Report Number: WUCSE-2004-32

2004-06-18

The IBar: A Perspective-based Camera Widget

Cindy Grimm, Karan Singh, and Nisha Sudarsanan

We present a new widget, the IBar, for controlling all as-pects of a perspective camera. This widget provides an in-tuitive interface for controlling the perspective distortion in the scene by providing single handles that manipulate one or more projection parameters simultaneously (e.g., distance-to-object and lens aperture) in order to create a single per-ceived projection change (increasing the perspective distor-tion without changing the scene size). We demonstrate that novice users more easily learn how to manipulate the camera using the IBar.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2004-32

The IBar: A Perspective-based Camera Widget

Authors: Grimm, Cindy; Singh, Karan; Sudarsanan, Nisha

June 18, 2004

Abstract: We present a new widget, the IBar, for controlling all aspects of a perspective camera. This widget provides an intuitive interface for controlling the perspective distortion in the scene by providing single handles that manipulate one or more projection parameters simultaneously (\eg distance-to-object and lens aperture) in order to create a single perceived projection change (increasing the perspective distortion without changing the scene size). We demonstrate that novice users more easily learn how to manipulate the camera using the IBar.

WUCSE-2004-32 The IBar: A Perspective-based Camera Widget

Category: Research

Figure 1: Changing the perspective distortion of the scene.

Abstract

We present a new widget, the IBar, for controlling all aspects of a perspective camera. This widget provides an intuitive interface for controlling the perspective distortion in the scene by providing single handles that manipulate one or more projection parameters simultaneously (*e.g.*, distance-to-object and lens aperture) in order to create a single perceived projection change (increasing the perspective distortion without changing the scene size). We demonstrate that novice users more easily learn how to manipulate the camera using the IBar.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling

Keywords: Camera control, Projection, Perspective, Rendering

1 Introduction

Camera control for 3D rendering is a difficult problem. A full perspective matrix [Michener and Carlbom 1980] has 11 degrees of freedom — 6 to control the position and orientation of the camera, and 5 to control the projection. Specifying a perspective matrix with just a mouse and a keyboard can be a challenging task. In this paper we present a single screen-space widget that provides intuitive manipulation of *all* of the camera using just the mouse (with optional key modifiers). This widget changes pairs of parameters simultaneously (where appropriate) in order to present the user with more intuitive controls.

Part of the difficulty of camera control is the complex inter-relationships between the 3D objects in the scene, the 3D position and orientation of the camera, the internal camera parameters, and the final 2D scene layout. All of this information is communicated to the user *through the perspective rendering itself*. This places a heavy cognitive burden on the user, since they must build up a mental model of the 3D scene and the camera, along with a mental model of how changing camera parameters affects the perspective rendering. To help users clarify the 3D relationships, many systems provide additional renderings (usually the X, Y, Z axes), in which a representation of the camera is displayed along with the scene. The user can then manipulate this representation to position and orient the camera, and even change internal camera parameters (see Figure 2). Although this greatly facilitates placing the camera, it does little to illuminate the 2D qualities of the perspective projection itself.

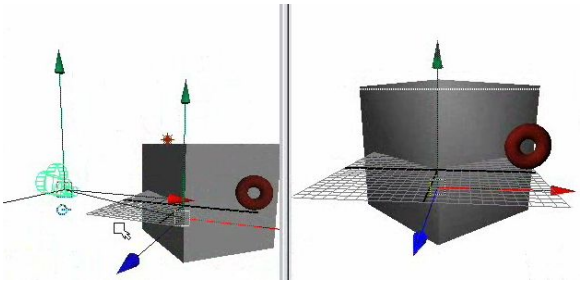


Figure 2: A traditional Computer Graphics approach to specifying a camera. Left: The camera in relationship to the scene. Right: The perspective view. (Program shown is Maya).

