Improving Navigation Efficiency with Artificial Force Field

Tsai-Yen Li and Hsu-Chi Chou

Computer Science Department, National Chengchi University
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11623, ROC

e-mail: {li, s8607}@cs.nccu.edu.tw   Tel: 886-2-29387170

Abstract

Evolution of 3D graphics acceleration hardware has enabled novel user interface designs for many new applications running on a regular desktop personal computer. However, navigating in an architectural environment remains difficult for a novice user because of the limited view and low-level control that the current interface provides. In this paper, we propose to use an artificial force field to help a user walkthrough an architectural environment efficiently and naturally. For a given environment, a repulsive force field is computed off-line in accordance with the obstacles in the environment. An artificial force generated from this field is applied to the user viewpoint in order to reduce the chance of getting stuck by the obstacles. This force is dynamically modified at run time according to user behaviors such that the generated motions are compliant with user intention. Experiments show that artificial force field is an effective method for improving navigation efficiency. In addition, we have also integrated the force field method with a path planner proposed in our previous work. We have found that the force field method is complementary to the path-planning method, and the merits of both methods can be combined to further improve navigation efficiency.

Keywords: Intelligent User Interface, Architectural Walkthrough, Virtual Reality, Artificial Force Field, Navigation Control.

1. Introduction

Architectural walkthrough is a form of user interface design that adopts Virtual Reality (VR) technologies (especially 3D graphics) to help a user navigate through realistic architectural virtual environment. This form of interface has become popular due to the rapid development of Virtual Reality Modeling Language (VRML) (or Web3D)[17] and 3D graphics acceleration hardware. This development also has enabled many new applications that were traditionally only possible on expensive workstations. For example, in addition to exploring a real architecture building, many popular games such as DOOM or QAKE also adopt similar first-person view of control. In other words, experiencing 3D graphics on a desktop computer is now prevalent. However, since a virtual environment typically uses 3D display, it remains a challenge for a novice user equipped with a 2D mouse to perform efficient navigation.

A typical 3D browser supports several user control modes, such as WALK, FLY, EXAMINE, etc. The WALK mode corresponds to the architecture walkthrough interface, which is the focus of this paper. Most browsers also allow a user to have the option of turning on the collision detection function. With this function turned on, the system will prevent a user's viewpoint from penetrating obstacles in the environment and therefore increase the degrees of realism. However, a novice user in such an interface often runs into a situation of getting stuck at a corner of an unfamiliar scene. A user usually needs to perform several maneuvers before making any progress. It is a frustrating experience for a novice user especially when the display frame rate is not high enough for precise control.

In addition to the problem of low frame rates, we think the level of control that a user needs to provide is too low compared to the experience in our daily life. For example, we usually only raise our intention through high-level goal specification when we walk in the real life and the body locomotion will be automatically generated to satisfy the intention. There exist systems that can perform auto-navigation service for its user. However, a user enjoying the experience of architectural walkthrough usually does not want to give up the control Therefore, it remains a challenging task to design an interface that considers both navigation experience and ease of control. In our previous work, we have proposed a method to incorporate path-planning techniques into the control loop of user navigation. The system voluntarily computes a collision-free path to assist navigation when the user runs into a difficult situation.

In this paper, we propose another novel approach of using an artificial force field in the control loop of 3D interactions to reduce the chance that a viewpoint collides with environmental obstacles. A repulsive force field is constructed in a preprocessing step for the freespace of a given virtual environment to resist a user from getting too close to the obstacles. While influencing user control with this force field to avoid collisions, we also have to take into account user intentions. According to user's motion behavior (veloc-

ity and orientation), we dynamically adjust the force field to reflect user intention such that a user does not feel awkward under the influence of such a force field.

We organize the remaining of the paper as follows. We will review some related researches in intelligent user interface design in the next section. We will then present the idea of force field and how we dynamically modify it to reflect user intention in Section 3. We will then show the details of our implementation in Section 4, and the experimental settings and results, in Section 5. Finally, we will conclude our work in the last section.

2. Related Work

The work pertaining to our research is traditionally classified as the category of computer-human interaction, especially in the field of intelligent user interface design. One can also find work in Artificial Intelligence and Robotics that aims to adding such intelligence to man-machine interface. We will review 3D user interface design first and then narrow down to issue of intelligent 3D user interface.

2.1. 3D computer-human interface design

Many researches have been undertaken to invent efficient ways to communicate with a computer and to evaluate the effectiveness of these interfaces. Among these interfaces, being capable of interacting with virtual 3D environments has been considered as a design trend for future computer-human interfaces. The VR-types of interfaces such as Head Mounted Display (HMD), 3D tracking devices, data gloves, force feedback joysticks, haptic devices, etc, are all good examples of such interfaces. New metaphors such as eyeball in hand, and flying vehicle in hand have also been proposed and tested.[1] It is reported that most users like the idea of eyeball-in-hand metaphor in the context of virtual space exploration. However, the great challenge comes when we are asked to manipulate a 3D virtual scene only with a regular 2D mouse on a desktop computer. Some work has been carried out to design intuitive interfaces for controlling 3D rotations with 2D devices.[4][14]

Most of these interface designs use the direct manipulation metaphor that is shown to be more comprehensible, predictable, and controllable than the delegation types of intelligent user interfaces in several application domains. However, it is still under debates which metaphor is more effective in general.[15] We think there will not be a clear-cut answer to this question. Instead, effectiveness would greatly depend on the types of applications, users, and tasks at hand. For example, some people may prefer to sit back and take a guided tour when visiting a new environment while other adventurous people may prefer to have a full navigation control.

2.2. Intelligent user interface

Although many intelligent user interfaces have been proposed in the literature, most of them are not for 3D manipulation.[12][13] Exceptions include using motion-planning techniques to provide task-level controls. For example, Drucker and Zeltzer [2] argue that a task-level viewpoint control is crucial for exploring virtual scenes such as virtual museums since the users should be allowed to concentrate on scene viewing instead of be distracted by low-level navigation controls. Li, et al.[9] also proposed an auto-navigation system capable of generating customized guided tours based on high-level user inputs. Kuffner [6][7] also utilizes fast path-planning techniques to assist real-time animations. Other work also suggests using vector fields [3] to guide animation. However, most of these approaches use geometric reasoning techniques as a tool for control delegation. They use a third-person view to specify the desired tasks, which is very different from the first-person view commonly used in the direct manipulation metaphor.

Force field methods, such as potential field methods, were originally proposed to solve path-planning problems for robotic applications.[5] Attractive forces were used in addition to repulsive forces from environmental obstacles. The idea of using a force field to improve user navigation has been independently developed in [16]. However, they assume that a 3D pointing device, such as a 3D mouse, is available to the user while we assume that a user is only equipped with a regular 2D mouse. In addition, the ways that we compute the force field, detect collisions, and overcome motion oscillations are different. Furthermore, we evaluate and compare the effects of the force field method, the path-planning method, and the combination of both methods.

In [10], we have proposed an intelligent user interface that incorporates path-planning techniques into the control loop to help a user move around an obstacle. A fast path planner based on randomized roadmap is called to compute a collision-free path whenever the viewpoint would collide with environmental obstacles. Our experimental results show that the navigation efficiency for a given scene can be improved by as much as 73 percent. In [11], we further extend this method to consider large virtual environments.

3. The Force-Field Approach

We assume that under the walkthrough mode the virtual environment can be represented as a 2D layout map such as the one shown in Figure 1. The environment is a bounded workspace comprised of polygonal obstacles. The portion of workspace outside the obstacles is called *freespace*. The basic idea of the force-field approach is that obstacles in the workspace generate repulsive forces to the viewpoint in the freespace such that it becomes more difficult to move the viewpoint close to the obstacles. For example, in
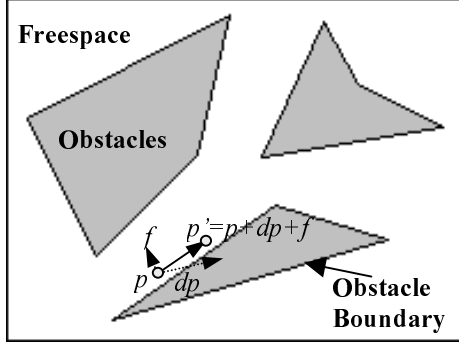
Figure 1: obstacles in a bounded workspace

Figure 2: repulsive forces generated from obstacle boundaries

Figure 3: resultant forces from several obstacle boundary points

Figure 1, the displacement of the viewpoint configuration dp is offset by a force vector f to produce the next viewpoint configuration, p'. With this approach we hope that the chance of getting stuck by the environmental obstacles can be reduced, and therefore the overall navigation efficiency can be improved. In this section, we will first describe the desirable features of a force field for our application and then define a basic force field according to physical laws. We will then refine the force field to overcome some undesirable effects as a result of composing the forces from multiple obstacles. Finally, we will describe how we modify the force field dynamically at run-time according to user navigation behaviors such as navigation speed, moving direction, and navigation history.

3.1. Computing a basic force field

The force field proposed in this paper is an artificial force field designed specifically to assist user navigation. Nevertheless, the computation of this force field is motivated by physical laws. The Newton's law of gravitation suggests that two objects are attracted by a force whose magnitude is proportional to the square inverse of the distance between the two objects. In addition, the force is proportional to their masses. The ideal force field in application is very similar to this law expect that the force is repulsive instead of attractive. In addition, the repulsive force is from the boundary of obstacles instead of from the interior of the obstacles. Therefore, the mass or volume of an obstacle is not taken into account when we compute the force field. This field is a static vector field in the freespace, and therefore, we can compute the force in a preprocessing step for each point in the freespace up to a resolution. This force is the basis for influencing the viewpoint motion at run time. Usually it needs to be scaled by a constant and adjusted dynamically according to user behaviors. For example, when the viewpoint is not moving, no forces would be applied.

Assume that there exists a point $\bar{p} = (x, y)$ in the freespace and the closest boundary point of an obstacle from this point is $\bar{e} = (x', y')$. Let $\bar{v} = \bar{p} - \bar{e}$. Then, we define the repulsive force for $\bar{p}$ as:
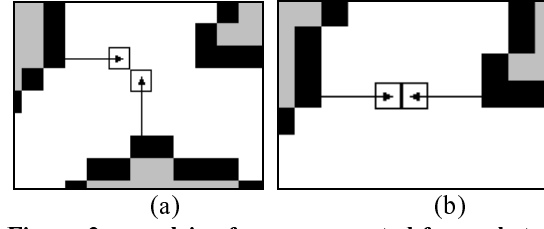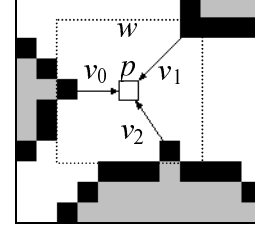
$$\bar{F}_p = \frac{1}{|\bar{v}|^2} \times \frac{\bar{v}}{|\bar{v}|}, \tag{1}$$

where $\bar{v}/|\bar{v}|$ is the unit vector pointing outward from obstacle boundary.

The computation of this force is straightforward except for determining the closest obstacle boundary points such as the examples shown in Figure 2(a). We assume that the workspace is represented as a bitmap, where filled cells represent obstacle regions. For such a bitmap this information can be easily determined by a wave-propagation algorithm that incrementally propagates wavefronts from obstacle boundaries.[8] The complexity is linear to the size of freespace.

3.2. Improving the force field

Although the closest boundary point is supposed to generate the largest repulsive force, using the force from a single point only create undesirable problems. For example, the forces applied to the two neighboring points in Figure 2(b) have opposite directions. Consequently, large discontinuities in the force field may result in motion oscillations at axial regions of freespace. To fix this problem and create a smooth force field, we have to consider more than the closest point.

Instead of considering the closest point only, we consider all the points in a window centered at a given point $\bar{p} = (x, y)$ in the freespace. Assume that there are k points of obstacle boundary in the window of size $s \times s$ and their coordinates are $\bar{e}_0 = (x'_0, y'_0)$, $\bar{e}_1 = (x'_1, y'_1)$, ..., $\bar{e}_{k-1} = (x'_{k-1}, y'_{k-1})$. Let $\bar{v}_i = \bar{p} - \bar{e}_i$, then the force $\bar{F}_p$ acting on $\bar{p}$ is defined as:

$$\bar{F}_p = \sum_{i=0}^{k-1} \frac{1}{|\bar{v}_i|^2} \times \frac{\bar{v}_i}{|\bar{v}_i|}. \tag{2}$$

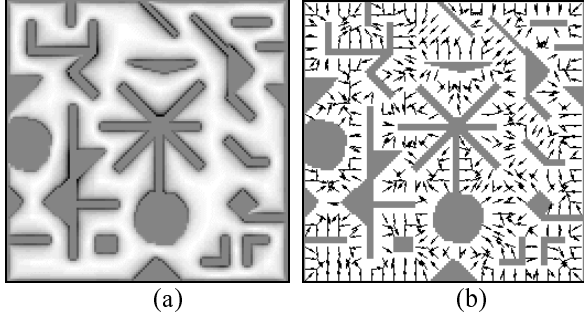For example, for the point p in Figure 3, there are

Figure 4: force magnitudes (a) and directions (b) in an artificial force field

Figure 5: top view of the maze environment

Figure 6: a snapshot of the VRML browser

three boundary points in w (7x7) applying repulsive forces (v_1, v_2, and v_3) to p. The resultant force is computed as $v_1+v_2,+v_3$. An example force field computed with this method is shown in Figure 4(a) and 4(b). In Figure 4(a), the magnitude of the force field is depicted as a gray-scale image (the darker, the stronger the force) while in Figure 4(b), the directions of the force vectors are depicted as arrows. The oscillation phenomenon is avoided since for the forces from the opposite sites of a point along the axial direction cancelled each other out. Consequently, the force field becomes a smooth vector field pushing the viewpoint outward from obstacle boundaries. This force field is a static field that can be computed in a preprocessing step. The size of the influence window, w, can be defined as large as the size of the workspace. However, in our implementation a reasonable size (such as 21x21) is usually good enough for capturing the major repulsive forces.

3.3. Dynamically adjusting the force field

The main advantage of the direct manipulation metaphor is that it allows the user to have the feeling of having the full control of the viewpoint motion. Therefore, it is crucial to consider the naturalness of the viewpoint motion after applying such an artificial force field. Any modifications to the viewpoint motion should not violate the user's intention. For example, when the user does not drag the mouse, the viewpoint should not move. When the user changes the magnitude of input command, say speeding up, the resulting viewpoint must move relatively faster. Even if there are temporary violations of user intention, the user should be able to adjust the input command to reflect his intention interactively.

In order to respect a user's input, we adopt a dynamic update scheme to modify the strength of the force field at run time. This adaptive scheme is based on a user's navigation behavior, such as speed, orientation, and history. First, we think the force should be proportional to the speed of the viewpoint. Thus, no forces should be applied to a static viewpoint. Second, the applied force should reflect a user's intention through the moving direction and navigation history. For example, when a user intends to move toward obstacle boundary, we should reduce the magnitude of
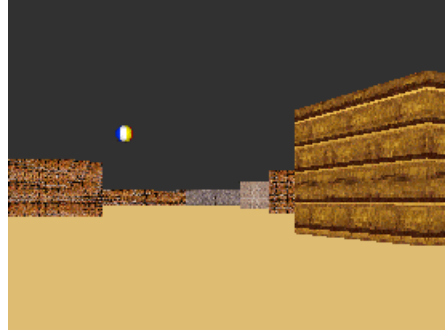
the force gradually until the viewpoint reaches obstacle boundary. If the viewpoint moves outward from the obstacle boundary, we should assist the user in increasing the strength of the repulsive force.

The adjusted force is computed with the following formula:

$$\vec{F}_p{}' = \mu_v \times \mu_d \times \vec{F}_p, \qquad (3)$$

where $\vec{F}_p$ is the pre-computed force from eq.(1). μ_v and μ_d are two coefficients that modify the magnitude of the force. The speed coefficient μ_v is defined as $\mu_v = \kappa v$, where v is the viewpoint's linear speed and κ is a constant scalar. The direction coefficient μ_d is defined as $\mu_d(t_n) = \eta \times \mu_d(t_{n-1})$, where $\mu_d(t_{n-1})$ is the direction coefficient in the previous time step and η is a scaling factor that could be larger than or smaller than one depending on whether dot product of the viewpoint's movement vector and the applied force vector is greater than 0 (same direction) or less than 0 (opposite direction). In other words, when the force is applied along or against the direction of the viewpoint's motion, its magnitude will be enlarged or reduced, respectively. With this update scheme, the user is capable of moving close to an obstacle boundary if he/she intends to do so since η becomes smaller and smaller. When the user moves away from the obstacle, the magnitude of η is gradually recovered. In order to prevent η from becoming too larger, we also set a reasonable upper bound for the maximal value of η.

4

4. Implementation

4.1. Connecting to a VRML browser

In order to make the research result be more portable, we choose to modify the open source VRML browser implemented based on the Java3D SDK library.[18] This SDK and the VRML browser are all available for FTP on the public domain.[18] The force field computation programs, implemented in Java, read in a 2D-layout map of obstacles as well as a 3D VRML file. The top view of a typical maze-like scenario for our experiment is depicted in Figure 5 and a snapshot of the VRML browser interface is shown in Figure 6. For collision-detection purpose, we decomposed the workspace into a 128x128 uniform grid of cells. In these programs, we have modified the routine for processing mouse events and the routine for updating the next viewpoint configuration. In a typical walkthrough interface, a user issues his command by dragging a vector that is decomposed into horizontal and vertical components, representing the rotational and translation velocities, respectively. These velocities correspond to a transformation that transforms the viewpoint to its new configuration in the next time frame. This transformation is modified according to a vector proportional to the artificial force proposed in this paper to keep the viewpoint away from the obstacles.

4.2. System parameters

Several parameters determine how the force field affects user navigation. First, the static force field is computed in a preprocessing step. Each grid point in the workspace is associated with a force that is computed as resultant force from all boundary points in a window of size 21x21. The speed coefficient, μ_v, in eq.(3) is proportional to the magnitude of navigation input and bounded by a reasonable value. Second, the scale-down factor, η, for the direction coefficient in eq.(3) is set to 90% and the lower bound for this coefficient is 20%. When the coefficient is scaled up, the scaling factor, η, is set to 110% and the its upper bound is 200%.

5. Experiments

5.1. Experimental settings

Ten subjects were invited to test the implemented system. Each of them is asked to perform four runs of experiments. Each run of the navigation experiments uses a different assistance method. These four methods include the original method with no assistance, the path-planning method proposed in [10], the force field method proposed in this paper, and the method of combining the two aforementioned methods. They are given a short instruction about how to use the browser
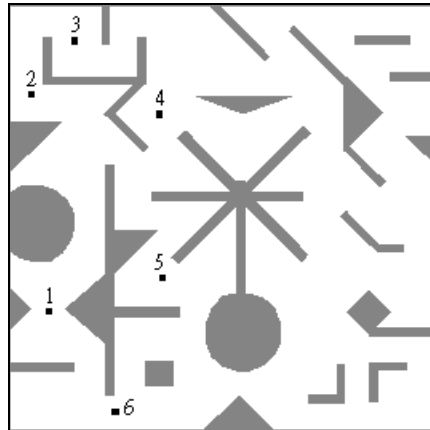

Figure 7: sequential landmarks in the experimental environment

and how to perform the experiment. The experiment asks a user to navigation through several landmarks labeled in sequence as shown in Figure 7. The user navigates in the maze with the first-person WALK mode provided by the VRML browser. In each run of the experiment, a user is asked to move the viewpoint from landmark 1 through landmark 6 as quickly as possible. A bouncing ball is placed in the scene to help the users identify the targeting landmark. In order to control the experimental environment, these landmarks are placed in a way that the navigation paths executed by different user will be of the same topology. In addition, the order that a user tries out these four methods is random such that the effects of getting familiar with the test environment is minimized.

5.2. Experimental results

The experimental results consist of two parts: objective statistic data and subjective user comments. The comparisons of the navigation performance under different methods are summarized in Table 1. Two data items: overall execution times and number of detected collisions for each method are listed. The average overall navigation time without any assistance is 160 seconds while the times with the path-planning method and the force field method are 130 and 125 seconds, respectively. This result implies that the force field method could be as effective as the path-planning method.

Another interesting result we have found is that when we apply both methods, the average navigation performance (107 seconds) is the best among all methods as shown in the fourth category of Table 1. This result implies that the force field method complements the path-planning method and therefore can be applied to enhance the overall efficiency. Intuitively speaking, the force field is used to reduce the chance of collisions with obstacles while the path-planning method resolves the collision situation if it occurs. For this experimental scene, the navigation performance can be improved by as much as 49 percents.

