Online Submission ID:

Implementation of the IBar: A Perspective-based Camera Widget

Category: Research

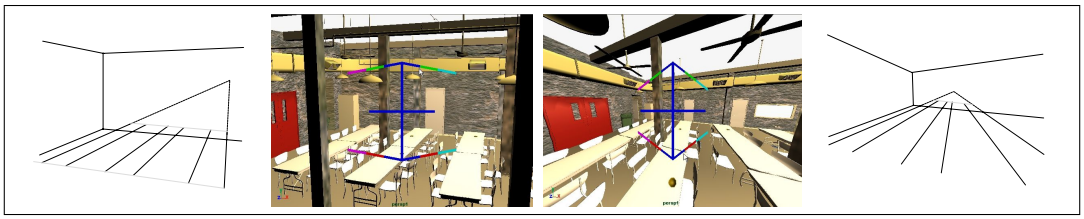


Figure 1: Using the IBar to adjust the perspective distortion of a scene.

Abstract

We present the implementation of 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).

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

Keywords: Camera control, Projection, Perspective, Rendering

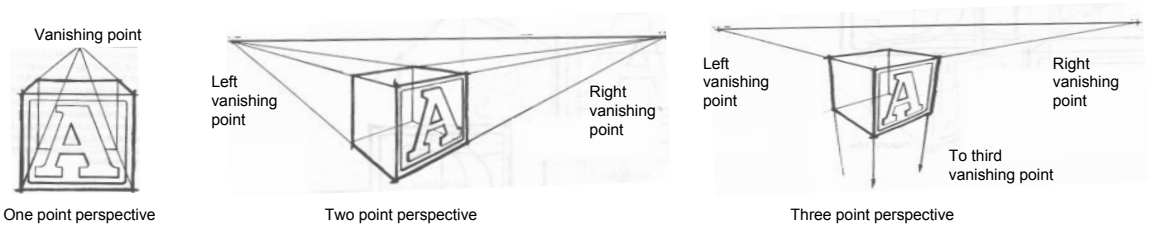


Figure 2: Terms used by artists to describe perspective projections. From: *Perspective Drawing and Applications*.

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 previous work [Singh et al. 2004] the authors presented a single screen-space widget that provides intuitive manipulation of *all* of the camera parameters 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. In this paper we detail the implementation of this widget.

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 2 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.

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. This paper provides complete implementation details of the IBar, which was originally described in a previous paper [Singh et al. 2004].

We briefly review existing camera models first (Section 2). The functionality of the IBar is described in Section 3. The corresponding equations are given 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, pan, and roll) 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 a 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

A schematic diagram of the IBar widget is shown in Figure 3. Conceptually, the IBar represents the two-point perspective 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

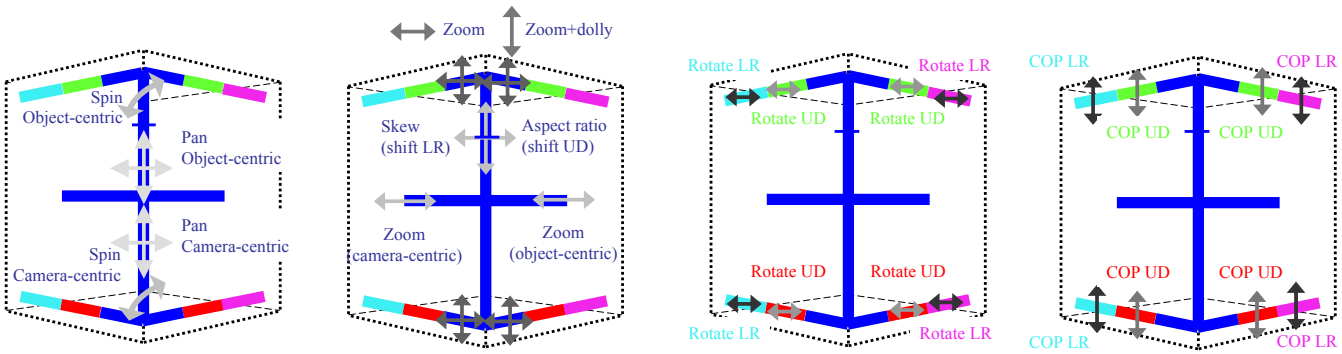


Figure 3: A schematic of the IBar widget. Arrows mark handle locations and available movement directions. Moving the first third of any IBar limb moves all four limbs simultaneously. Moving the second third of the top (or bottom) limb moves both top (or bottom) limbs simultaneously. Moving the last third 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. **Optional:** Camera-centric operations (pan, spin, zoom) are mapped to the bottom half of the stem and tbe left horizontal bar, object-centric operations to the top half of the stem and the right horizontal bar.