For mathematicians, and most of the Computer Graphics community, perspective projection is simply a 4×4 matrix that projects a 3D scene into 2D, taking straight lines to straight lines in the image plane and maintaining the depth ordering. Artists, however, have a much more complex vocabulary that *qualitatively* describes perspective projection — they are primarily concerned with describing the visual features of the projection in the 2D plane [Cole 1976; Carlbom and Paciorek 1978]. Figure 3 illustrates some of these features. The terms 1,2, and 3 point perspective refer to the number of vanishing points defined in the image; note that this depends on the camera's positional relationship to objects in the scene. The left and right vanishing points define a horizon or eye line; changing the location of this line can dramatically change how the scene is perceived. When sketching out a scene, artists simplify the objects in the scene, reduc-

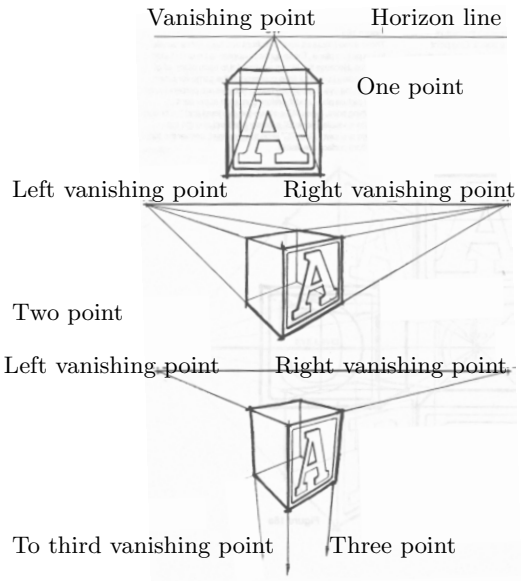


Figure 3: Terms used by artists to describe perspective projections. From: *Perspective Drawing and Applications.*

ing them to collections of lines, points and simple curves. This allows them to visualize the primary vanishing points, lines-of-sight, and horizon lines in the 2D plane.

Traditional camera manipulation techniques do not support this type of visualization — they instead support the photographer's approach to exploring the projection space. A skilled photographer learns to "see" through the lens of the camera, flattening out the scene in their mind's eye and evaluating it for its 2D aesthetics. Current graphics systems allow the user to manipulate the camera as if it were held in the hand. In this model, widening (or shrinking) the field of view makes the objects in the scene smaller (or bigger). Moving closer (or further) to the object does exactly the same thing — except that it also changes the perspective, *i.e.*, , the vanishing points move.

In this paper we propose an alternative approach to the camera specification problem that is more closely aligned with the artist's concept of perspective. We place a single screen-space widget, called the *IBar*, into the image. The shape of the IBar provides information about the current vanishing points and horizon lines, and allows the artist to manipulate those entities directly, rather than indirectly through moving the camera, changing the field of view, *etc.*

Contributions: The IBar provides a natural interface for manipulating the parameters of the camera that influence perspective distortion. This supports dramatic camera affects (Figure 1) that are currently difficult to specify because they require simultaneous editing of several camera parameters.

We cover existing camera models first (Section 2). The functionality of the IBar is described in Section 3. The corresponding equations are given in Section 5. We performed a small, informal user study; the results are discussed in Section 4.

2 Related work

For mouse-based systems, camera control paradigms fall roughly into two categories, camera-centric and object-centric. In the camera-centric paradigm, operations are applied to the camera as if it were a real object in the scene. This mirrors camera placement in the real world, and many of the camera operations (dolly in, pan, *etc.*) reflect that. The external parameters, position and orientation, can be specified either "through the lens", or by manipulating a pictorial representation of the camera in a second window. The internal camera parameters, with the exception of focal length, are changed through textual input.

In the object-centric paradigm, the camera is centered on an object and the viewpoint is rotated relative to the object (as if there were a virtual trackball around the object [Hultquist 1990]). The camera can also be zoomed in and out. This paradigm is useful when there is a single object in the scene (or one object of importance) and the user is simply choosing a direction from which to view it.

Three or six degrees of freedom devices permit other interesting navigation techniques [Bowman et al. 1997], such as the palm-top world [Stoakley et al. 1995], the "grab and pull" approach [Poupyrev et al. 1996] and virtual fly-throughs [Wloka and Greenfield 1995]. The latter can also be used in mouse or keyboard-based systems if the camera's movement is restricted to a well-defined floor plane (most first-person shooters use this approach).

An alternative approach to directly specifying the camera is to use image-space constraints [Blinn 1988; Gleicher and Witkin 1992]. In this approach, points in the scene are constrained to appear at particular locations, or to move in a specified direction, and the system solves for the camera parameters that meet those constraints. The IBar is, in some sense, a specialization of the constraint approach, where the points are the points of the cube. However, unlike the constraint approach, changes to the IBar result in well-defined changes to the camera parameters. This provides more precise control and repeatability at the cost of generality.

