Fly Over, a 3D Interaction Technique for Navigation in Virtual Environments Independent from Tracking Devices

Pierre Boudoin, Samir Otmane, Malik Mallem

Fly Over, a 3D Interaction Technique for Navigation in Virtual Environments Independent from Tracking Devices

Pierre Boudoin* Samir Otmane* Malik Mallem*
(*)IBISC Laboratory - University of Evry/CNRS FRE 2873, France
E-mail: pierre.boudoin@ibisc.univ-evry.fr, samir.otmane@ibisc.univ-evry.fr, malik.mallem@ibisc.univ-evry.fr

ABSTRACT

A good 3D user interface associated with 3D interaction techniques is very important in order to increase the realness and the easiness of the interaction in Virtual Environments (VEs). 3D user interfaces systems often use multiple input devices, especially, tracking devices. So, the search for new metaphors and techniques for 3D interaction adapted to the navigation task, independently from devices used, is an important stage for the realization of future 3D interaction systems that support multimodality, in order to increase efficiency and usability.

In this paper we propose a new tracking device-independent 3D interaction model called *Fly Over*. The core principles of *Fly Over* make it compatible with all 3D/2D-pointing devices.

CR Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces-Interaction styles, I.3.6 [Computer Graphics]: Methodology and Techniques-Interaction Technique.

Additional Keywords:3D user interfaces, 3D interaction techniques, multiple devices, navigation task, virtual environment, Egocentric navigation.

1 INTRODUCTION

3D user interfaces implies the co-existence of different interfaces and devices, especially tracking/pointing devices. The main issue existing in these systems is that the utilization of multiple tracking devices implies different ways to use each of them. So each time the user changes from one device to another, he needs to undergo a new training process. Generally, this issue occurs when the user has to change from a 3D interaction technique to another because majority of these techniques are highly dependant from hardware. These changes happen when the user starts another one of the four 3D interaction tasks (Navigation, selection, manipulation and control command) or when he wants to use another device.

In order to lessen this issue, we have designed a new 3D interaction technique called *Fly Over*, which is dedicated to multiple tracking devices. Whatever the tracking device used, *Fly Over* works in the same way, because it uses a generic principle based on the only knowledge of the position of the pointing device. It uses a partition of the 3D space around the user hand to allow him to execute some basic actions like translate and rotate.

This article is organized as follows. Section 2 will briefly review related work about, navigation task, 3D navigation techniques and how to recover the six degrees of freedom (translations and rotations) with at least 3 degrees. Section 3 presents the *Fly Over* model. Finally, section 4 describes the experimental setting on a semi-immersive VR/AR platform and the utilization of *Fly Over* for a navigation task.

2 RELATED WORK

The navigation task is probably the most utilized task in immersive VEs. Navigation permits user to move in the VE, in order to obtain different views of the scene and, so, to locate himself in the three dimensional space [1]. Navigation task may be divided in three subtasks: *Explore*: for navigating without any objective(s) in order to increase the user knowledge about the VE configuration; *Search*: for navigating with a specific aim, ask to locate an object or a route and to travel to it; *Inspect*: for navigating with precision in order to get a particular view of the VE or of an object. A good navigation technique must realize efficiently these three subtasks, whereas avoiding sickness feelings.

Many researchershave worked on basic principles in 3D Navigation. So, a lot of metaphors for navigation in immersive VEs have been explored. There, are some techniques, which drew our attention. *Gaze-directed steering* [2], *Free flying* [3], *Eye-in-hand* [4], *Pointing* [2] techniques: describe the direct control of the virtual camera by the user. In the case of *World-in-miniature* [5, 6] *and Map-based* [7] techniques, the user has a tiny representation or a map view of the VE and he points out where he wants to go. In the case of *Fisheye views* [8] *and Perspective wall* [9] techniques, a distortion of views of the VE is used in order to expand the space seen. However, these techniques may sometimes disorient the user [10]. The *Speed-coupled flying* [10] technique allows the user to navigate within local as well as global views of the world by coupling speed control to height and tilt control. Fukatsu designed a technique to intuitively control the "bird's eye" overview display of an entire large-scale virtual environment in a display system that present a user with both global views and local view simultaneously [11].