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. This makes it simple to center, align, and frame the camera around an object in the scene.

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. Details on drawing the IBar are in Section 4.1.

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:*

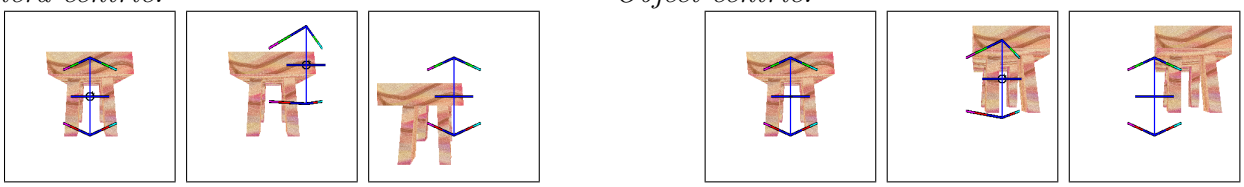


Figure 4: **Pan:** Move the IBar using the handle at the center.

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

Camera-centric: *Object-centric:*

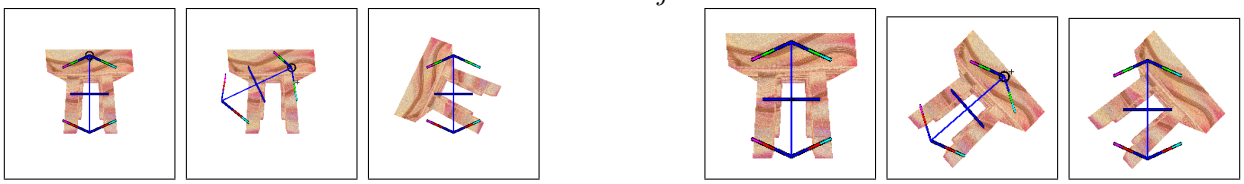


Figure 5: **Camera spin:** To rotate the camera about its `Look` vector, rotate the IBar using the top (or bottom) of the stem.

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.

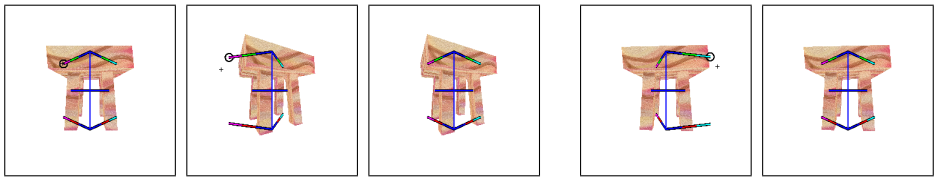


Figure 6: **Camera rotate left-right:** To rotate the camera left or right, scale the appropriate side limbs. Lengthening the right limbs is equivalent to shortening the left limbs.

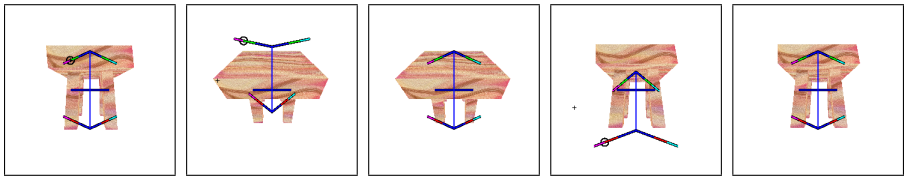


Figure 7: **Camera rotate up-down:** To rotate the camera into or out of the film plane, scale the appropriate top or bottom limbs. Lengthening the top limbs is equivalent to shortening the bottom limbs.

3.4 Zoom and dolly-in

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

Camera-centric: *Object-centric:*

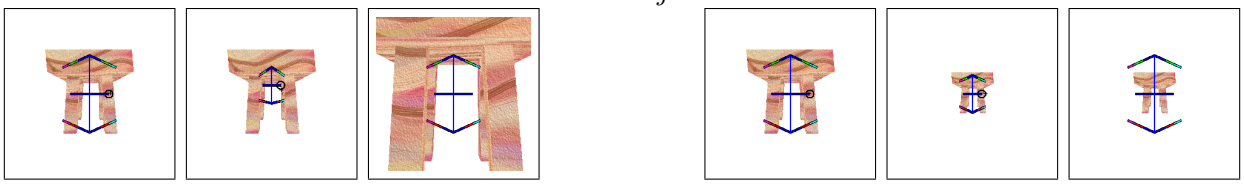


Figure 8: **Zoom in/out:** To zoom the camera in and out without changing the perspective distortion, scale the middle of the IBar.