3 The IBar Widget

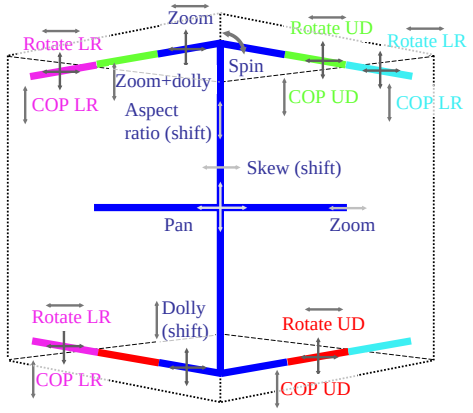


Figure 4: A schematic of the IBar widget. Arrows mark handle locations and available movement directions. Moving the first third of any IBar limb moves all four limbs simultaneously. Moving the second third of the top (or bottom) limb moves both top (or bottom) limbs simultaneously. Moving the last third of the left (or right) limb moves both the left (or right) limbs simultaneously. Moving the mouse left-right scales the length of the limb, moving up-down changes the angle.

A schematic diagram of the IBar widget is shown in Figure 4. Conceptually, the IBar represents the two-point per-

spective rendering of a cube centered on the `Look` vector of the camera. The IBar is inspired by the use or vanishing points to control perspective; changing the IBar indicates the desired change to the perspective rendering of the cube. The internal parameters of the camera (center of projection, focal length) are reflected in the shape of the IBar. Except for when the IBar is being moved, it always appears in the middle of the screen at a constant size (one-half of the screen height).

Moving or rotating the entire IBar corresponds to moving or rotating the camera; the exact behavior depends on whether or not the user wishes to use the IBar in camera or object-centric mode. In object-centric mode the IBar represents a cube, and changing the shape of the IBar indicates how the cube should be re-drawn. For example, moving the IBar up and to the right moves the center of the scene up and to the right.

In camera-centric mode, the user moves the IBar in the scene to the desired position relative to the scene, as shown in the current rendering. The IBar then snaps back to its default position and orientation, dragging the scene with it.

Changing the angles and lengths of the limbs corresponds to moving or changing the vanishing points. This causes the camera to move (rotation around the cube or dolly-in), change focal length, move the center of projection, or some combination thereof. To simplify symmetric changes, different parts of the limbs change either two or four of the limbs simultaneously. The size of the limbs is changed by left-right mouse movement, the angles by up-down movement. The IBar always snaps back to the center of the screen after the end of a manipulation.

3.1 Visual cues

The angles of the limbs provide information about the vanishing points of the rendering. The relative differences in the limb angles indicate in which direction the center of projection has been shifted; if all of the limb angles are the same size, then the center of projection is in the middle of the screen. The absolute angles of the limbs indicate where the vanishing points are — this is a combination of the distance of the cube from the camera and the focal length. The horizon line can also be explicitly indicated by the placement of the horizontal bar (Section 3.6).

The IBar represents a unit cube at a distance d from the camera. If the user has specified a focus distance [1] then the cube will be placed at that distance. Alternatively, the user can select a point in the scene to define the focus distance.

To keep the projected size of the cube constant on the screen we scale the width and height (but not the depth) by:

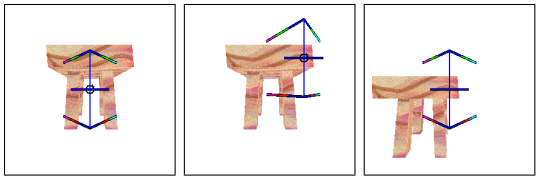
$$s = dH/f \quad (1)$$

where f is the focal length. The limbs of the IBar are the projection of the adjacent cube edges.

3.2 Screen-space position and orientation

We begin by describing the manipulations that change the position and orientation of the cube in the image plane. The mouse movements and widget handles are identical for both the camera- and object-centric manipulations, but the behavior is different.

Camera-centric:



Object-centric:

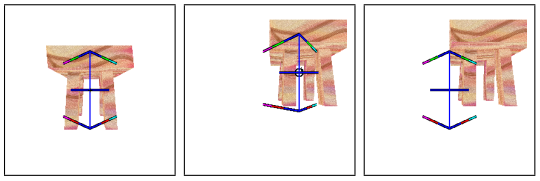
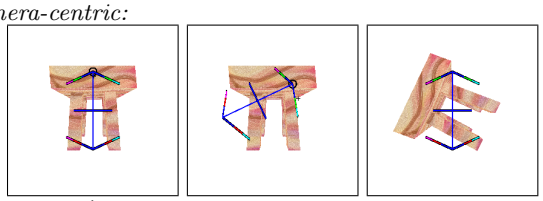


Figure 5. **Pan:** Move the IBar using the handle at the center.

