Context Sensitive Flying Interface

Colin Ware* and Daniel Fleet*

Faculty of Computer Science
University of New Brunswick

Abstract

The requirement to change scale frequently is common to many 2D and 3D applications. Users must "zoom in" to examine details and "zoom out" to appreciate the context. This presents a problem in the context of "fly by" interfaces that use a flying metaphor to enable the user to change the point of view and explore a data space. The problem is that radical changes in velocity sensitivity may needed when working at different scales. A method is described that uses continous depth sampling to modulate the flying speed. The distribution of depths in the current frame of animation is used to set the Device to Control ratio so that it is always comfortable when operating over a range of scales. This is called Depth Modulated Flying (DMF). A family of related methods are evaluated in a task that requires subjects to search for small targets in a scene. The results show that scaling the velocity control by the near point in the scene and by the average point in the scene are equally effective.

CR Categories and Subject Descriptors: H.5.2[User Interfaces]: Input devices and strategies; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques.
Additional Keywords: Camera control, Viewpoint control, 3D Interaction.

1 INTRODUCTION

In interacting with geographical spatial data it is not uncommon to work at multiple scales. In many cases the scale problem is dealt with by using multiple windows - a large working window and a small overview window to provide context [12, 6]. An alternative is to make zooming in and out so easy that when contextual information is desired, the user can rapidly move out for an overview and back in to continue work. This is the approach taken Pad++ [2] and in other 2D display systems such as [1].

*Faculty of Computer Science, University of New Brunswick, P.O. Box 4400, Fredericton, NB. Canada, E3B 5A3
cware@UNB.ca | dfleet@UNB.ca

In a notable study of the uses of scaling in 3D environments, Robinett and Holloway suggest that an efficient navigation technique is to "shrink the world down until the destination is within arm's reach and then expand the world, continuously steering the center of expansion so as to arrive at the correctly scaled destination" [7]. Once the world is shrunk, the hand can be placed at a center of interest in the 3D space and the world scaled up around that point. In this way the center of interest appears at arms length. We wished to achieve the same functionality without an explicit scale operation.

1.1 Application Requirements

We are engaged in building a highly interactive system to visualize 3D geographic data. This is being done with special attention to the needs of oceanographers. Our data includes Digital Elevation Maps (DEMs) of the ocean floor, sometimes draped with information about bottom type - for example its softness. We also include seismic data that provides a cross section of the strata beneath the seabed when this is available. Artifacts such as pipelines, ships and buoys may be placed on or above the sea floor. Some of our data must be viewed and interacted with at multiple scales. In one example we have a DEM representing the entire North-Western Atlantic; embedded in this is a DEM representing a few hundred kilometers offshore from Halifax, Canada, and embedded in this is a DEM representing a few hundred meters of sea floor around a wreck. The resolutions of these data sets vary by four orders of magnitude.

Our challenge is to create a viewpoint manipulation interface that works effectively over these large scale ranges and allows us comfortably navigate and interact with the data without having to manually adjust a speed control. We have found that a flying interface is effective in this application domain but as we encountered more examples that required large scale changes the fixed velocity control became a problem. A stop-gap measure was the addition of a user interface for rescaling the scene, but this seemed to require too much user intervention. We needed an automatic solution. Before we describe our solution we introduce some of the prior work on flying interfaces and some of the other features of our system.

1.2 Flying Interfaces

Flying interfaces are common to many virtual environments. As Chuck Blanchard said "Nobody walks in VR, they all fly" [4]. Ware and Slipp [9] found that by using a non-linear mapping between hand movement and velocity, subjects in a controlled experiment could navigate through a tunnel that

changed in width over the course of its length by four orders of magnitude. However, this stretched the limits of their ability to control their motion. At the extremes (both small and large scale navigation) this was not an easy interface to use. An interesting result of the Ware and Slipp study was that subjects tended to maintain a constant velocity relative to the local scale of their environment. This suggests that an interface that adapts the motion sensitivity to the local environment should be desirable.

Point of view (POV) navigation is a method for moving the viewpoint based on selected points in the scene (MacKinlay et al. [5]). In this technique, the time to reach a target is proportional to the logarithm of the distance to the surface. A simple way of implementing this is to make the velocity towards the point of interest directly proportional to the distance to that point. $V = Kd$.