Dolly only: *Dolly plus zoom:*

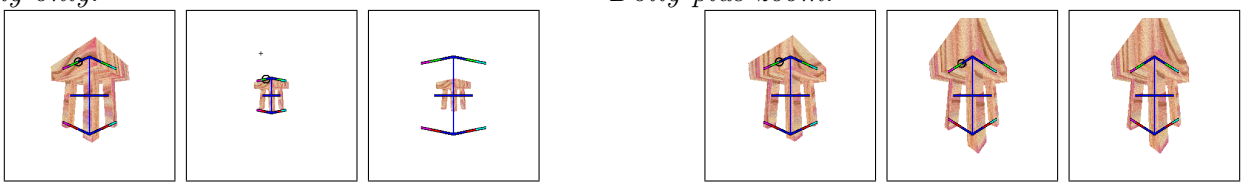


Figure 9: **Dolly in/out:** To dolly the camera in/out without affecting the focal length, change the angles on all four limbs simultaneously while holding the shift key. **Dolly in/out with zoom:** To dolly the camera in/out while simultaneously change the focal length, change the angles on all four 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.

Aspect ratio: *Skew:*

Figure 10: **Aspect ratio:** To change the aspect ratio, grab a point on the IBar stem and move up-down while holding the shift key. **Skew:** To change the skew, grab a point on the IBar stem and move left-right while holding the shift key.

Vertical: *Horizontal:*

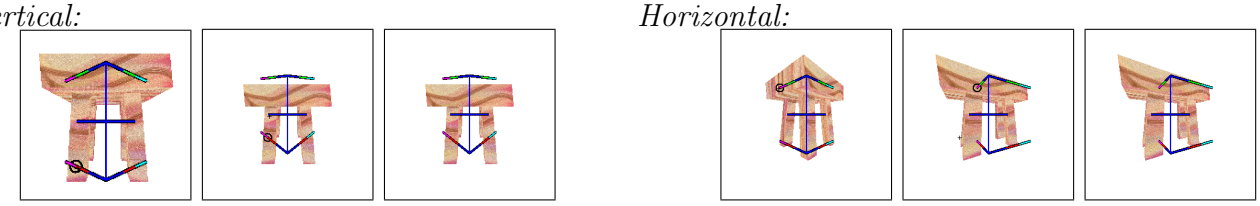


Figure 11: **COP:** To change the center of projection, make the angles of the top limbs different from the bottom ones (moves the center of projection up/down). Similarly, making the angles of the left limbs different from the right moves the center of projection left-right.

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.

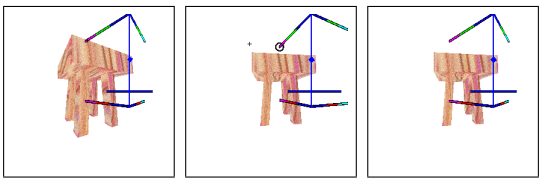


Figure 12: **Object rotate:** Rotating the IBar around an arbitrary point in the scene.

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 mode and all of the bottom limbs to object-centric mode. This has the advantage of eliminating modes, but it does increase the number of distinct handles.

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 (refer to Figure 3). 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.

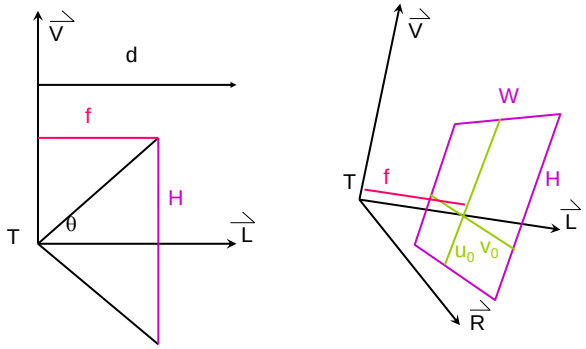
One advantage or the IBar is that it does not require the exclusive use of a mouse button, enabling the application to utilize all of the mouse buttons for other tasks, such as manipulating objects. To support this, we first alter the drawing routine so that the IBar is highlighted whenever the mouse is over an active part of the IBar. Second, we use the `ALT` key to temporarily disable the IBar so that the user can click through the IBar.