Camera-centric:



Object-centric:

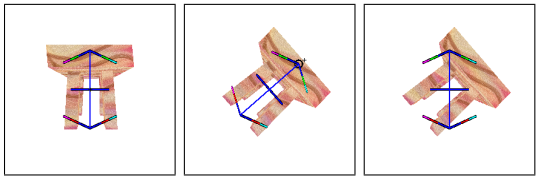


Figure 6: **Camera spin:** To rotate the camera about its `Look` vector, rotate the IBar using the top (or bottom) of the stem.

3.3 Rotation

These two operations support the traditional virtual trackball [Hultquist 1990] camera manipulation. The rotation point is the center of the cube; this point can be tied to an object if desired (see above). Because the IBar snaps back after every manipulation, the object can be rotated through all 360 degrees.

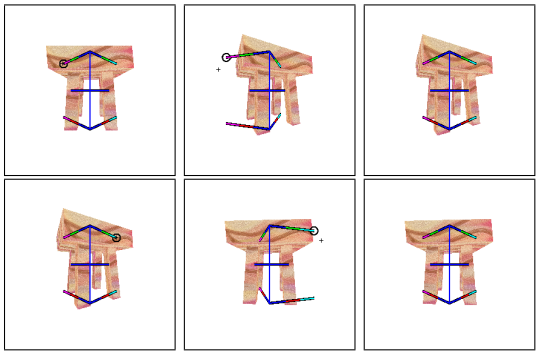


Figure 7: **Camera rotate left-right:** To rotate the camera left or right, scale the appropriate side limbs. Lengthening the right limbs is equivalent to shortening the left limbs.

[1] The focus distance is used to specify a depth of focus; it does not affect the perspective matrix.

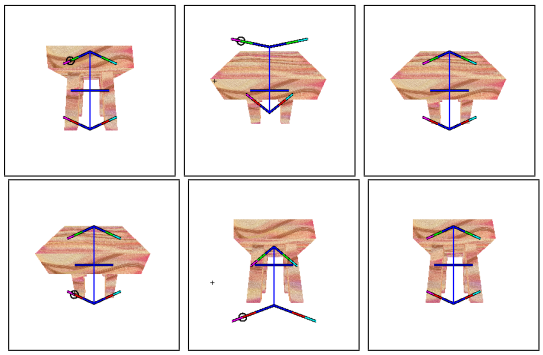
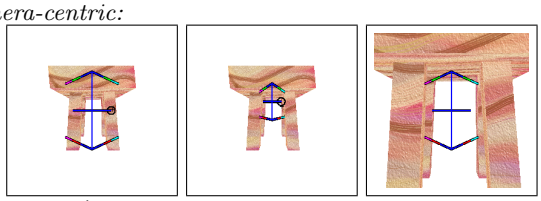


Figure 8: **Camera rotate up-down:** To rotate the camera into or out of the film plane, scale the appropriate top or bottom limbs. Lengthening the top limbs is equivalent to shortening the bottom limbs.

3.4 Zoom and dolly-in

These operations change the size of the rendered objects, and, optionally, the perspective distortion.

Camera-centric:



Object-centric:

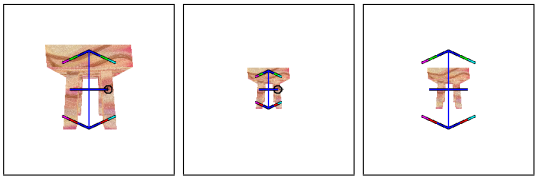


Figure 9: **Zoom in/out:** To zoom the camera in and out without changing the perspective distortion, scale the middle of the IBar.

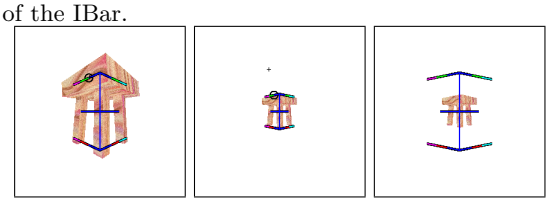


Figure 10: **Dolly in/out:** To dolly the camera in/out without affecting the focal length, change the angles on all four limbs simultaneously while holding the shift key.

