The IBar: A Perspective-based Camera Widget

Karan Singh
University of Toronto
karan@dgp.toronto.edu

Cindy Grimm, Nisha Sudarsanam
Washington University in St. Louis
cmg,nsudarsa@wustl.edu

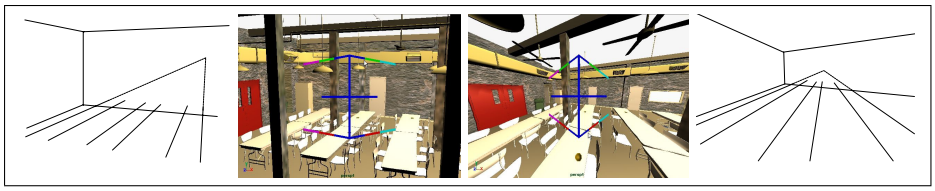


Figure 1: Changing the perspective distortion of the scene.

ABSTRACT

We present a new screen space widget, the IBar, for effective camera control in 3D graphics environments. The IBar provides a compelling interface for controlling scene perspective based on the artistic concept of vanishing points. Various handles on the widget manipulate multiple camera parameters simultaneously to create a single perceived projection change. For example, changing just the perspective distortion is accomplished by simultaneously decreasing the camera's distance to the scene while increasing focal length. We demonstrate that the IBar is easier to learn for novice users and improves their understanding of camera perspective.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI)

Additional Keywords and Phrases: Camera control, Perspective, Widgets.

Introduction

Camera control for 3D rendering is a difficult problem. A full perspective matrix [7] 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 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.

Part of the difficulty of camera control is the complex interrelationships 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.

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA.

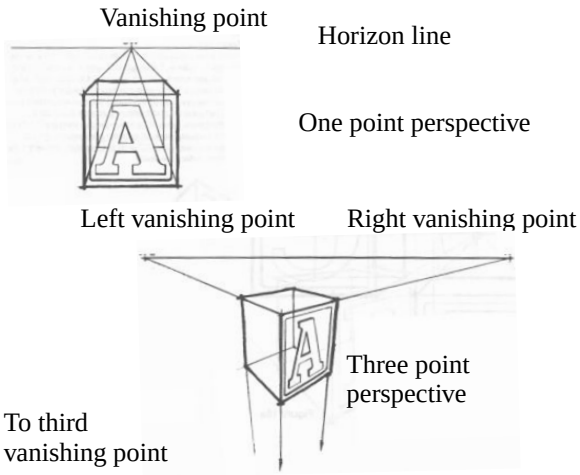


Figure 2: Terms used by artists to describe perspective projections. From: *Perspective Drawing and Applications*.

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 [2, 7]. 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; 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, reducing 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. 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, the vanishing points or perspective distortion of the scene are controlled by a combination of the camera's focal length, film offset and camera position relative to the scene.

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.

Contributions: The IBar is an effective camera control widget. In particular it provides a natural interface for manipulating camera parameters that influence perspective distortion, simplifying the control and animation of dramatic camera effects seen in Figure 1. Visualization of the mathematics of projection also makes the IBar easy to understand and use for novices.

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 commercial graphics system such as Maya.

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 [6]). 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 but is less useful in complex scenes.

An alternative approach uses image-space constraints [1, 4, 3], where points in the scene are constrained to appear at particular locations, 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.

The IBar Widget

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

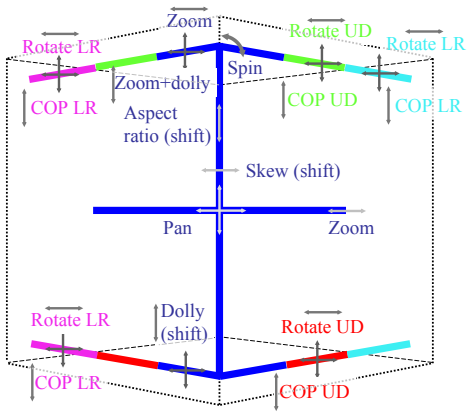


Figure 3: Arrows mark handle locations and movement directions. First third of any IBar limb moves all four limbs simultaneously. Second third moves both top (or bottom) limbs simultaneously. Last third moves both left (or right) limbs simultaneously. Moving left-right changes the limb length, moving up-down changes the angle.

rendering of a cube centered on the `Look` vector of the camera (see table). The IBar is inspired by the use of 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 with respect to the box represented by the IBar. The exact behavior depends on whether or not the IBar is in camera or object-centric mode (discussed below for each operation).

Changing the angles of the limbs corresponds to moving or changing the vanishing points. This causes the camera to move along the look vector, 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.

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.

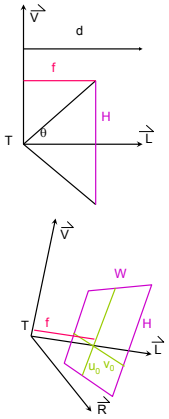
The IBar represents a unit cube at a distance d from the cam-

era. 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 $d(H/f)$ where f is the focal length. The limbs of the IBar are the projection of the adjacent cube edges.

Name	Variable
Screen size	W,H
Position	T
Look, Up	$\vec{L}$, $\vec{V}$
Right	$\vec{R} = \vec{V} \times \vec{L}$
Focus distance	d
Focal length	f
Center of proj.	(u_0, v_0)
Film plane scale	$s = d(H/f)$

Table 1: Camera parameters.