However, most techniques are highly dependent from hardware interface because their design implies a specific action(s) (translation, rotation, gesture, clicking, pressing, etc) with a specific user's body part (hand, head, finger, legs, etc). For example: *Gaze-directed steering* technique [2] needs user's head tracking, *Pointing* technique [5] needs user's hand tracking, *Map-based travel* technique [7] needs a 2D display and a pointer, *Grabbing the air* technique [12] needs pinch-gloves.

These techniques are efficient for an isolated navigation task. But if we consider a global action in a VE (including navigating, selecting or manipulating objects), different devices may be needed and switches between tasks and devices may be difficult to handle and add a lot of cognitive load for the user.

Navigation is composed of two geometrical transformations: translation and rotation. We wanted our technique to be generic and to be independent from the tracking devices used, so we had to study these basic transformations and especially how to create 3D rotations with only 2D (e.g. mouse), 3D (e.g. SpaceBall) or 6D (e.g. Flystick) pointing devices. We have taken inspiration from Chen [13] and Shoemake [14] works which uses a mouse (2D) and adapt them in order to make our technique work with 3D and 6D pointing devices.

3 FLY OVER MODEL

This new 3D interaction model - called *Fly Over* - is based on the following four constraints:

-To be compatible with all common 2D, 3D or 6D pointing devices (mouse, hand/head/finger tracking, force feedback) that could return a 2D/3D position/orientation of the user or an object he manipulates;

-To maintain the same logic of use for all devices, even if the employed technologies are very different from each other;

-To be natural;

-To be associated with a short training duration.

In order to fulfill our four constraints, we propose a generic model based on two main ideas. First, all basic interaction tasks (navigation, selection, manipulation) may be turned into simple pointing tasks. Second, the 6D space (3D position and 3D orientation of the user) may be parted into subspaces where different pointing tasks may be performed. This concept is very interesting because it permits to use all 2D, 3D and 6D pointing devices (mouse, Flystick, SPIDAR, etc).

3.1 Generic model specification

The interaction between a user (in the real world) and VE consists on a loop in which a set of devices gives information to the user whereas the user may modify VE by using another set of devices. At each time, the action of the user on VE may be depicted as a real vector *M* of dimension in an area .

Whereas standard 3D interaction techniques transform directly the value of *M* into an action in VE (selection, manipulation, navigation, control command tasks), we use an intermediate space in VE in which a pointing task is carried out. Depending on the value of , a determined action is performed in VE.

We will call F, the projection function which maps into . The idea is to transform a complex task involving m dimensions into simpler tasks involving dimensions less or equal than 3. This task may be executed simultaneously or sequentially.

In order to model the fact that many simpler tasks may be available at each time, we divide into *n* subspaces .

is built as follows:

,

Each may be associated to a set of couples by using a function *G*:

,

Where:

is an elementary action (e.g. unitary translation, rotation, etc.)

is an optional real value parameter that may indicate the magnitude of the elementary action in VE.

All the process may be summarized by:

(1)

4 *FLY OVER* FOR NAVIGATION TASK: *FLY OVER-N*

4.1.1 Model

Fly Over-N is a particular model derived from the generic *Fly Over* model in order to be dedicated to the navigation task. The number *n* of Z_i subspaces and their dimension has been determined by the following observations from VE navigation experience:

-Managing simultaneously translation and rotation in VE on a (semi-)immersive VR platform may cause nausea for the user;

-The users naturally choose their orientation in order to have the aimed object in front of them, and then translate to the object.

These observations were compatible with the fact that, within the *Fly Over* model, it is possible to decouple the 6D navigation task into two 3D subspaces, modeled as two concentric spheres centered on O, and so defines the two interaction areas: a subspace dedicated to the translation of the user in VE (Z_2 area) and another one dedicated to the orientation of the user in VE (Z_1 area). This leads to the *Fly Over–N* model depicted in figure 1.