POV navigation is an excellent method for zooming in and out of scenes and working at a variety of scales. A great benefit of POI navigation is that it is scale independent. It takes the same time to halve the distance to a target if we are working at astronomical scales, microscopic scales, or anything in between. This is unlike most flying interfaces that have an implicit default velocity built in. What POV navigation lacks is a method for setting an arbitrary view. In addition, it is not designed to be used for creating smooth camera paths through a scene such as might be made for an illustrative movie.

POV navigation is not the only way objects have been used in controlling viewpoint navigation. Drucker and Zeltzer generated a kind of potential space, using simple functions of nearby surfaces to guide camera paths [3]. This allowed, for example, the camera to move when the view of a subject was blocked by another object.

In the DMF interface, we combine a flying interface with the idea of making velocity proportional to distance. The idea is to sample the Z buffer in order to obtain information about the depth distribution in the environment and modulate flying speed accordingly. In one variation we also use a differentially weighted sum of the depths in the scene in a way that varies with the direction and rate of turn.

1.3 The Cylopean Scale

There is a particular geometric manipulation that is at the heart of our visualization environment. We call this the "cyclopean scale" and although it has been described previously [10], we describe some of its salient features here because they bear directly on our depth modulated flying interface.

We initially developed the cyclopean scaling method to help solve some of the problem associated with stereo displays. The manipulation is easy to describe in geometric terms. The scale consists of re-sizing the virtual environment about a point midway between the two eyes of the observer as shown in Figure 1.

This manipulation achieves a number of things:

- It brings the virtual environment to a position just behind the screen where the focus information presented to the eye is consistent with the degree that the eyes have to converge. (A lack of convergence focus consistency is thought to be a cause of eye strain.)
- The stereo depth for a distant scene is enhanced. Note that the two stereo images of distant objects are identical except for a translation equal to the eye separation. However if cyclopean scale is applied, then stereo depth appears.
- Since the scene is scaled to appear just behind the screen, it is usually in a natural position for direct manipulation operations.

The widgets that make up the flying interface are shown in Color Plate 1. This interface consists of a set of three widgets that control forward and backward motion, rotation about horizontal and vertical axes, and lateral translation in the plane of the screen (= virtual windshield) respectively. In order to move forward, for example, the user depresses the mouse button over the "forward" widget and drags the mouse in a forward direction. The greater the movement, the greater the increase in velocity. The widget retains motion information and so if the movement is repeated the forward velocity can be increased indefinitely [10].

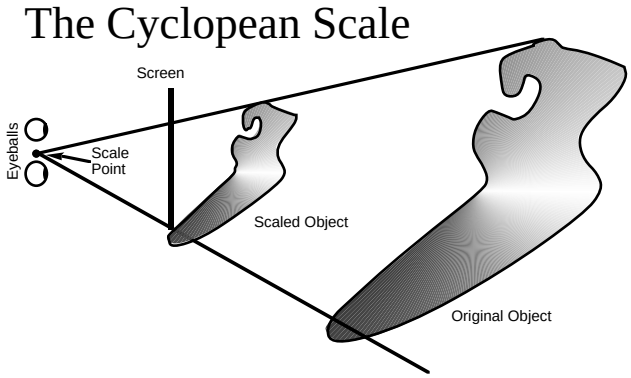

Figure 1 The Cyclopean scale is a re-scaling of the scene about the midpoint between the two eyes.

2 DEPTH MODULATED VELOCITY

In order to modulate velocity based on depth information, we sample the Z buffer continuously so that the depth information from a given frame is used to modify the velocity for the next frame. We sample the Z buffer with 15 equally spaced horizontal lines. The reason for limiting the number of samples is the overhead of reading the Z buffer. Also, line sampling is more efficient than point sampling for our graphics hardware.

We have implemented five different methods for modulating flight velocity based on the depth samples. In all of these methods the speed can be controlled via the set of widgets shown in Color Plate 1.

1) No velocity scaling: This interface does not use the depth buffer information. Velocity is varied with the user interface widgets only.

*ViewpointVelocity = K*WidgetVel*

The constant K is set to make it possible to navigate near to the smallest targets in the scene, and still have reasonable speed in macro scale.

2) Near point scaling: In this mode, the near point of the scene is determined by sampling the Z buffer. This is then used to modify the forward and sideways motion obtained from the widgets.

*ViewpointVelocity= K*WidgetVel/(nearPoint)*

3) Far point scaling: In this mode, the far point of the scene is determined by sampling the Z buffer. This is then used to modify the forward and sideways motion obtained from the widgets.