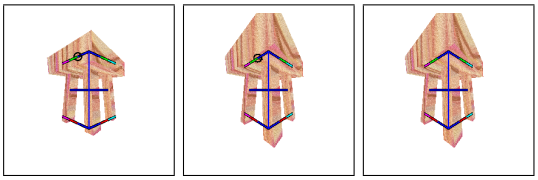
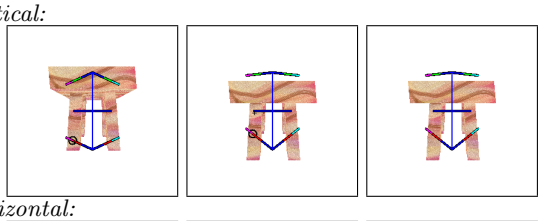


Figure 11: **Dolly in/out with zoom:** To dolly the camera in/out and simultaneously change the focal length, change the angles on all four limbs simultaneously.

3.5 Internal Camera Parameters

There are 5 internal camera parameters; center of projection (2), focal length, skew, and aspect ratio. Focal length was discussed earlier in conjunction with dolly in. Aspect ratio changes the ratio of the height to the width. Skew essentially performs a shear in the image plane.

Vertical:



Horizontal:

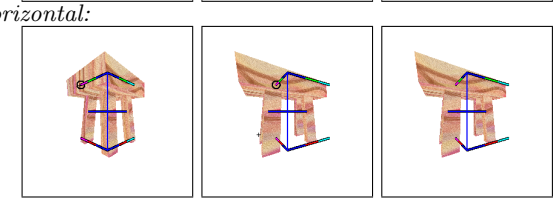


Figure 12: **COP:** To change the center of projection, make the angles of the top limbs different from the bottom ones (moves the center of projection up/down). Similarly, making the angles of the left limbs different from the right moves the center of projection left-right.

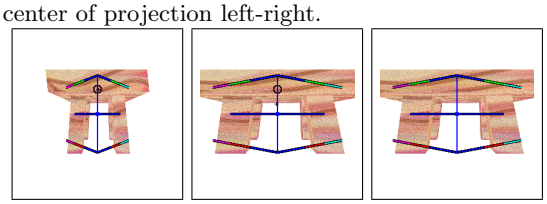


Figure 13: **Aspect ratio:** To change the aspect ratio, grab a point on the IBar stem and move up-down while holding the shift key.

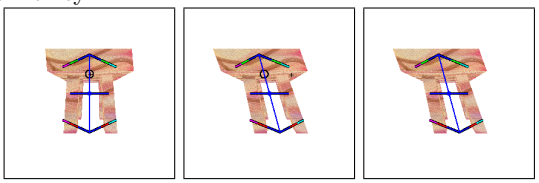


Figure 14: **Skew:** To change the skew, grab a point on the IBar stem and move left-right while holding the shift key.

3.6 Options

The IBar can be extended in a couple of ways. First, the horizontal bar can be placed to indicate the horizon line. Second, the IBar can be placed at a given point in the scene, allowing the user to both visualize the perspective distortion at that point, and to rotate the camera around an arbitrary point in the scene.

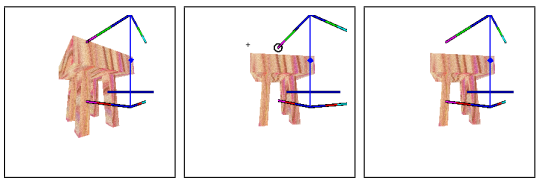


Figure 15. **Object rotate:** Rotating the IBar around an arbitrary point in the scene.

Third, there are several possible methods for switching between camera- and object-based approaches. Option one is to use a toggle switch. Option two is to use a key-modifier such as the control key. Option three is to take advantage of the multiple handles for each camera operation. For example, there are two zoom handles (left and right). We can map the left handle to the camera-centric zoom and the right handle to the object-centric version. Similarly, we can map all of the top limbs to camera-centric and all of the bottom limbs to object-centric. This has the advantage of eliminating modes, but it does increase the number of distinct

	Maya first		IBar first	
Interface	Learning	View	Learning	View
Maya avg.	11	8.2	7	6.53
Maya SD.	2.2	2.5	2.7	3.8
IBar avg.	12	6.3	7.6	8.6
IBar SD.	5.7	2.9	3.7	5.7

Table 1: Results of the user study, 10 participants. All times are in minutes.

handles.

