navigation as compared to the experienced users, they took about the same time for construction using the object-centered navigation.
Both groups unanimously preferred the object-centered navigation for the construction task. The reasons stated were: it allows subjectively traveling faster; it allows directly getting to a desired position; it avoids the many false and unintended motions. It was also considered to be easier to handle and understand.

7. CONCLUSION

In this paper we have introduced a new object-centered concept for navigation in virtual construction scenarios that provides a set of dynamically computed viewpoints for each object and a simple algorithm for the camera animation for motion in the scene.

We have shown that the discretized set of object-centered viewpoints serves well as an interface for speech based navigation and allows us to perform imprecise motions such as “a little” left/right and “close to”, to name some. Both, the object-centered navigation and the object-centered navigation with speech interaction have been tested in different virtual construction scenarios. Although the speech recognition sometimes made erroneous interpretations of the spoken commands, the test persons approved the concept.

It was generally stated that the object-centered navigation was very helpful for the virtual construction scenario.

Extensions in the path finding concept will allow applying the object centered navigation in arbitrary “blind” virtual reality environments.

8. FUTURE WORK

In addition to the here tested virtual construction scenarios, the concept of object-centered viewpoints is applicable to a variety of other navigation problems in virtual reality.

For the use of object-centered navigation in complex environments, future work must consider an algorithm for global path finding and a refined hierarchical concept. Additions should allow performing complex navigation path calculations in unknown environments such as e.g. large buildings. Moreover, it provides the basis for an integration of the navigation system with tagged information systems, such as e.g. virtual product catalogs or virtual location manuals.

REFERENCES

[BBCH00] Bowman, D.; Billinghurst, M.; Cugini, J.R., Hinckley, K.; et. al.: 20th Century 3D User Interface Bibliography: an annotated bibliography on 3D user interfaces and interaction: http://www.mic.atr.co.jp/~poup/3dui/3duibib.htm

[Broo86] Brooks, F. P. Jr.: Walkthrough, a dynamic graphics system for simulating virtual buildings; Proceedings of the ACM Workshop on Interactive 3D Graphics, 1986, pp. 9-21.

[Cosm99] CosmoPlayer2.1 online documentation, http://www.cai.com/cosmo/.

[Ding00] Dingel, Th.: Authoring von interaktiven Verhaltens-komponenten für 3D-Modelle im WWW; Diploma thesis, FH Darmstadt, March 2000.

[HPCK94] Hinckley, K.; Pausch, R.; Goble, J.C.; Kassell, N.F.: A survey of design issues in spatial input; Proceedings of the ACM symposium on User interface software and technology, 1994, Pages 213-222.

[ISO97] ISO/IEC 14772-1:1997: The Virtual Reality Modeling Language (VRML97); http://www.web3d.org/.

[JKLS00] Jung, B.; Kopp, S.; Latoschik, M.E.; Sowa, T.; Wachsmuth, I.: Virtuelles Konstruieren mit Gestik und Sprache; Künstliche Intelligenz, 2/00, pp. 5-11.

[MaCR90] Mackinlay, J.D.; Card, S.K.; Robertson, G.G.: Rapid Controlled Movement Through a Virtual 3D Workspace; Proceedings of SIGGRAPH’90, pp. 171-176. www.hitl.washington.edu/publications/.

[StCP95] Stoakley, Richard; Conway, Matthew J.; Pausch Randy: Virtual Reality on a WIM: Interactive Worlds in Miniature; in Proceedings of CHI’95.

[WaOs90] Ware, Colin; Osborne, Steven: Exploration and Virtual Camera Control in Virtual Three Dimensional Environments; Proceedings of the 1990 symposium on Interactive 3D graphics, 1990, pp. 175-183.