*ViewpointVelocity = K*WidgetVel/(farPoint)*

4) Average depth scaling: In this mode, a geometric average depth in the scene is determined by sampling the Z buffer. This is then used to modify the forward and sideways motion. Any points that sample the far clipping plane (because the scene does not fill the entire screen) are not counted in the average.

*ViewpointVelocity = K*WidgetVel/(averageDepth)*

5) Average depth scaling with hot spot: In this mode, the velocity is modulated using a weighted average of the depth samples. The points inside a rectangular region (hot spot) are weighted five times as heavily as surrounding points. This region comprises one quarter of the screen. When moving forward the hot spot is in the center of the screen. When turning the hot spot is moved in the direction of turn. Thus, for example, when turning to the right the hot spot moves toward the right hand edge of the screen by an amount that depends on the rate of turn. The idea behind this is that objects coming into view are likely to be more important to the viewer and they should therefore be weighted more heavily.

*ViewpointVelocity=K*WidgetVel/(weightedAverageDepth)*

2.1 Evaluation: Searching for Small Targets

We used an exploration task to evaluate the DMF interface. The goal was to find a number of letters placed on or in specially marked boxes scattered throughout the scene. The landscape on which these boxes were scattered consisted of a section of the floor of Passamoquoddy Bay, between Maine and New Brunswick. The task was designed to require viewpoint navigation at a variety of scales. Because the letters were very small and some of the boxes were small, this task required that users frequently change scale. In the extreme case, the smallest letter was 1/30,000 of the size of the scene. The scale change required from perceiving the entire scene to being able to read the smallest character was approximately 500:1.

In some cases, the target letter was placed on the front of a target box. In other cases, one of the faces of the box was open and the subject was required to fly into the box in order to find the target that was placed on the inside far wall. This usually required the subject to orbit around the target box looking for the open face before moving in towards the target. Some of the target boxes were very small and placed on the top of other boxes to help the user locate them from a distance.

2.2 Procedure

Fourteen subjects, all undergraduate or graduate students, were trained. They were first shown the interface, and then guided them through a navigation task. They were required to use each of the different velocity modulation schemes during training.

Each experimental trial involved the subjects finding three target digits on three different boxes that were randomly placed in the scene. When the subject had flown in close enough to actually read a digit, they entered it on the keyboard and proceeded to the next box. When they entered the digit, the target box on which the letter was placed changed color to black, so that the subject would not accidentally look at the same box again.

A trial block consisted of a subject finding three targets in a scene in arbitrary sequence, using one of the five velocity scaling methods. Each subject was tested in each of the five conditions in a random order. Following this the whole test sequence was repeated so that each condition was tested twice.

There were two dependent variables. On each trial the time to find the target was recorded. After each condition, subjects were asked to rate the interface on general ease of use using a seven point scale with 0 representing "hopeless" and 6 representing "excellent". Subjects were also encouraged to comment on the interface as they performed the task.

3 RESULTS AND DISCUSSION

The results are summarized in Figure 2. They show that the near point and the two average distance DMF methods are clearly superior both on user ratings and on objective time to perform the task. In the top graph, the time data has been normalized by dividing by the mean for each subject to remove effects of individual differences. The typical time to perform a single search was approximately 90 seconds. The interface that used average depth with a hot spot was also effective but no better than the uniformly weighted average depth method. The subjective user rating results (shown in the lower graph) are more consistent than the objective results. All subjects rated the far point and constant interfaces as inferior by a large margin.

The DMF interface is intended as an alternative to another flying interface that we developed a number of years ago. This previous interface is based on a 6 degree-of-freedom input device we called the Bat [8,9]. Both flying interfaces are used for data exploration and as a virtual camera controls for making "fly-by" movies of scientific data. In our system, making a movie typically involves flying a path through the scene at low resolution and then playing the flight back one frame at a time in high resolution to create a single frame animation. Our experience with making these kinds of videos suggests that the new DMF interface will be superior to the previous one that was based on the Bat.