4 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.

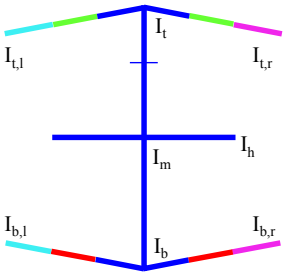
Name	Variable
Screen size	W,H
Position	T
Look, Up	$\vec{L}$, $\vec{U}$
Right	$\vec{R} = \vec{U} \times \vec{L}$
Focus distance	d
Focus point	$O_f = T + d\vec{L}$
Focal length	f
Aperture angle	$\theta = 2\tan^{-1}(H/f)$
Center of proj.	(u_0, v_0)
Film plane scale	$s = d(H/f)$
Aspect ratio, skew	α, γ
Height of cube	s_c

Table 1: The camera and its parameters.



4.1 Drawing the cube

To create the IBar we build a cube with one edge centered on the `Look` $\vec{L}$ vector and oriented parallel to the `Up` $\vec{U}$ vector (refer to Figure 13. This is the *stem* of the IBar. To keep the projected size of an object at distance d from the camera constant we use the following scale factor:



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

Figure 13: Drawing the IBar.

Let s_c be the desired height of the cube on the screen; for the images in this paper, $s_c = 1/2$. The center of the cube edge is placed at:

$$I_m = T + d\vec{L} + (\frac{s}{\alpha}u_0 - \gamma v_0)\vec{R} + v_0 s\vec{U} \quad (2)$$

The second two terms counter-act any shift caused by a non-zero center of projection, skew, or non-unity aspect ratio. The top t and bottom b points of the IBar stem are at:

$$I_t = I_m + s_c s\vec{U} \quad (3)$$

$$I_b = I_m - s_c s\vec{U} \quad (4)$$

The end-points of the four adjacent cube edges (limbs) are found by extending back in the `Look` and `Right` directions:

$$I_{t/b,l/r} = I_{t/b} \pm s_c s\vec{R} + s_c s\vec{L} \quad (5)$$

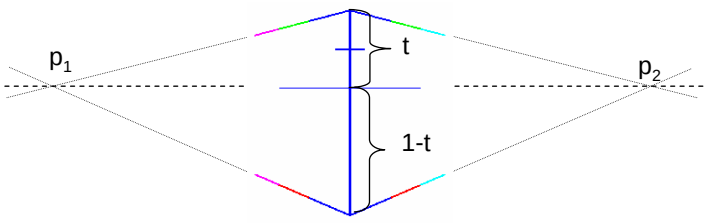


Figure 14: Calculating where to place the horizon bar.

The horizontal bar can be placed either at I_m, or so that it indicates the horizon line (see Figure 14). To place the horizon line, first project the limbs and the stem of the IBar into the film plane. Intersect the projected left limbs and the right limbs, to produce two points p_1, p_2. Intersect the line formed by connecting p_1 and p_2 with the line of the stem; the percentage t along the stem is used to move the horizontal bar end-points:

$$I_h = (1-t)I_t + tI_b \pm \frac{s_c s}{2}\vec{R} \quad (6)$$

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

$$I_m = O_f + (\frac{s}{\alpha}u_0 - \gamma v_0)\vec{R} + v_0 s\vec{U} \quad (7)$$

In this case, d is defined to be $||O_f - T||$.

4.2 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 both the scene and the IBar are drawn with the new camera. In camera-space mode the scene is drawn with the original camera but the IBar is drawn with the new 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 mouse variables used to update the camera.

Name	Variable
Projected IBar point, *e.g.*,	$l_* = P(I_*)$
Mouse down position	p
Current mouse position	q
Mouse move	$\vec{v} = q - p$

Table 2: Manipulation parameters. All values are in camera coordinates, $[-1, 1] \times [-1, 1]$. $P(I_*)$ is the screen-space projection of the IBar point by the current camera (see Appendix A)

.