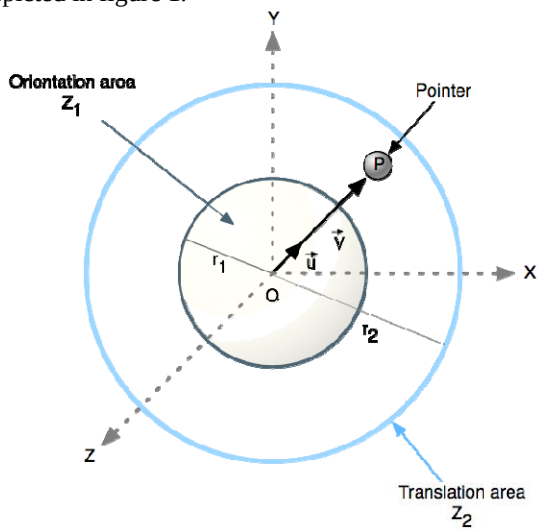

Figure 1. *Fly Over-N* areas in VE. Z_1 is a sphere in which associated actions in VE are rotations. Z_2 is associated to translations in VE.

So, within a limited space around the hand of the user, the user is able to orient himself. When the user extends his hand out of this area, his movements are mapped to translations.

4.1.2 Actions and parameters

In the *Fly Over-N* model (see figure 1), the actions in VE to define in order to navigate were obvious: translate and orientate. We have also settled the parameters to associate with the two actions.

-**Translate** requires two parameters: a direction vector and the translation magnitude. They may be determined by the only knowledge of the pointer position in the blob's referential (P). Thus, the unit vector of gives the navigation direction and we get the norm with this equation: .

-**Orientate** consists in pointing in a given direction. So, we just need to use the unit vector of to accomplish an orientation.

But in practice, we need another action in order to stop moving:

-**Stop moving** requires putting the pointer into a rest area around the blob's origin and with a small a very small radius

However, two kinds of problems occur when user moves the pointer from interaction area 1 to 2. Indeed, a discontinuity exists when a user crosses from the orientation area to the translationarea, which are two different actions in VE. It produces a sickness effect. Another problem occurs when the user wants to modify his orientation in VE. Again, it may produce sickness effect if the user moves the pointer too quickly from one location to an opposite one in the orientation area Z_1. We have proposed a solution to lessen the sickness effects.

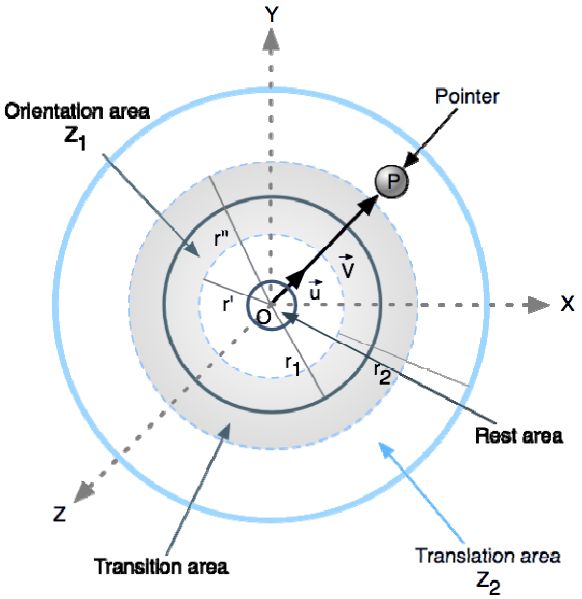

Figure 2. The three interaction areas of *Fly Over-N*.

To solve the first problem, the main idea is to build an intermediate area (see figure 2) between the orientation and translation areas in *Z*, in which translation and orientation elementary actions are performed simultaneously. The effect will be to smoothen the transition from orientation action to translation action and vice versa. The transition area is defined by r' and r'' radiuses.