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.

Pan: Panning the camera is accomplished by dragging the middle of the IBar. In camera-centric mode, the point in the scene under the mouse is "snapped" to the center of the screen when the button is released. This is useful for centering the camera on a particular point. In object-centric mode, the scene moves simultaneously with the mouse.

Object-centric:

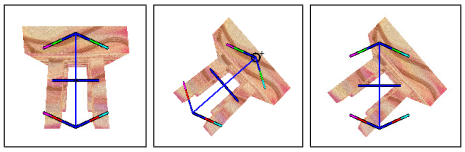


Figure 4: **Roll:** To rotate the camera about its `Look` vector, rotate the IBar using the top (or bottom) of the stem. In camera-centric mode, the vertical bar is aligned with the desired vertical axis for the scene.

Roll left-right, up-down: The camera is rotated by changing the lengths of the second (up/down) or third (left/right) segment. These two operations support the traditional virtual trackball [6] manipulation. Additionally, coarse and fine control is provided by either expanding or shrinking the limbs. 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.

Zoom and dolly: These operations change the size of the rendered objects, and, optionally, the perspective distortion. To zoom the camera in and out without changing the perspective distortion, scale the middle of the IBar. To move the camera in or out while changing the focal length to keep the object the same size, change the angles on all four limbs simultaneously. (Holding the shift key down changes just the focal length.) In camera-centric mode, the IBar is scaled to frame an object (or objects); at mouse release, the framed object is scaled to fill one half of the screen.

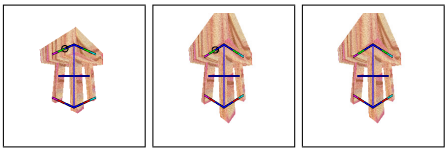


Figure 5: **Dolly in/out with zoom:** To dolly the camera in/out and simultaneously change the focal length, change the angles on all four limbs simultaneously.

Internal Camera Parameters

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

Vertical:

Horizontal:

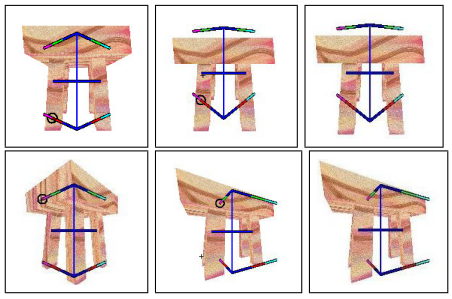


Figure 6: **COP:** To change the center of projection, make the angles of the top limbs different from the bottom ones (moves the center of projection up/down). Similarly, making the angles of the left limbs different from the right moves the center of projection left-right.

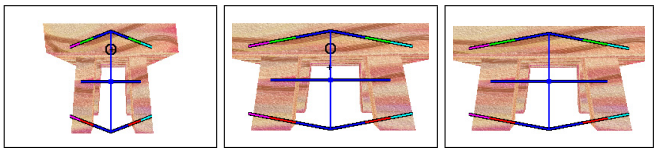


Figure 7: **Aspect ratio:** To change the aspect ratio, grab a point on the IBar stem and move up-down while holding the shift key.

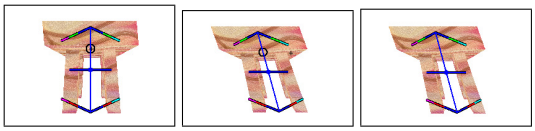


Figure 8: **Skew:** To change the skew, grab a point on the IBar stem and move left-right while holding the shift key.

Options

We describe several extensions to the basic IBar. The first one places the horizontal bar so that it indicates the horizon line. The second places the IBar at a specified point in the

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

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.

Third, there are several possible methods for switching between camera- and object-based modes, for example, by using a toggle switch or a key-modifier. An alternative is to take advantage of the multiple handles available for each camera operation. For example, there are two zoom handles (left and right). We map the left handle to the camera-centric zoom and the right handle to the object-centric version. This has the advantage of eliminating modes, but it does increase the number of distinct handles.

Finally, the shift-key is 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 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.

User study

We performed a small user study to compare a Maya implementation of the IBar with Maya's traditional camera interface. Our observations are as follows. First, our knowledgeable participants performed, on average, as well as the novice users. 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. Third, most of the users found the IBar more intuitive. For a complete description of the study, see [5].

The IBar has been in use in our lab for several months in a variety of applications (surface modeling, visualization, and scene construction). Anecdotal evidence suggests that the IBar is not visually distracting, but it can accidentally "grab" mouse events intended for other manipulations. A simple toggle switch and visually highlighting when the IBar is active greatly reduces these problems.

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.

Appendix: Implementation

In this section we define the equations that correspond to the camera manipulations in the previous section. Our camera parameters are summarized in Table 1; the perspective matrix is built from these parameters in the usual way [7]. For complete details, see [5].

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-centric mode the scene is drawn with the new camera. In camera-centric 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).

Pan: The camera is moved by the mouse vector projected into the film plane.

Uniform zoom: The focal length f is scaled by the ratio of limb lengths before and after manipulation.

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

Rotate (lengthening the left-right or top-bottom limbs): This is the standard track-ball rotation [6].

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. Let $y = l_y - \vec{l'}_y$, where l is the original and l' the adjusted limb. The new focal distance d' is $-1/2 + (W/H)/(4y)$ and the new focal length is fd'/d.

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. