Workspace-Driven, Blended Orbital Viewing in Immersive Environments

Scott Frees and David Lancellotti

Ramapo College of New Jersey
505 Ramapo Valley Road
Mahwah, NJ 07430
{sfrees,dlancell}@ramapo.edu

Abstract. We present several additions to orbital viewing in immersive virtual environments, including a method of blending standard and orbital viewing to allow smoother transitions between modes and more flexibility when working in larger workspaces. Based on pilot studies, we present methods of allowing users to manipulate objects while using orbital viewing in a more natural way. Also presented is an implementation of workspace recognition, where the application automatically detects areas of interest and offers to invoke orbital viewing as the user approaches.

Keywords: Immersive Virtual Environments, Context-Sensitive Interaction, 3DUI, interaction techniques.

1 Introduction

One of the key benefits of most immersive virtual environment configurations is the ability to control the viewpoint using natural head motion. When wearing a tracked, head-mounted display, users can control the direction of their gaze within the virtual world by turning their head the same way they do in the real world. This advantage typically holds in other configurations as well. We refer to this viewpoint control as *egocentric* viewing, as the axis of rotation of the viewpoint is centered atop the user's physical head.

There are, however, situations where egocentric view control may not be optimal. In order to view an object from different perspectives, users are normally forced to physically walk around the area. Depending on the hardware configuration (wired trackers, etc.), this can be cumbersome. One alternative is orbital viewing [5], where the user's gaze is fixed towards the area of interest. Head movements and rotations are mapped such that the user's viewpoint orbits around the target location, constantly being reoriented such that the target is in view. Physical head rotations to the left cause the virtual viewpoint to swing out to the right, such that the user is looking at the right side of the object. Looking down has the effect of moving the viewpoint up above such that the user is looking down at the object.

Orbital viewing has its own disadvantages. Firstly, it is quite ineffective if the user wishes to survey the world –it locks the gaze direction such that the user is always looking towards the same 3D location in the world. Switching between orbital and

R. Shumaker (Ed.): Virtual and Mixed Reality, Part I, HCII 2011, LNCS 6773, pp. 185–193, 2011.

egocentric viewing can also be disruptive and confusing. Orbital viewing also presents interaction challenges when manipulating objects, as the act of orbiting while manipulated an object can affect the user's ability to control the object's position accurately. This paper presents additions to orbital viewing that allows it to be more effective in general interactive environments.

Finally, we describe how orbital viewing can be integrated into an application by linking it to known areas of interest within the virtual world – which we call "hotspots". We present an outline of how we automatically infer the existence of such locations based on user behavior.

2 Related Work

Koller et al. [5] first presented orbital viewing as an alternative method for viewing objects, in particular a WIM [7]. Several of orbital viewing's limitations were addressed in this work, such as the possibility of occlusion, importance of controlling the radius of orbit, and the possibility of increased disorientation when transitioning into / out of orbital viewing. We do not address the occlusion problem in our work, but offer implementations that relate to radius control and disorientation.

Many alternatives to egocentric view control have been presented in the literature using a wide range of approaches for both immersive environments [2, 7, 9] and desktop applications [8]. Our work is not aimed at comparing orbital viewing with other alternative techniques however; it is focused on determining and improving orbital viewing's effectiveness in interactive systems in general.

Much of our work is aimed at developing viewpoint control techniques that effectively handle changing workspaces. The user's workspace can be described as the location and size of the area the user is currently most interested in. The size and location of the workspace can greatly influence an interaction technique's effectiveness, described in detail in [1]. Our techniques for implicitly recognizing workspace are based in part on the concept of focus and nimbus, presented by Greenhalgh and Benford [3]. In addition, our concept of modeling workspaces as artifacts within the environment in which users can invoke orbital viewing is similar to the idea of landmarks introduced by Pierce and Pausch [6], where landmarks were anchor points for navigation techniques in large-scale workspaces.

3 Identifying Problems with Orbital Viewing

This research started as an investigation into whether or not orbital viewing could aid in manipulation tasks requiring the user to view objects from multiple perspectives. In our user studies, participants were presented with a virtual object (as shown in Fig. 1) that fits within a translucent object of the same shape (only slightly larger). The user was asked to position the object to fit within the target as many times as possible within a 3-minute trial. This task was selected because it would require the user to view the object from all directions in order to make the fit.

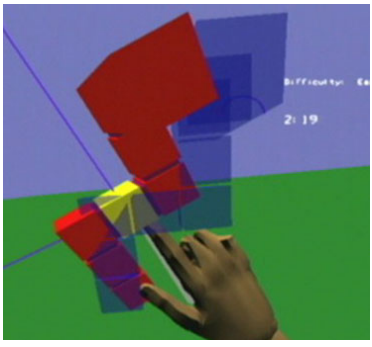

Fig. 1. User manipulating the object such that it will match the target's (blue translucent object) orientation

We conducted this experiment with 24 participants. Each participant underwent two training trials using egocentric viewing and two training trials with standard orbital viewing – where the orbital viewing configuration was centered on the target object. They completed two recorded trials with egocentric and orbital viewing in a randomized order.

ANOVA analysis of completions per trial showed no statistically significant effect for view control type ($p = 0.143$). User feedback, however, provided us with some clear areas where orbital viewing could be improved. Survey data also indicated that while orbital viewing did not help performance on the task, many participants found it more comfortable (as opposed to walking around the object).

Our first observation was that orbital viewing could make it difficult to control objects precisely. As the head rotates, the user's virtual viewpoint and avatar orbit around a center point. This becomes a problem when the user is holding a virtual object (perhaps with a hand-held stylus). If the physical relationship between the hand and head are preserved, head movement results in hand movement – and thus the virtual object will move. In Section 4.1 we present a modification that temporarily breaks the relationship between the viewpoint and hand to improve manipulation.

Another area where orbital viewing caused problems was at the beginning of trials, where if the trial was to use orbital viewing the user was abruptly switched into that mode when timing began. We observed confusion and awkwardness during these transitions. Furthermore, our overall goal is to take advantage of orbital viewing in general virtual environment applications if/when the user's workspace becomes small. If the user is to switch between egocentric and orbital viewing as their workspace size changes, the disorientation during transitions needed to be resolved. In Section 4.2, we discuss our blended view control approach that works to reduce this problem.

4 Implementation of Viewing Techniques

The implementation of egocentric viewing is straightforward – a tracking device of some kind provides position and orientation information for the user's head. The viewpoint, or virtual camera, rotates and moves in a one-to-one correspondence with

the tracker - commonly implemented by making the viewpoint a child object of the tracker representation in the scene graph (perhaps with offsets to account for eye position and offsets). An alternative way of describing viewpoint control in 3D graphics is the specification of two 3D points, *eye* and *look*. In this scenario, the orientation of the viewpoint or virtual camera is constrained to point at the "look" point. The look and eye points are rigidly attached to each other (at some fixed distance). Rotations of the viewpoint (driven by the tracker) result in translations and rotations of the look point. In effect, in egocentric viewing, the look point *orbits* the eye point (or viewpoint), as depicted in Fig. 2 (A). In our system, we used a Virtual Research 1280 HMD and Polhemus FASTRAK system. The application and interaction techniques were implemented using the SVE [4] and CDI toolkits [1].

When orbital viewing is active, the situation reverses – instead of the tracker being mapped to rotations of the eye point; head rotations are mapped to the look point. The look point's position is fixed, and the viewpoint orbits the look point. This is implemented by detaching the viewpoint from the tracker and making it a child of the look point. Rotational information (not position) is mapped from the head tracker to the look point. The result is that head movements cause the viewpoint to orbit around the look point, as shown in Fig. 2 (B).

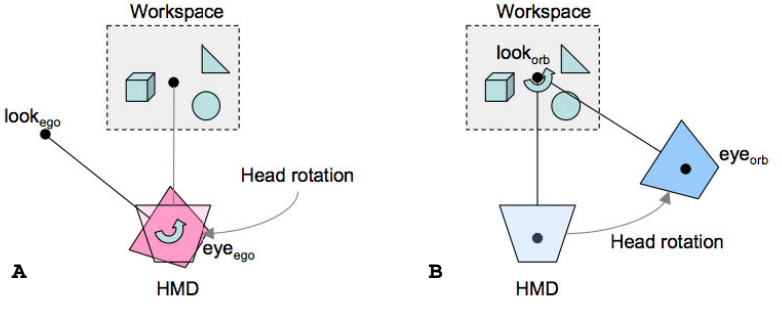

Fig. 2. (A) Egocentric viewing; (B) Orbital viewing

In an interactive system, the user must be given a method to control the radius of the orbit. In our implementation, where the user points the stylus in the general direction they wish to move and hold the button down. The vector representing the stylus direction is projected onto a vector connecting the eye and look points. If the vector projects away from the look point, the radius is increased, and vice versa.

4.1 Object Manipulation While Using Orbital Viewing

In orbital viewing, head rotation results in viewpoint rotation and translation (viewpoint *orbits* the workspace). This presents a question: should the viewpoint and virtual hand move together, as a unit, or should the virtual hand stay fixed while only the viewpoint moves? Both choices could create confusion, however the former creates real interaction difficulties since the virtual hand may move while the physical hand remains still.

Our implementation allows the hand to orbit with the viewpoint only when *not* interacting with an object. This means that if the user's hand is within view when rotating the head, the virtual hand appears still, as it moves proportionately with the viewpoint, as depicted in Fig. 3 (A). This configuration is implemented by moving the virtual hand in the scene graph such that it is a child of the viewpoint. Its local position with respect to the viewpoint is recalculated each time the stylus tracker moves such that it is equal to the distance between the stylus and HMD trackers.

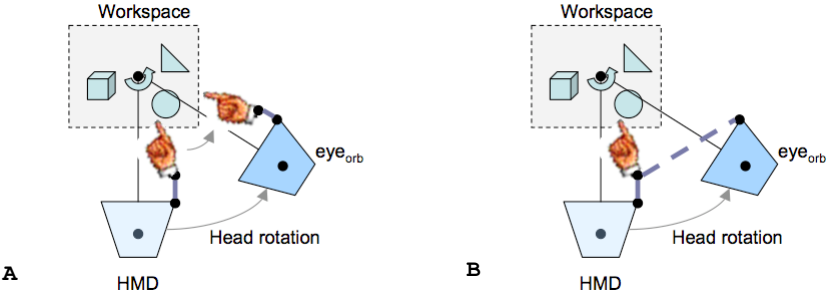

Fig. 3. (A) Virtual hand moves with viewpoint. (B) Virtual hand stays in original (physical) position, viewpoint moves without it.

In our system the user's virtual hand is tracked by a hand-held stylus, which has a single button. The user can translate and rotate an object by intersecting their virtual hand with it and holding a stylus button down. While manipulating an object, we switch to the setup depicted in Fig. 3 (B). When holding an object the virtual hand "sticks" in its original position - the cursor/virtual hand still responds to physical hand movement, but it does not orbit with the viewpoint. This creates an inconsistency between the physical hand/head and the virtual hand/viewpoint. Once the user releases the object, the virtual hand "snaps" back to where it would have been if it had orbited with the viewpoint, which restores the user to a workable state going forward.

At first look, this technique may seem confusing, as one would think it would be distracting to have your viewpoint potentially move large distances while the hand stays stationary. In our experience, however, it has proven effective. When users are manipulating objects, their attention is focused on their hand and the object and they do not expect their virtual hand to move unless the physical hand does. When head rotations cause their viewpoint to orbit the object of interest, the experience mimics real world experiences of holding an object in hand and "peering" around it to see it from another perspective. During user trials, participants needed little explanation of the technique, they adapted to it without difficulty. We suspect preserving the mapping of physical hand movements (or absence of) to the virtual cursor is far more important than preserving the relationship between the viewpoint and virtual hand, however we would like to investigate this more directly in future trials.

4.2 Blended Orbital Viewing

Switching between egocentric and orbital viewing can be disorienting, and often times being in complete orbital viewing mode may not be optimal (the user may be over-constrained). We have implemented a mechanism for blending the two viewing techniques together seamlessly, such that the user can be experiencing view control anywhere along the continuum between egocentric and orbital viewing.

We describe both techniques by maintaining two 3D points – *look* and *eye* – rigidly attached to each other. In egocentric viewing, the head tracker directly rotates the eye (viewpoint), where in orbital viewing it maps to the look point. Rather than pick a configuration, we instead keep track of both sets of points - giving us eye_{ego} and $look_{ego}$ for egocentric and eye_{orb} and $look_{orb}$ for orbital viewing using four hidden objects inserted into the scene graph.

The actual look and eye points (and thus the viewpoint's orientation) are defined by a view control variable, VC. At VC = 0, the user is in full egocentric viewing and the viewpoint is directly mapped to $look_{ego}$ and $eye_{ego.}$ When VC = 1, the user is engaged in orbital viewing – the viewpoint is mapped to $look_{orb}$ and $eye_{orb.}$ For values between 0 and 1, the eye (viewpoint) and look positions are linear interpolations of their corresponding egocentric and orbital pairs. Once the interpolated look and eye points are known, the actual virtual camera is repositioned and oriented to adhere to the configuration depicted in Fig. 4. A similar interpolation procedure is performed to position the virtual hand and cursor.

Blending on Transitions. The most straightforward use of blended orbital viewing is during transitions between full egocentric and orbital viewing. When the workspace becomes small (either determined explicitly or through observation) and the interface invokes orbital viewing, the abrupt change in viewpoint mapping can become extremely distracting. Worse yet, abruptly transitioning from orbital viewing into normal egocentric viewing can leave the user looking in a completely different direction than they were a moment before (without actually physically moving their head).

To alleviate these effects, we never move the VC value discretely; rather we gradually adjust it over a period of three seconds. The actual time interval is somewhat arbitrary, but it represented a good compromise derived from several user trials. Shorter transition intervals did little to alleviate disorientation, and significantly longer intervals tended to interfere with the task at hand. We fully expect different users might respond better to different interval values.

Blending Based on Workspace Size. Often a user might be particularly interested in a region of the world containing several objects. While this workspace may be relatively small, it could be large enough that locking into a fixed 3D location at its center using pure orbital viewing could be a hindrance to the user. By selecting a VC value between 0 and 1, the user benefits from orbital viewing (i.e. head movements cause them to orbit the center point, giving them a better perspective) while still allowing them to view a larger field of view (i.e. the look point moves within the workspace). This is depicted in Fig. 5, where the VC value is 0.75.

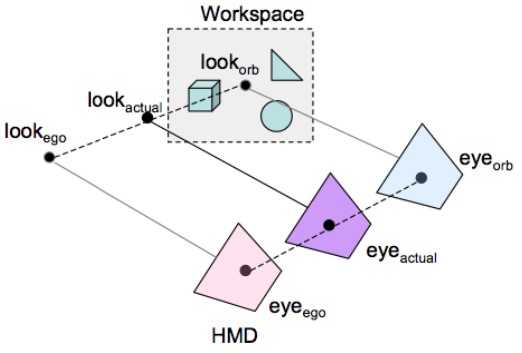

Fig. 4. Example of blended orbital viewing with VC = 0.5

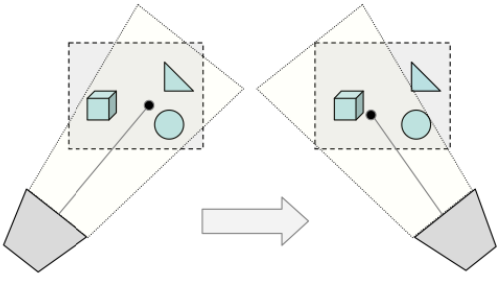

Fig. 5. With VC = 0.75, when user rotates physical head right, most of the rotation results in an orbit around to the right side (due to orbital component), however the look point also moves (due to egocentric component)

This technique could also be useful when the user gradually becomes more focused on a specific area. In the beginning of the sequence they may be surveying the world, using egocentric viewing. Over time, the user may start to focus on a set of objects, perhaps manipulating them. As they become more focused on a smaller space, the VC can gradually be increased towards orbital viewing.

5 Workspace Recognition

The user's current workspace is part of the contextual information an application can utilize when deciding the interaction technique to deliver. When working within a small, confined area, a viewing technique such as orbital viewing may be beneficial - while when working in a larger area it could be too limiting. To make decisions based on workspace, we require a way to define it quantitatively.

Our model defines three *size ranges* for the workspace: small, medium, and large; supporting different view control techniques for each range. We base view control decisions on the ranges instead of physical dimensions directly in order to allow the thresholds to be varied easily between applications or users. Current workspace can

be determined in a variety of ways – ranging from asking the user to explicitly indicate the volume to inferring it by user activity. We use a combination of explicit and implicit workspace recognition. *Potential* workspaces in the world are identified through observation – which we refer to as "hotspots". Users must explicitly "attach" to the hotspot before orbital viewing is invoked.

Workspace is inferred by recording where the user has been "looking" and interacting. We divide the virtual world into a three-dimensional grid – with each location assigned a numeric score. As the user's viewpoint moves, we increment the score of each location within the field of view. Interaction with an object near the location also increases its score. Periodically the system reduces the score of *all* grid locations by a small amount. Overtime, heavily active locations achieve high enough scores to be promoted to "hotspots" – which become permanent artifacts within the world and anchor points for orbital viewing. A hotspot has a precise location. Its size is recorded as "small" if there are no hotspots immediately adjacent to it. If there is a hotspot at an adjacent grid location, its size is set to "medium".

Hotspots are implemented such that the required score, frequency and magnitude of increments, and granularity of the 3D grid points easily varied; we have not yet done formal studies to determine an optimal configuration (if one even exits).

In addition to dynamic hotspots, we also support predefined hotspot locations that can be placed within the world. Regardless of the type of hotspot, the user interface associated with it is the same. As a user approaches a hotspot, they are given a textual indication on the top of their HMD's view suggesting that orbital viewing could be used. If they wish to invoke orbital viewing, they simply touch their stylus to their head and click its button. The user is then transitioned from a VC = 0 (egocentric viewing) up to the maximum VC associated with the hotspot. Typically, we choose VC = 1 for small (standalone) hotspot locations and VC = 0.75 for hotspots with proximate neighbors (part of medium sized workspaces). The VC is adjusted gradually as described in Section 4.1.1. To exit orbital viewing, the user touches their stylus to their head and clicks – which gradually reduces the VC back to 0.

6 Conclusions and Future Work

We have created several features that can be implemented with orbital viewing that we've seen reduce confusion and disorientation while allowing users to manipulate objects in more natural manner. In follow-up experiments post-trial user feedback has improved. We believe the system for recognizing workspaces, or hotspots, in the virtual world is a solid stepping stone in automatically recognizing areas of interest, and that using blended orbital viewing to ease transitions is a valuable addition to the view control interface.

Our greatest challenge has been quantifying performance results. The user experiment described in Section 3 was specifically designed to be difficult if users did not try to view the object from various angles. Our expectation was that if orbital viewing really helped the user to do this, participants would attain more completions on those trials. Even with our improvements however, the actual manipulation task is hard enough that it seems to outweigh the effects of the view control method. In subsequent experiments with the same design, view control technique still has not

appeared significant. Reducing difficulty (making the target larger such that it is easier to fit the control object within it) tends to reduce the importance of the viewing technique. We have considered experiments that only require the user to view an object or area from multiple directions (without needing to interact), however we feel this would be a trivial comparison – as obviously rotating one's head would be faster than walking around to the other side of the object. We are currently investigating alternative methods of evaluating the effect orbital viewing has on general interaction.

Acknowledgements. This work was funded by the National Science Foundation, grant number IIS-0914976.

References

1. Frees, S.: Context-Driven Interaction in Immersive Virtual Environments. Virtual Reality 14, 277–290 (2010)
2. Fukatsu, S., Kitamura, Y., Toshihiro, M., Kishino, F.: Intuitive control of "birds eye" overview images for navigation in an enormous virtual environment. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 67–76 (1998)
3. Greenhalgh, C., Benford, S.: Massive: A collaborative virtual environment for teleconferencing. ACM Transactions on Computer-Human Interaction 2(3), 239–261 (1995)
4. Kessler, G.D., Bowman, D.A., Hodges, L.F.: The Simple Virtual Environment Library, and Extensible Framework for Building VE Applications. Presence: Teleoperators and Virtual Environments 9(2), 187–208 (2000)
5. Koller, D., Mine, M., Hudson, S.: Head-Tracked Orbital Viewing: An Interaction Technique for Immersive Virtual Environments. In: Proceedings of the ACM Symposium on User Interface Software and Technology, pp. 81–82 (1996)
6. Pierce, J., Pausch, R.: Navigation with Place Representations and Visible Landmarks. In: Proceedings of IEEE Virtual Reality, pp. 173–180 (2004)
7. Stoakley, R., Conway, M., Pausch, R.: Virtual Reality on a WIM: interactive worlds in miniature. In: Proceedings of CHI 1995, pp. 265–272 (1995)
8. Tan, D.S., Robertson, G.G., Czerwinski, M.: Exploring 3D Navigation: Combining Speed-coupled Flying with Orbiting. In: CHI 2001 Conference on Human Factors in Computing Systems, Seattle, WA (2001)
9. Tanriverdi, V., Jacob, R.: Interacting with Eye Movements in Virtual Environments. In: Proceedings of SIGCHI on Human Factors in Computing Systems, pp. 265–272 (2000)