Table 1: statistic data (navigation time in seconds and number of collisions) of ten users using the intelligent interface

Method / User	Neither		Planner		Force Field		Force Field & Planner	
	Time	Collision	Time	Collision	Time	Collision	Time	Collision
1	128	136	82	149	83	45	82	64
2	211	231	214	487	184	135	139	151
3	91	191	98	167	130	166	99	206
4	113	142	78	159	101	66	116	316
5	176	171	178	430	161	122	109	101
6	199	427	103	236	80	139	73	95
7	129	129	119	339	115	79	116	171
8	112	43	108	81	118	33	102	41
9	120	227	95	318	98	126	88	91
10	322	496	226	732	176	111	148	117
Avg.	**160**	**219**	**130**	**310**	**125**	**102**	**107**	**135**
+	**0%**		**23%**		**29%**		**49%**	

The users' subjective comments suggest that the force-field method could be more natural than the path-planning method for navigation experience. The number of detected collisions reveals some of the rationale behind. When the force-field method is used, the number of collisions is much smaller than other methods as expected. However, when the path-planning method is used, this number is even larger than the case of using neither method. This might imply that the users tend to rely on the path planner to resolve the collision situation instead of trying hard to avoid it. When the force-field method is applied as well, we found that this number can be effectively reduced.

6. Conclusion

In this paper, we have proposed a effective method on designing an intelligent user interface for architectural walkthrough applications. An artificial force field is computed to assist users in navigating through difficult areas where the users often get stuck with the traditional user interface. This method has been successfully integrated with the low-level control loop in a VRML browser. Our preliminary experimental results show that this method can reduce the overall navigation time as effectively as the path-planning method. In addition, since both methods complement each other, the improvement on navigation efficiency is even more significant when both methods are applied simultaneously.

Acknowledgments

References

[1] Chen, Mountford, and Sellen, "A Study in Interactive 3D Rotation Using 2D Control Devices," *Computer Graphics*, 22(4):121-128 ,1988.

[2] S. M. Drucker and D. Zeltzer, "Intelligent Camera Control in a Virtual Environment," *Graphics Interface'94*, pp. 190-199, 1994.

[3] P. K. Egbert, and S. H. Winkler, "Collision-Free Object Movement Using Vector Fields," in *IEEE Computer Graphics and Applications*, 16(4):18-24, July, 1996.

[4] M.R. Jung, D. Paik, D. Kim, "A Camera Control Interface Based on the Visualization of Subspaces of the 6D Motion Space of the Camera," *Proc. of IEEE Pacific Graphics'98*, 1998.

[5] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Proc. of IEEE Robotics and Automation Conference*, pp500-505, 1985.

[6] J.J. Kuffner. "Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control," *Proc. of CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, Nov. 26-28, 1998.

[7] J.J. Kuffner and J.C. Latombe. "Fast Synthetic Vision, Memory, and Learning Models for Virtual Humans," *Proc. of CA '99: IEEE International Conference on Computer Animation*, Geneva, Switzerland, May 26-29, 1999.

[8] J. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.

[9] T.Y. Li, J.M. Lien, S.Y. Chiu, and T.H. Yu, "Automatically Generating Virtual Guided Tours," *Proc. of the Computer Animation '99 Conference*, Geneva, Switzerland, pp99-106, May 1999.

[10] T.Y. Li, and H.-K Ting ., "An Intelligent User Interface with Motion Planning for 3D Navigation," *Proc. of the IEEE Virtual Reality 2000 Conf.*, March 2000.

[11] T.Y. Li, and C.C Chang, "Path Planning with Incremental Roadmap Update for Large Environments," *Proc. of 2001 the IEEE International Conference on Robotics and Automation*, May 2001.

[12] H. Lieberman, "Integrating User Interface Agents with Conventional Applications," in *Proc. of ACM Conference on Intelligent User Interfaces*, San Francisco, January 1998.

[13] M. Maybury and W. Wahster (eds), Readings in Intelligent User Interfaces, Morgan Kaufmann: Menlo Park, CA.

[14] Neilson and Olsen, "Direct Manipulation Techniques for 3D Objects Using 2D Locator De-

vices," in *Proc. of the 1986 Workshop on Interactive 3D Graphics*, pp 175-182, 1986.

[15] B. Shneiderman and P. Maes, "Direct Manipulation vs. Interface Agents," *Interactions*, 4(6): 42-61, Nov./Dec. 1997.

[16] D. Xiao, R. Hubbold, "Navigation Guided by Artificial Force Fields, " in *Proc. of the ACM CHI'98 Conference*, pp179-186, 1998.

[17] VRML97 International Standard, URL: http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm

[18] The Java3D and VRML working group, http://www.vrml.org/WorkingGroups/vrml-java3d/