Then, we introduce two weighting coefficients ( ). is used to modulate the magnitude of the orientation, while is used to modulate the magnitude of the translation. These coefficients depend on the distance between the pointer and the blob's origin: (see figure 3).

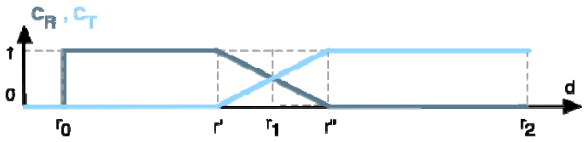

Figure 3. Weighting coefficients variations.

So, when the pointer is in this transition area, the user accomplishes translation and orientation actions at the same time, but not in the same proportion, depending on which area the pointer is the nearest. Obviously, when the pointer is in the rest area (see figure 2), the two weighting coefficients are null in order to stop moving.

Based on this updated model, the translation vector may be written as follows:

(2)

With:

.

Where:

: Magnitude of the translation vector,
: Distance from blob origin to pointer,
: Weighting coefficient depending on *d*.

To solve the second problem, we haveintroduced a rotation speed in the orientation modification process. The principle is to point in a start direction to an end direction with a speed of rotation depending on . These starting and ending directions are defined by the positions of the pointer ( ) in the blob's referential, taken at two moments ( ) and so, the start and end directions are and . The magnitude of the rotation speed is given by the following equation:

(3)

The result is a smooth transition, which lessen the sickness effect.

5 EXPERIMENTAL FRAMEWORK

5.1 VR platform

Our experiments have been performed on a semi-immersive multimodal platform (see figure 4), which permits to follow the gestures of the user's hand and finger positions (wireless Flystick 1 coupled to two ARTTrack1 infrared cameras, wireless 5DT Data-Gloves Ultra 14) and has a 6D force feedback device (SPIDAR-G [15]). Each device is associated with a specific server. We utilized the VRPN library [16] to implement the gathering of all our data from the different servers and Virtools™ to make the interactive virtual environments needed in our experiments and to implement *Fly Over*. We also have got a wide display and digital projector with active stereo capabilities.

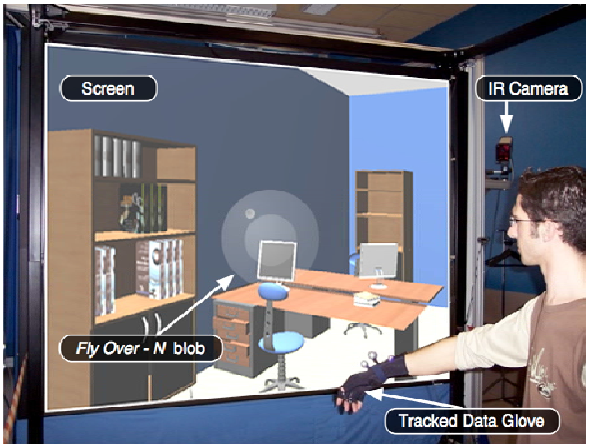

Figure 4. Our semi-immersive VR/AR platform. We can see Fly Over-N working with optical tracked wireless Data Gloves

5.2 *Fly Over-N* in practice

5.2.1 *FO – N* using 6 dof optical tracked devices

In this subsection, we explain how we have used *Fly Over-N* with an optical tracked wireless Data-Glove. First, the user must be in the tracked area. When he is in position and wants to start the navigation, he must initialize the *Fly over-N* technique. To do so, he specifies the origin of *Fly Over-N* blob in the world coordinates by performing a specific initializing action (see figure 5), which may be a pre-defined hand gesture (if Data-Gloves are used) or pressing a button (when a Flystick is used). Once this step is completed, the blob is around the user hand. Now the blob is in an idle state (see figure 6).

In order to navigate in the VE, the user must move the pointer by moving the hand into the blob and doing a specific action for allowing the displacement (see figures 7), for example, closed fist.

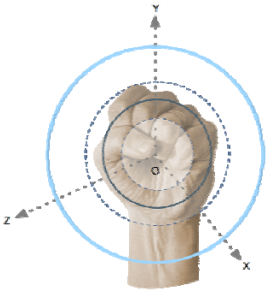

Figure 5. *FO-N blob* initialization step.