Finally, the shift-key can be used to constrain the interaction in one of two ways. We currently use the shift-key to select the less-common camera interaction (see Figure 4). The movement of the limb is constrained to be either vertical or horizontal, depending on the direction the user first moves. Both directions are enabled by holding down the shift key.

A second option is to use the shift key to constrain the motion, and allow simultaneous horizontal and vertical changes to the limbs as the default.

4 User study

We performed a small user study to compare the use of the IBar with a traditional camera interface (Maya). The user group consisted of 10 students, 7 of which had little or no experience with a 3D camera. Each user was randomly assigned to start with either the Maya or the IBar interface. They were given a list of written instructions on how to manipulate the camera and allowed to play with the interface until they were satisfied that they understood how it worked. The scene they practiced with is the same scene we used for the study, a $3x3$ array of colored tables (see Figure 16). We then presented the users with a sequence of 3-5 screen shots [2]. For each screen shot the user manipulated the camera from its default position until they were satisfied that they had matched the screen shot. They then repeated the entire process (learning and matching) for the other manipulation technique.

The data set: We created 20 screen shots, that were one or two manipulations away from the default view. Each user completed two or three different scenes for each interface (for a total of 4 or 6), depending on time. The scenes were all different because we felt that, once a user had reconstructed a view, doing so again would be easier, even if the manipulation techniques were different.

The collected data: We kept track of the amount of time spent learning the interface and how long it took the user to match the screen shot. Each user was asked how often they used a 3D interface, and to rate their knowledge of the mathematics of 3D projection. We also asked users to rank each interaction mode on a scale from one to five; the IBar average score was 3.3, Maya was 2.9.

The results are summarized in Table 1. Our observations are as follows. First, our three participants who ranked themselves as knowledgeable performed, on average, as well as the novice users. (Note that none of the 10 participants had ever used Maya before; our expert users were graphics students who had implemented a camera with a simple key-based interface.) Second, the people who used the IBar first and Maya second spent less time learning both interfaces than the people who started with Maya. This appears to be because the IBar taught them more about the camera transformations. The times for matching screen shots were similar between the two interfaces, with the times being less for whichever interface was used second.

Only three users (two of which had started with the IBar) managed to match one or two more difficult scenes that required more than two camera manipulations. Times ranged from 2 to 20 minutes, and were equivalent for both techniques. The people who were successful were the ones who thought about the transformations, rather than blindly manipulating the widgets. These were also the people who found the IBar more intuitive.

[2] We had originally planned for 10-20 screen shots, but due to the time it took for the subjects to match the scene (approx. 5-15 minutes) we reduced the number.

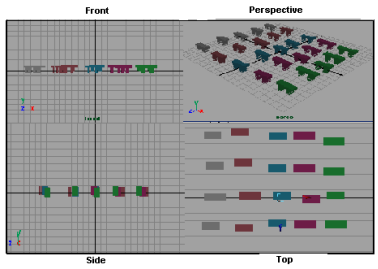
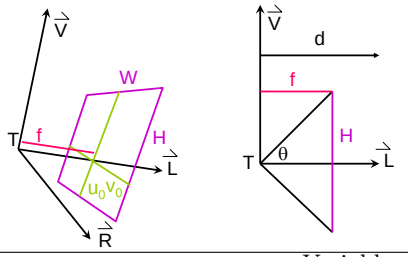


Figure 16: Our test scene, default view.

5 Implementation

In this section we define the equations that correspond to the camera manipulations in the previous section. Our camera parameters are summarized in the following table; the perspective matrix is built from these parameters in the usual way [Michener and Carlbom 1980], for completeness's sake we summarize the matrices in Appendix A. If the user has selected an object to define the focus distance (Section 3.1) then d is the distance from the camera's position T to the object.



Name	Variable
Screen size	W,H
Position	T
Right	$\vec{U} = \vec{V} \times \vec{L}$
Up	$\vec{V}$
Look	$\vec{L}$
Rotation	$R = \left[\vec{U}^T \vec{V}^T \vec{L}^T\right]$
Focus distance	d
Focal length	f
Aperture angle	$\theta = 2\tan^{-1}(H/f)$
Center of projection	(u_0, v_0)
Film plane scale (Eq. 1)	$s = d(H/f)$

Table 2: The camera and its parameters.

5.1 Drawing the IBar

The IBar represents a unit cube with the front edge centered on the `Look` vector and oriented parallel to the `Up` vector. The four adjacent edges extend back in the `Look` and `Right` directions. The x and y directions are scaled to account for the focal length, but the z component is not. The IBar is shifted in the film plane to counter-act any non-zero center-of-projection.

