Automatic Viewing Control for 3D Direct Manipulation

Cary B. Phillips[†]
Norman I. Badler
John Granieri

Computer Graphics Research Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6389

Abstract

This paper describes a technique for augmenting the process of 3D direct manipulation by automatically finding an effective placement for the virtual camera. Many of the best techniques for direct manipulation of 3D geometric objects are sensitive to the angle of view, and can thus require that the user coordinate the placement of the viewpoint during the manipulation process. In some cases, this process can be automated. This means that the system can automatically avoid degenerate situations in which translations and rotations are difficult to perform. The system can also select viewpoints and viewing angles which make the object being manipulated visible, ensuring that it is not obstructed by other objects.

Introduction

3D direct manipulation is a technique for controlling positions and orientations of geometric objects in a 3D environment in a non-numerical, visual way. Although much research has been devoted to 3D direct manipulation of geometric objects, no existing system has adequately integrated the controls for viewing into the direct manipulation process. Evans, Tanner, and Wein [3], Nielson and Olson[6], and Chen et al [1] all discuss techniques for manipulation that are sensitive to the viewing direction, but they do not address how the view can be manipulated. Ware and Osborne[10] discuss the viewing process in general, in terms of metaphors that it suggests, and Mackinlay et al [5] discuss an effective technique for manipulating the viewpoint, both in proximity to other objects and through large distances. Neither of these relate the viewing process to direct manipulation.

Our direct manipulation system includes a mechanism for automatically placing the virtual camera at a viewpoint which avoids the problems with degenerate axes suffered by most direct manipulation schemes. The basic idea is to rotate the camera through small angles to achieve a better view. Our system also rotates the camera to avoid viewing obstructions. This viewing operation is an integral part of the manipulation system, not a separate viewing facility which the user must explicitly invoke.

The problem of automatic viewing placement for manipulation is different from that of automatic camera control in animation. Karp and Feiner[4] describe a system called ESPLANADE that automatically visualizes simulations. It automatically finds camera placements which provide a good view of movement during an animation. This is an adjunct to the process of animation, not an interactive technique.

3D Direct Manipulation

Several techniques have been developed for describing three dimensional transformations with a two dimensional input device such as a mouse or tablet. Nielson and Olson [6] describe a technique for mapping the motion of a two dimensional mouse cursor to three dimensional translations based on the orientation of the projection of a world space coordinate triad onto the screen. This mapping makes it difficult to translate along an axis parallel to the line of sight, because the

[†]Cary Phillips' current address: Pacific Data Images, 1111 Karlstad Dr, Sunnyvale, CA 94089

axis projects onto a point on the screen instead of a direction.

Rotations are considerably more complex, but several techniques have been developed, with varying degrees of success. The most naive technique is to simply use horizontal and vertical mouse movements to control the world space euler angles which define the orientation of an object. This technique provides little kinesthetic feedback because there is no natural correspondence between the movements of the mouse and the rotation of the object. A better approach, described by Chen et al [1], is to make the rotation angles either parallel or perpendicular to the viewing direction. This makes the object rotate relative to the graphics window, providing much greater kinesthetic feedback, but it also makes the available rotation axes highly dependent on the viewing direction.

3D Manipulation in *Jack*

Our interactive system is called *Jack*™†, and it is designed for modeling, manipulating, animating, and analyzing human figures, principally for human factors analysis. The 3D direct manipulation facility in *Jack* allows the user to interactively manipulate figure positions and orientations, and joint angles subject to limits[7]. *Jack* also has a sophisticated system of manipulating postures through inverse kinematics and behavior functions [8, 9]. *Jack* runs on Silicon Graphics IRIS workstations, and it uses a three button mouse to control translation and rotation. Within the direct manipulation process, the user can toggle between rotation and translation, and between the local and global coordinate axes, by holding down the `CONTROL` and `SHIFT` keys, respectively.

With translation, the user controls the movement by moving the mouse cursor along the line which the selected axis makes on the screen. This is similar to the projected triad scheme of Nielson and Olson[6], and it ensures good kinesthetic correspondence. Pairs of buttons select pairs of axes and translate in a plane. A 3D graphical translation icon located at the origin of the object being manipulated illustrates the selected axes and the enabled directions of motion.

The user can control rotation around the x, y, and z axes, in either local or global coordinates. Only one axis can be selected at a time. A graphical wheel icon illustrates the origin and direction of the axis. The user controls the rotation by moving the cursor around the perimeter of the rotation wheel, causing the object to rotate around the axis. This is analogous to turning a crank by grabbing the perimeter and dragging it in circles. This is somewhat similar to Evans, Tanner and Wein's turntable technique[3], but it provides greater graphical feedback.