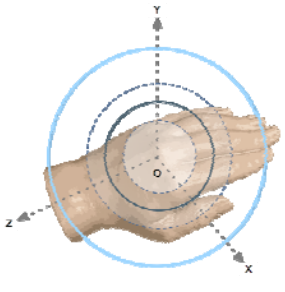

Figure 6. *FO-N blob* idle state.

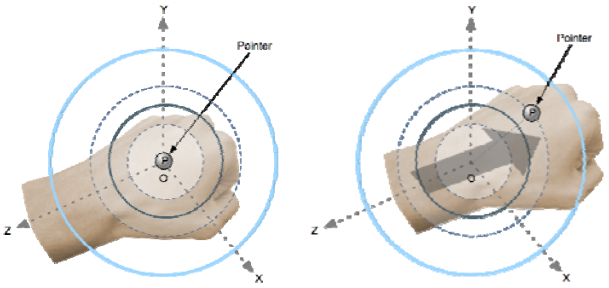

Figure 7. *FO-N* blob displacement using an optical hand-tracking device.

To cease navigating and go back to the idle state, the user just needs to stop doing the action allowing the displacement. For example, open the hand if Data-Gloves are used or release the button when the Flystick is used.

If the user needs to re-initialize the technique, he may execute at any time the initializing action.

5.2.2 *FO – N* using a SPIDAR

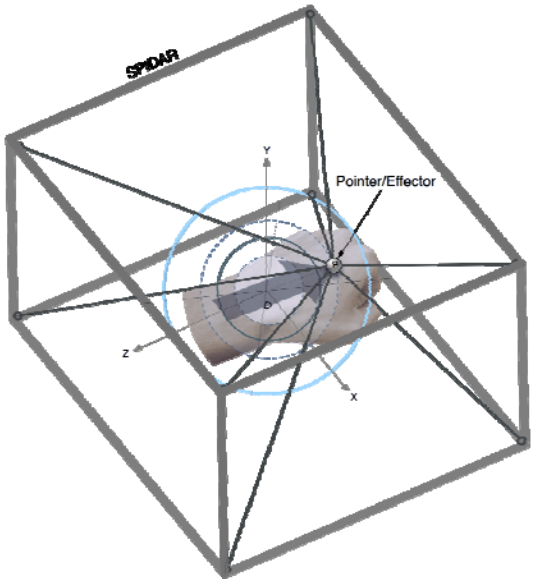

Figure 8. *FO-N* blob displacement state using a SPIDAR.

Using *Fly Over – N* with a SPIDAR is straightforward. The blob is situated in the middle of the

workspace of the SPIDAR. The pointer is the representation of the effector of the SPIDAR in the virtual environment (see figure 8). So, navigation is done in the same way as seen before.

When we use the SPIDAR, the initialization step is not needed because the origin of the *Fly Over – N* blob will always be the middle of the SPIDAR.

5.2.3 *FO – N* using a mouse

Navigation using *Fly Over – N* with a simple 2D mouse was easy to implement. The mouse being only a 2D pointing device, we needed to map the mouse's referential to the blob's referential in two times. When the left mouse button is pressed we can navigate in the $(o,\vec{X},\vec{Z})$ plane (see figure 9) and when this is the right mouse button, which is pressed, navigation is done in the $(o,\vec{X},\vec{Z})$ plane (see figure 10).

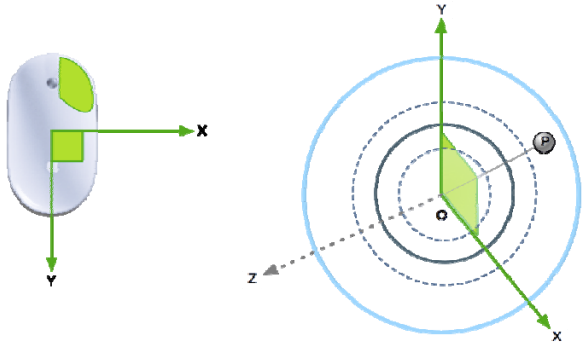

Figure 9. *FO-N* blob displacement state in the $(o,\vec{X},\vec{Y})$ plane using a mouse.