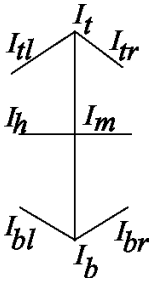
The middle of the IBar is at:

$$I_m = T + d\vec{L} + (su_0)\vec{U} + (sv_0)\vec{V} \quad (2)$$

The top and bottom points of the IBar are at:

$$I_t = I_m + \frac{s}{2}\vec{V} \quad (3)$$

$$I_b = I_m - \frac{s}{2}\vec{V} \quad (4)$$



The endpoints of the horizontal bar are at:

$$I_h = I_m \pm \frac{s}{4}\vec{U} \quad (5)$$

The four endpoints of the IBar are at:

$$I_{t/bl/r} = I_t \pm s\vec{U} \pm s\vec{L} \quad (6)$$

5.2 Manipulating the IBar

The camera parameters are changed when the user manipulates the IBar. The IBar is then drawn with the new camera parameters; hence the manipulations are indirectly reflected in the changed projection. In object-space mode the scene is drawn with the new camera. In camera-space mode the scene is drawn with the original camera; when the manipulation is finished, the final camera is created by inverting the appropriate action (for instance, panning in the opposite direction).

The following is a summary of the variables (mouse and projected IBar) used to update the camera.

Name	Variable
Projected limb base	l_b
Projected limb	$\vec{l}$
$\vec{l}$ normalized	$\hat{l} = \vec{l}/\|\|\vec{l}\|\|$
Mouse down position	p
Current mouse position	q
Mouse move	$\vec{v} = q - p$

Table 3: Manipulation parameters. All values are in camera coordinates, $[-1,1] \times [-1,1]$. The limb is whichever limb (four arms or horizontal bar) or stem that was selected. The base is the base of the selected limb (one of I_t, I_b, or I_m).

Pan (changing I_m): The camera is moved by the mouse vector projected into the film plane:

$$T' = T + sv_x\vec{U} + sv_y\vec{V} \quad (7)$$

Uniform zoom (changing I_h or all limb lengths): The focal length f is scaled by the length change of the limb:

$$f' = f\frac{<\hat{l}, p - l_b>}{<\hat{l}, q - l_b>} \quad (8)$$

Spin (Rotating the stem): The `Up` and `Right` vectors are rotated around the `Look` vector by (R_z is a rotation around the z axis):

$$r = \tan^{-1}(q_y/q_x) - \pi \qquad \textit{top selected} \quad (9)$$

$$r = \tan^{-1}(q_y/q_x) + \pi \qquad \textit{bottom selected} \quad (10)$$

$$\vec{U,V}' = R_z\vec{U,V} \quad (11)$$

Rotate (lengthening the left-right or top-bottom limbs): The camera is rotated about the focus point ($f_p = T + d\vec{L}$). The rotation is either around the `Up` vector (left-right limbs) or the `Right` vector (top-bottom limbs).

$$\alpha = v_x\pi/2 \quad (12)$$

$$U, \vec{V}, L' = R^T R(\alpha) RU, \vec{V}, L \quad (13)$$

$$T' = (T + d\vec{L}) - d\vec{L}' \quad (14)$$

where $R(\alpha)$ is either a rotation around the X (top-bottom) or the Y (left-right) axis. If the selected limb is the bottom one, and the rotation is top-bottom, then $\alpha = -v_x\pi/2$.

Dolly with zoom (changing the angle of all four limbs): The focal distance is adjusted by the change in angle, then the focal length is modified so that the object does not change size. The desired focal distance change is found by moving the limb in 3D, projecting it, and comparing the resulting angle. The limb is moved by:

$$\Delta_y = \pm sv_y \quad (15)$$

where the sign is taken so that the limbs move in the appropriate direction. The new focal distance and focal length are then:

$$y = (l_b)_y - \vec{l'}_y \quad (16)$$

$$d' = -1/2 + (W/H)/(4y) \quad (17)$$

$$f' = fd'/d \quad (18)$$

where l' is the limb adjusted by Equation 15.