Drawbacks

A drawback of the manipulation technique in *Jack* is the inability to translate an object along an axis parallel to the line of sight, or to rotate around an axis perpendicular to the line of sight. In these cases, small differences in the screen coordinates of the mouse correspond to large distances in world coordinates, which means that the object may spin suddenly or zoom off to infinity. This is an intrinsic problem with viewing through a 2D projection: kinesthetic correspondence dictates that the object's image moves in coordination with the input device, but if the object's movement is parallel to the line of projection, the image doesn't actually move, it only shrinks or expands in perspective.

In the past, we adopted the view that the first prerequisite for manipulating a figure is to position the camera in a convenient view. Although the viewpoint manipulation techniques in *Jack* are quite easy to use, this forced the user through additional step in the manipulation process, and the user frequently moved back and forth between manipulating the object and camera.

3D Viewing

The computer graphics workstation provides a view into a virtual 3D world. It is natural to think of a graphics window as the lens of a camera, so the process of manipulating the viewpoint is analogous to moving a camera through space. Evans, Tanner, and Wein describe viewing rotation as the single most effective depth cue, even better than stereoscopy [3]. In order for an interactive modeling system to give the user a good sense of the three-dimensionality of the objects, it is essential that the system provide a good means of controlling the viewpoint.

Control over the viewpoint is especially important during the direct manipulation process, because of the need to "see what you are doing." The whole notion of direct manipulation requires that the user see what is happening, and feel the relationship to the movement of the input devices. If the user can't see the object, then he or she certainly can't manipulate it properly.

Jack uses Ware and Osborne's *camera in hand* metaphor[10] for the view. The geometric environment in problems in human factors analysis usually involve models of human figures in a simulated workspace. The most appropriate cognitive model to promote is one of looking in on a real person interacting with real, life-size objects. Therefore, *Jack* suggests that the controls on the viewing mechanism more or less match the controls we have as real observers: move side to side and up and

† *Jack* is a trademark of the University of Pennsylvania.

down while staying focused on the same point.

The viewing adjustments in *Jack* are easy to invoke from within the direct manipulation process, and this is a very common thing to do. The typical way of performing a manipulation is to intersperse translations and rotations with viewing adjustments, in order to achieve a better view during the process. The context switch between viewing and manipulation is very easy to make.

Automatic Viewing Adjustments

Much of this viewing adjustment as an aid to manipulation can be automated, in which case the system automatically places the camera in a view which avoids the problems of degenerate axes. This can usually be done with a small rotation to move the camera away from the offending axis. This automatic camera rotation can even be helpful by itself, because it provides a kind of depth cue.

To prevent degenerate movement axes from causing problems during direct manipulation, *Jack* uses a threshold between the movement axis and the line of sight, beyond which it will not allow the user to manipulate an object. To do so would mean that small movements of the mouse would result in huge translations or rotations of the object. This value is usually 20°, implying that if the user tries to translate along an axis which is closer than 20° to the line of sight, *Jack* will respond with a message saying "can't translate along that axis from this view," and it will not allow the user to do it. The same applies to rotation around axes perpendicular to the line of sight. In these cases, the rotation wheel projects onto a line, so the user has no leverage to rotate it.

The automatic viewing adjustment invokes itself if the user selects the same axis again after getting the warning message. *Jack* will automatically rotate the camera so that its line of sight is away from the transformation axis. To do this, it orients the camera so that it focuses on the object's origin, and then rotates the camera around both a horizontal and a vertical axis, both of which pass through the object's origin. The angles of rotation are computed so that the angular distance away from the offending axis is at least 20°.

This technique maintains the same distance between the camera and the object being manipulated. In general, this "zoom factor" is much more subjective and is difficult for the system to predict. In practice, we have found it best to require the user to control this quantity explicitly.

The reason for the repeated axis selection is to ensure that the user didn't select the axis by mistake. It is common to position the view parallel to a coordinate axis to get a 2D view of an object. If the user likes this view, then it would be wrong to disturb it. For example, if the user positions the view parallel to the z axis to get a view of the xy plane, and then accidentally hits the right mouse button, the view will not automatically change unless the user confirms that this is what he or she wants to do.

Automatic view positioning also takes place when the object is not visible. This may mean that the object is not visible at all, or only that its origin is not visible. For example, a human figure may be mostly visible but with its foot off the bottom of the screen. In this case, a command to move the foot will automatically reposition the view so that the foot is visible.

Smooth Viewing Transitions

Both the horizontal and vertical automatic viewing rotations occur simultaneously, and *Jack* applies them incrementally using a number of intermediate views so the user sees a smooth transition from the original view to the new. This avoids a disconcerting snap in the view. *Jack* applies the angular changes using an ease in/ease out function which ensures that the transition is smooth.