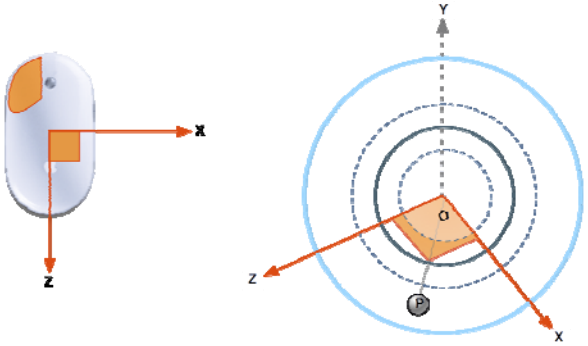

Figure 10. *FO-N* blob displacement state in the $(o,\vec{X},\vec{Z})$ plane using a mouse.

6 Preliminary Evaluations

12 young students (10 males and 2 females) participate to the preliminary evaluation. 2 of them considered themselves as experts in using VE systems whereas 4 considered themselves as intermediate and 6 as beginners. However, none of them have already utilized *FO–N*.

Virtual Environment was the representation of a part of our laboratory. For *FO–N*, the device used to navigate was a Flystick. Figure 11 shows the experimental setting. The participants were asked to follow 3 times as precisely as possible a trajectory in VE depicted with a thin red line, going from point A to point B (see figure 12). Duration of the experiment was not considered.

The target trajectory was built to be sinuous. The main question was: is it easy to follow the target trajectory with *FO-N*?

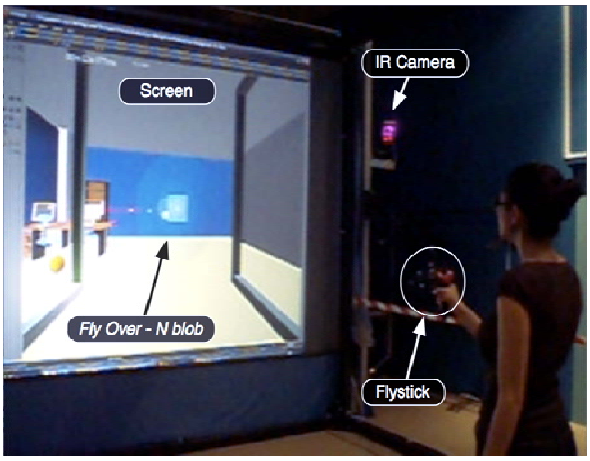

Figure 11. Experimental setting with the use of *FO–N* on the semi-immersive VR/AR platform. Users navigate by moving a Flystick in their hand, which position is computed by two infrared cameras.

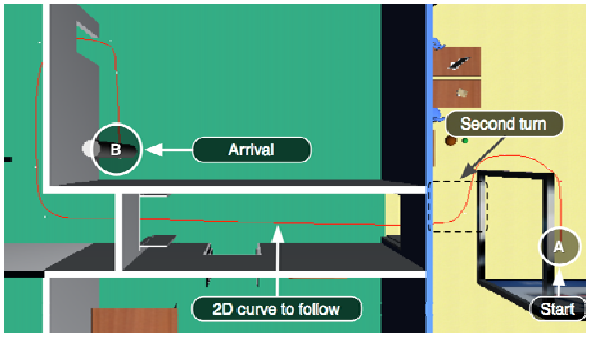

Figure 12. Target trajectory.

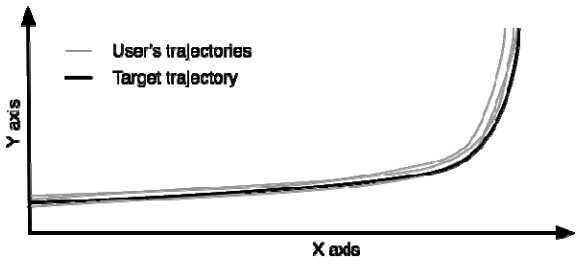

Figure 13. Comparison between the target trajectory and the trajectories followed by one user in the second turn with the blob representation.