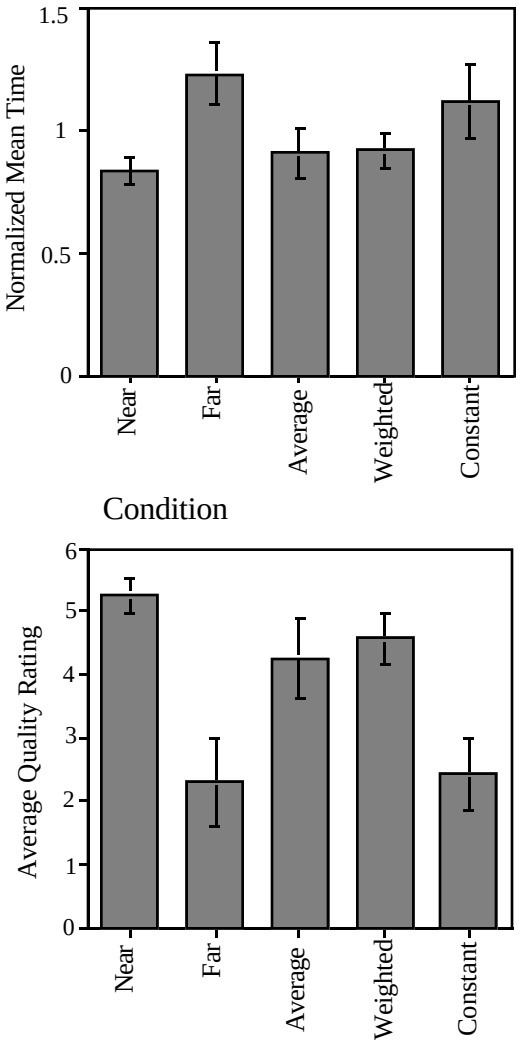

Figure 2. The top graph shows the averge of the normalized times take to perform the search task. The lower graph shows the mean ratings of the quality of the interface. Vertical bars represent two standard errors above and below the mean.

For data exploration, our DMF mouse-based flight interface has the practical advantage that it is easier to learn initially than the Bat flying interface. However, it lacks overall flexibility because it is only possible to manipulate at most two degrees-of-freedom at any time. For the experienced user, the Bat velocity control interface allows the user to rapidly adopt any arbitrary view of the data. However, DMF features can easily be incorporated into any flying interface, and we plan to experiment with such additions to the Bat interface. The DMF interface could also be combined with a POV navigation technique [5], allowing the user to either use the POV method, by selecting a point on the surface, or use the 3D widgets to create free-form flight paths. The only difficulty with this is likely to be the problem of overloading mouse buttons.

Overall, our impression is that modulating velocity based on depth sampling creates a markedly improved exploration interface, especially where large changes of scale are involved. However, people attempting to implement this technique should be warned that there can be technical problems. Z-buffer sampling is slow on certain machines although it is part of the OpenGL standard. Using GL tends to be much faster on older SGI machines. The way in which depth information is encoded in the Z-buffer may also vary from machine to machine. There is a problem sampling scenes that contain fine point or line features that are isolated in space. If the scene contains such features, they will only be sampled occasionally, leading to rapid changes in the depth scaling factor unless full-scene depth sampling can be achieved.

5 REFERENCES

1. Bartram, L., Ho, A., Dill, J and Henigman, F. (1995) The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Spaces. ACM UIST'95 Proceedings 207-216.
2. Bederson, B.B. and Hollan, J.D., 1994 PAD++: zooming graphical interface for exploring alternate interface physics.UIST'94 Proceedings (Marina Del Ray, CA) ACM Press 17-26.
3. Drucker, S.M. and Zeltzer, D. (1994) Intelligent Camera Control in a Virtual Environment. Graphics Interface, Proceedings, 190-199.
4. Hayes, N. (1993) Nobody Walks in VR-They all Fly, IEEE Computer Graphics and Applications. 13(3) May. p 85.
5. Mackinlay, J.D., Card, S.K., and Robertson, GG. "Rapid Controlled Movement Through a Virtual 3D Workspace. Proceedings of SIGGRAPH'90 (Dallas, Texas, August 1990). In Computer Graphics, 24, 3, 171-176.
6. Masui, T, Minakuchi, Borden, G.R. and Kashiwagi, K, 1995 Multi-View Approach for Smooth Information Retreival. V. ACM UIST'95 Proceedings199-206
7. Robinett, W and Holloway, R. (1992) Implementation of Flying, Scaling and Grabbing in Virtual Worlds, 1992 Symposium on Interactive 3D Graphics. Special Issue of Computer Graphics, ACM Press.
8. Ware, C., and Osborne, S., (1990) Exploration and virtual camera control in virtual three dimensional environemnts. Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah, March 1990). In Computer Graphics 24, 2, 175-183.
9. Ware, C., and Slipp, L. (1991) Using Velocity Control to Navigate 3D Graphical Environments: A comparison of Three Interfaces, Proceedings of Human Factors Society Meeting San-Francisco, September. Proceedings, 300-304.
10. Ware, C. (1996) Moving Motion Metaphors, ACM CHI'96 Conference Companion. 225-226.
11. Ware, C. (1995) Dynamic Stereo Displays. ACM CHI'96 Conference Proceedings, Denver, 310-316.
12. Ware. C. (1995) The DragMag Image Magnifier, ACM CHI'95 Conference Companion. 407-409. + CHI'95 Video Program.