The procedure for rotating the camera is sensitive to the interactive frame rate so that it provides relatively constant response time. If the camera adjustment were to use a constant number of intermediate frames, the response time would be either too short if the rate is fast or too long if the rate is slow. *Jack* keeps track of the frame rate using timing information available from the operating system in 1/60th's of seconds. We compute the number of necessary intermediate frames so that the automatic viewing adjustment takes about 1 second of real time.

Avoiding Viewing Obstructions

When manipulating an object using solid shaded graphics, it can be especially difficult to see what your are doing because of the inability to see through other objects. In some situations, this may be impossible to avoid, in which case the only alternative is either to proceed without good visibility or revert to a wireframe image. Frequently however, it may be possible to automatically change the view slightly so that the object is less obstructed. To do this, we borrow an approach from radiosity, the *hemicube* [2].

The hemicube determines the visibility of an entire geometric environment from a particular reference point, and we can use this information to find an unobstructed location for the camera if one exists. We perform the hemicube computation centered around the origin of the object being manipulated, but oriented towards the current camera location. This yields a visibility map of the entire environment, or what we would see

through a fish-eye lens looking from the object's origin towards the camera. If the camera is obstructed in the visibility map, we look in the neighborhood of the direction of the camera for an empty area in the hemicube map. This area suggests a location of the camera from which the object will be visible. From this, we compute the angles through which the camera should be rotated. We generate the hemicube map using the hardware shading and z-buffer, so its computation is quite efficient.

This type of hemicube is somewhat different from the type used radiosity because it is not necessarily centered around the surface of an object. In fact, it need not be associated with a surface at all, as when the direct manipulation operation is applied to a shapeless entity like a 3D control point or a goal point for an inverse kinematics operation. Therefore, our hemi-cube is actually not "hemi" at all, since we use all six sides of the cube. In cases when the direct manipulation operation is moving a geometric object, it is convenient to omit the object from the hemicube visibility computation altogether. Otherwise, most of the visibility map will be filled up with the object itself, even though it is usually quite acceptable to manipulate an object from a view opposite its coordinate origin.

In our current implementation, the hemicube maintains only occlusion information, not depth information. Therefore, it will fail to find suitable camera locations in an enclosed environment. In such cases, there are no holes in the visibility map at all, although there may be regions only occluded by very distance objects. These very distant objects don't matter unless we were considering placing the camera very far away. A better approach would be to retain depth information in the hemicube and search for a camera position which is unobstructed only between the camera and the object, allowing the distance between the object and the camera change as necessary, possibly causing the camera to move in front of other objects.

Conclusion

The control of a virtual camera is vitally important to many techniques for 3D direct manipulation system, although no one has previously addressed the two issues in an integrated manner. Our technique for automatically adjusting the view in conjunction with direct manipulation has been implemented, and it is an effective addition to the manipulation process. The automatic viewing rotations are usually very small so they do not interject large changes to the user's view of the geometric environment. Since the viewing adjustments are only activated on the second attempt at movement along a degenerate axis, the adjustments are seldomly invoked accidentally, minimizing the degree to which the adjustments are inappropriate.

References

[1] Michael Chen, S. Joy Mountford, and Abigail Sellen. A Study in Interactive 3-D Rotation Using 2-D Control Devices. *Computer Graphics*, 22(4), 1988.

[2] Michael F. Cohen and Donald P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics*, 19(3), 1985.

[3] Kenneth B. Evans, Peter Tanner, and Marceli Wein. Tablet Based Valuators That Provide One, Two or Three Degrees of Freedom. *Computer Graphics*, 15(3), 1981.

[4] P. Karp and S. Feiner. Issues in the Automated Generation of Animated Presentations. In *Proceedings of Graphics Interface '90*, 1990.

[5] Jock D. Mackinlay, Stuart K. Card, and George G. Robinson. Rapid and Controlled Movement Through a Virtual 3D Workspace. *Computer Graphics*, 24(4), 1990.

[6] Gregory Nielson and Dan Olsen Jr. Direct Manipulation Techniques for 3D Objects Using 2D Locator Devices. In *Proceedings of 1986 Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October 1987. ACM.

[7] Cary B. Phillips and Norman I. Badler. Jack: A Toolkit for Manipulating Articulated Figures. In *Proceedings of ACM SIGGRAPH Symposium on User Interface Software*, Banff, Alberta, Canada, 1988.

[8] Cary B. Phillips and Norman I. Badler. Interactive Behaviors for Bipedal Articulated Figures. *Computer Graphics*, 25(4), 1991.

[9] Cary B. Phillips, Jianmin Zhao, and Norman I. Badler. Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints. *Computer Graphics*, 24(2), 1990.

[10] Colin Ware and Steven Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Computer Graphics*, 24(4), 1990.