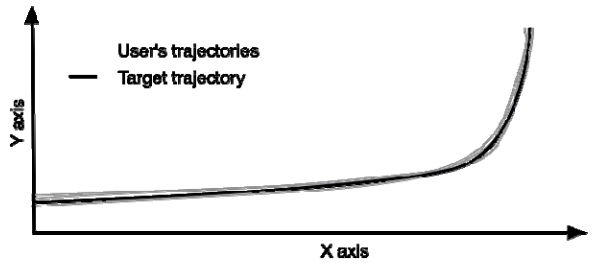

Figure 14. Comparison between the target trajectory and the trajectories followed by one user in the second turn without the blob representation.

Data shows that the use of *FO-N* gives smooth trajectories (figure 13). However, there exists a bias for *FO-N*, especially when the trajectory is curved, : users are performing trajectories that are near from the target trajectory but not centred on it. It seems this bias exists due to the presence of the blob on the screen in front of the user. Indeed calculus is done as if the blob was around the user's hand, but the representation of this blob is in front of the user's eyes. Users tried to make the blob follows the trajectory, which has probably caused this bias. We made some tests without the representation of the blob and this bias was significantly reduced (see figure 14). We think this blob should be displayed when the user is learning *Fly Over – N* and should be hidden when this learning is done in order to not distract him.

Participants were also given qualitative questionnaires after the experiment:

Q1-*Did you find easy to learn the FO-N?*

Q2-*Did you found easy to navigate with FO-N?*

Q3-*Did you found easy to follow the target trajectory?*

Q4-*Did you feel sickness?*

The possible answers were *Agree*, *Neutral* and *Disagree*.

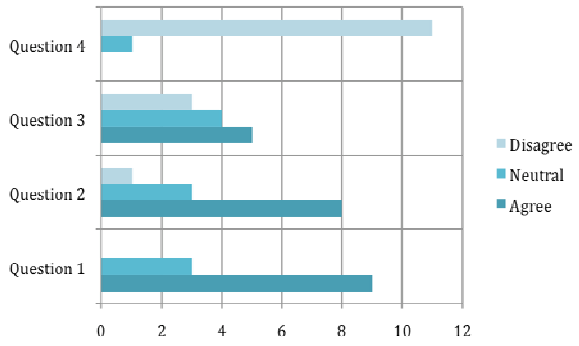

Figure 14. Questionnaires results.

The results (see figure 14) show that *FO-N* was well accepted and seems to be easy to use and to learn. Indeed, for Question 1, we got 9 Agree, 3 Neutral and 0 Disagree, which signify that users learnt to use *Fly Over – N* shortly. For Question 2, we got 8 Agree, 3 Neutral and 1 Disagree. For Question 3, we got 5 Agree, 4 Neutral and 3 Disagree. The results for these two questions show that *Fly Over – N* seems to have a good usability. Finally, for Question 4 we obtained a 0 Agree, 1 Neutral and 11 Disagree, which means that all the users felt comfortable with *FO-N*.

7 Conclusion and Prospects

In this paper, we propose a new tracking device-independent 3D interaction model, called *Fly Over*. This model is generic and is based on one main idea: 3D interaction tasks involving a set of devices may be turned into simple pointing tasks which may be performed simultaneously or sequentially by applying several projections from the input sensors space (which dimension may be quite important) to 3D spaces materialized in VE where pointing tasks are achieved. Due to this idea, *Fly Over* may be utilized the same way with various 2D, 3D or 6D devices.

For the navigation task, we describe a model called *Fly Over-N* derived from Fly Over generic model. In this model, the 6D space of the user (3D position and 3D orientation) may be seen as a set of hyperspaces in which a separate pointing task may be applied. The Fly Over-N model has been implemented and tested with an optical tracked wireless Data-Glove, a SPIDAR tracked Data-Glove, a mouse and an optical tracked Flystick. Preliminary evaluations seem to show that Fly Over generates smooth trajectories and is well accepted by the users.