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

$$T' = T + sv_x\vec{R} + sv_y\vec{U} \quad (8)$$

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

$$r_l = \frac{<\hat{l}, q-l>}{<\hat{l}, p-l>} \begin{cases} \hat{l} = \frac{l_h - l_m}{||l_h - l_m||} & l = l_m \quad or \\ \hat{l} = \frac{l_{t,l/r} - l_t}{||l_{t,l/r} - l_t||} & l = l_t \quad or \\ \hat{l} = \frac{l_{t,l/r} - l_b}{||l_{b,l/r} - l_b||} & l = l_b \end{cases} \tag{9}$$

$$f' = fr_l \tag{10}$$

Spin (Rotating the stem): The `Up` and `Right` vectors are rotated around the `Look` vector by:

$$r = \tan^{-1}(q_y/q_x) - \pi \qquad \textit{top selected} \tag{11}$$

$$r = \tan^{-1}(q_y/q_x) + \pi \qquad \textit{bottom selected} \tag{12}$$

$$\vec{U}' = R_{look}(r)\vec{U} \tag{13}$$

$$\vec{R}' = R_{look}(r)\vec{R} \tag{14}$$

where $R_{look}(r)$ is a 3×3 rotation matrix that rotates around the `Look` vector by r.

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

$$r = -v_x\pi/2 \tag{15}$$

$$R = \begin{bmatrix} - & \vec{R} & - \\ - & \vec{U} & - \\ - & \vec{L} & - \end{bmatrix} \tag{16}$$

$$\vec{L}' = R^T R(r) R\vec{L} \tag{17}$$

$$\vec{R}' = R^T R(r)\vec{R}\vec{L} \tag{18}$$

$$\vec{U}' = R^T R(r)\vec{U}\vec{L} \tag{19}$$

$$\vec{S} = su_0\vec{R} + sv_0\vec{U} \tag{20}$$

$$\vec{S'} = su_0\vec{R'} + sv_0\vec{U'} \tag{21}$$

$$T' = (f_p - \vec{S}) - d\vec{L'} - \vec{S'} \tag{22}$$

$$f'_p = f_p + \vec{S} - \vec{S'} \tag{23}$$

$R(r)$ 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 $r = v_x\pi/2$. Note that, in order to rotate around the focus point, we must first undo the center of projection shift ($\vec{S}$), rotate, then redo ($\vec{S'}$) the COP shift.

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 new focal distance is:

$$d' = \begin{cases} d(1 - v_y) & v_y \leq 0 \\ d(1 + 2v_y) & v_y > 0 \end{cases} \tag{24}$$

The new focal length is then:

$$f' = f\frac{d'}{d} \tag{25}$$

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.

$$u'_0 = u_0 + v_y \tag{26}$$

$$T' = T - sv_y\vec{U} \tag{27}$$

$$or \tag{28}$$

$$v'_0 = v_0 - v_y \tag{29}$$

$$T' = T + s(W/H)v_y\vec{R} \tag{30}$$

$$\tag{31}$$

Aspect ratio: The aspect ratio is found by the ratio of the stem length change:

$$\hat{l} = \frac{l_t - l_b}{||l_t - l_b||} \tag{32}$$

$$l = l_b \tag{33}$$

$$r_l = \frac{<\hat{l}, q - l>}{<\hat{l}, p - l>} \tag{34}$$

$$\alpha' = r_l\alpha \tag{35}$$

Skew: The new skew is adjusted by the x component of the mouse vector:

$$\gamma' = \gamma + v_x \tag{36}$$

4.3 Camera-based mode

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 swapping p, q 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 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.

A Projection matrix

$$k = near/far \tag{37}$$

$$P = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & 1 & v_0 & 0 \\ 0 & 0 & \frac{-1}{1+k} & \frac{k}{1+k} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{H}{ffar} & 0 & 0 & 0 \\ 0 & \frac{W}{ffar} & 0 & 0 \\ 0 & 0 & \frac{1}{far} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} - & \vec{U} & - & 0 \\ - & \vec{V} & - & 0 \\ - & \vec{L} & - & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & | \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & | \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} u & v & z & w \end{bmatrix}^T = P \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \tag{39}$$

$$P(\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T) = \begin{bmatrix} \frac{2u}{Ww} - 1 & \frac{2v}{Hw} - 1 \end{bmatrix}^T \tag{40}$$

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: Non-linear mapping for direct manipulation in VR. In *ACM Symposium on User Interface Software and Technology*, 79–80.

Singh, K., Grimm, C., and Sudarsanan, N. 2004. The ibar: A perspective-based camera widget. In *UIST*.

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.