Center-of-projection: The center of projection is changed to reflect the change in the ratio of the angles of the limbs (either left-right or top-bottom). The camera is then panned in the opposite direction to keep the IBar in the middle of the screen.

$$u_0' = u_0 + v_y \quad (19)$$

$$T' = T - sv_y\vec{U} \quad (20)$$

$$\textit{or} \quad (21)$$

$$v_0' = v_0 + v_y \quad (22)$$

$$T' = T - s(W/H)v_y\vec{V} \quad (23)$$

$$(24)$$

5.3 Camera-based

In camera-based mode the final camera is *not* the one calculated above, but a camera that moves the scene in the opposite direction. This is easily implemented by changing p, q, and v, and using the same equations. For the pan and zoom operations, swap the role of p and q, *i.e.*, $v = -v$. For the spin operation, negate the sign of the x component of v (which creates a rotation in the opposite direction).

5.4 Horizon line

To place the horizon line, first project the limbs and the stem of the IBar into 2D. Intersect the left limbs and the right limbs, to produce two points. Intersect the line formed by connecting these two points with the line of the stem; the percentage t along the stem is used to move the horizontal bars:

$$I_h = (1-t)I_t + tI_b \pm \frac{s}{4}\vec{U} \quad (25)$$

5.5 Placing the IBar

To place the IBar at an arbitrary point p_d in the scene, replace Equation 2 with:

$$I_m = p_d + (su_0)\vec{U} + (sv_0)\vec{V} \quad (26)$$

The focus distance is $d = p_d - T$. The camera manipulations of the previous section remain the same, except for the rotations, which now rotate around p_d. Equation 14 is the same, except for the calculation of T':

$$T' = p_d + R^T R(\alpha) R(T - p_d) \quad (27)$$

6 Conclusion

We have presented a simple, easy-to-use screen-space widget for controlling all aspects of a perspective projection, in particular the internal camera parameters. The widget allows the user to manipulate the camera using just the mouse, and provides visual clues about how the perspective will change with manipulation.

The IBar appears to provide a better conceptual insight, especially for novice users, into how the camera works than a traditional user interface. The combined zoom and dolly was particularly popular, as was the lack of menus and need for interaction mode changes.

A Projection matrix

$$k = near/far \quad (28)$$

$$P = \begin{bmatrix} 1 & 0 & u_0 & 0 \\ 0 & 1 & v_0 & 0 \\ 0 & 0 & \frac{-1}{1+k} & \frac{k}{1+k} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (29)$$

$$S = \begin{bmatrix} \frac{H}{ffar} & 0 & 0 & 0 \\ 0 & \frac{W}{ffar} & 0 & 0 \\ 0 & 0 & \frac{1}{far} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (30)$$

$$R = \begin{bmatrix} - & \vec{U} & - & 0 \\ - & \vec{V} & - & 0 \\ - & \vec{L} & - & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (31)$$

$$\begin{bmatrix} u \\ v \\ z \\ w \end{bmatrix} = PSR \left[\begin{array}{cccc} 1 & 0 & 0 & | \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & | \\ 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (32)$$

$$u' = u/w \quad (33)$$

$$v' = v/w \quad (34)$$

References

Blinn, J. 1988. Where am i? what am i looking at? In *IEEE Computer Graphics and Applications*, vol. 22, 179–188.

Bowman, D. A., Koller, D., and Hodges, L. F. 1997. Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. *IEEE Proceedings of VRAIS'97*, 7, 45–52.

Carlbom, I., and Paciorek, J. 1978. Planar geometric projections and viewing transformations. In *ACM Computing Surveys (CSUR)*, vol. 10.

Cole, R. V. 1976. *Perspective for Artists*. Dover Publications.

Gleicher, M., and Witkin, A. 1992. Through-the-lens camera control. In *Siggraph*, E. E. Catmull, Ed., vol. 26, 331–340. ISBN 0-201-51585-7. Held in Chicago, Illinois.

Hultquist, J. 1990. A virtual trackball. In *Graphics Gems*. 462–463.

Michener, J. C., and Carlbom, I. B. 1980. Natural and efficient viewing parameters. In *Computer Graphics (Proceedings of SIGGRAPH 80)*, vol. 14, 238–245.

o'Connor Jr., C., Kier, T., and Burghy, D. 1998. *Perspective Drawing and Application*. Prentice Hall.

Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. 1996. The go-go interaction technique: Nonlinear mapping for direct manipulation in VR. In *ACM Symposium on User Interface Software and Technology*, 79–80.

Stoakley, R., Conway, M. J., and Pausch, R. 1995. Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings CHI'95*.

Wloka, M. M., and Greenfield, E. 1995. The virtual tricorder: A uniform interface for virtual reality. In *ACM Symposium on User Interface Software and Technology*, 39–40.