Ongoing work is concerning the evaluation of *FlyOver- N* for a real 3D navigation task in submarine environments (French ANR Digital Ocean project) using different devices (mouse, SPIDAR, tracked Data Gloves and Flystick).

Due to the basic principle and its ability to work with any pointing devices, we believe that it may be utilized, in a near future for other 3D interaction tasks than navigation, such as manipulation and control command tasks. Our goal will be to show if our technique leads to a continuity feeling between tasks when switching from a device to another, and if the total training time is lessen, as we suppose to be.

8 Acknowledgments

We wish to thank "Le ConseilGénéral de l´ Essonne" and "Le C.N.R.S." for funding this work.

References

[1] Bowman, D.A.,Kruijff, E., LaViola, J.J., Poupyrev, I. 3D user interfaces: Theory and Practice. Addison- Wesley, pp. 1-26, 87-287. 2005.

[2] Mine, M. Virtual Environment Interaction Techniques (Technical Report TR95-018). UNC Chapel Hill CS Dept. 1995.

[3] Ware, C. and Jessome, D.. Using the Bat: a Six-Dimensional Mouse for Object Placement. In IEEE

Computer Graphics and Applications, 8(6), 65-70. 1988.

[4] Ware, C. and Osborne, S. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. Proceedings of the ACM Symposium on Interactive 3D Graphics, in Computer Graphics, 24(2), 175-183. 1990.

[5] Paush, R., Bumette, T., Brockway, D. and Weiblen, M. Navigation and locomotion in virtual worlds via flight into hand-held miniatures. In Proceedings of ACM SIGGRAPH, pp. 399-40. 1995.

[6] R. Stoakley, M. Conway. and R. Pausch, "Virtual Reality on a WIM: interactive worlds in miniature", In Proceedings of CHI'95, pp. 265-272. 1995.

[7] Bowman, D.A.,Wineman, J., Hodges, L., Allison, D. Designing Animal Habitats Within an Immersive VE. IEEE Computer Graphics &Applications, 18(5), pp. 9-13. 1998.

[8] Furnas, G. Generalized Fisheye Views.In Proceedings of CHI '86, ACM Press, pp16-23. 1986.

[9] Mackinlay, J., Robertson, G., Card, C. The Perspective Wall: Detail and Context Smoothly Integrated. Proceedings of the CHI '91, ACM Press, pp173-179. 1991.

[10] Tan, D. S., Robertson, G. G., and Czerwinski, M. 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Seattle, Washington, United States). CHI '01. ACM, New York, NY, 418-425. 2001.

[11] Fukatsu, S., Kitamura, Y., Masaki, T., and Kishino, F. 1998. Intuitive control of "bird's eye" overview images for navigation in an enormous virtual environment. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology (Taipei, Taiwan, November 02 - 05, 1998). VRST '98. ACM, New York, NY, 67-76. 1998.

[12] Mapes D., Moshell. J. A Two-Handed Interface for Object Manipulation in Virtual Environments. In Presence: Teleoperators and Virtual Environments, 4(4), pp. 403-416, 1995.

[13] Chen, M., Mountford, S. J., and Sellen, A. A study in interactive 3-D rotation using 2-D control devices. In Proceedings of the 15th Annual Conference on Computer Graphics and interactive Techniques, pp121-129. 1988.

[14] Shoemake, K. ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In Proceedings of the Conference on Graphics interface '92 (Vancouver, British Columbia, Canada), pp151-156. 1992.

[15] Sato, M. A String-based Haptic Interface: SPIDAR. ISUVR2006, Vol.191, 2006.

[16] Taylor II, R.M., Hudson, T.C., Seeger, A., Weber, H., Juliano, J. and Helser, A.T. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. In ACM Symposium on Virtual Reality Software and Technology, pp. 56-61, 2001.