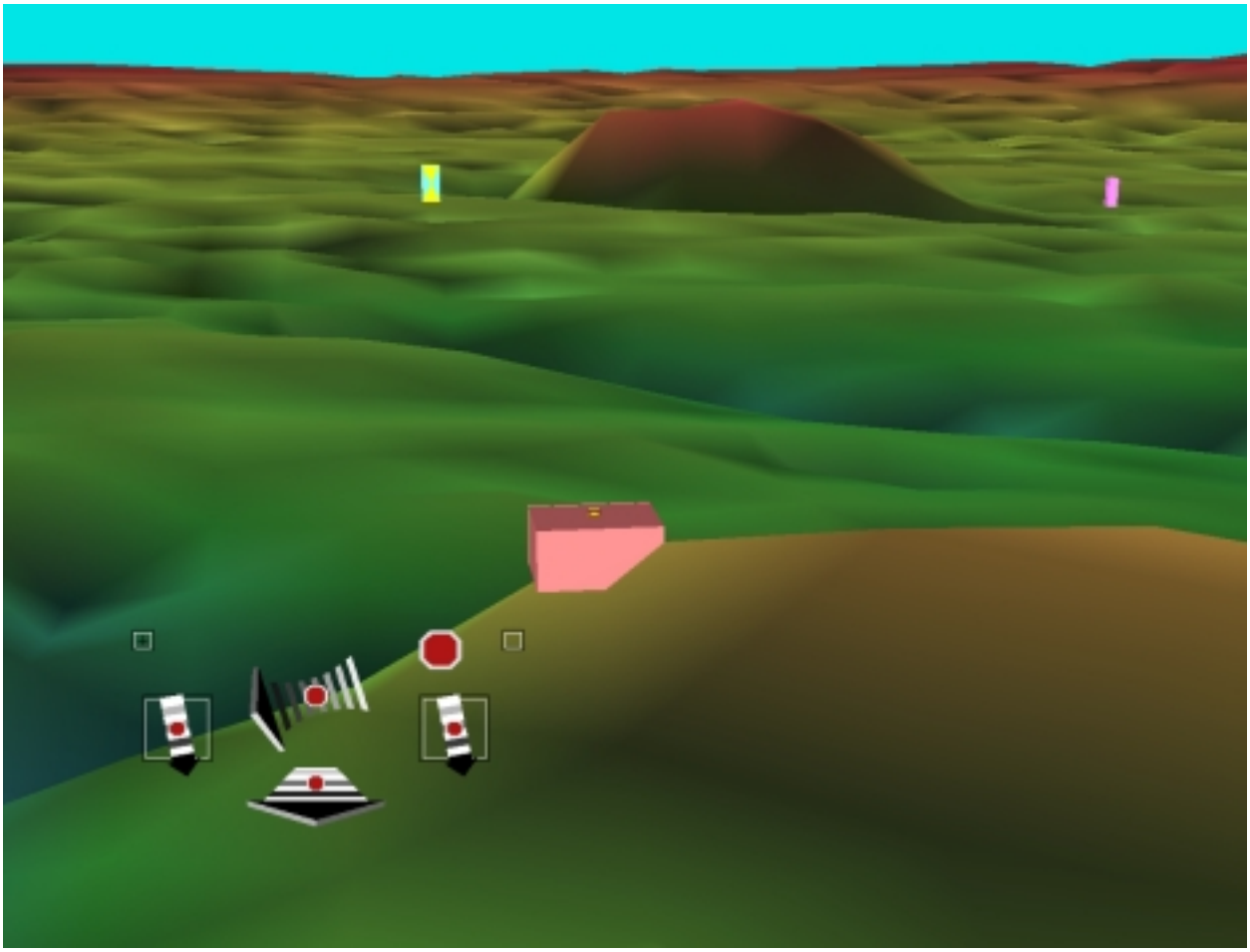

Plate 1. The widgets on the lower left are used in the flying interface. The Blocks scattered over the terrain each contain a very small hidden letter. The task was to find these letters.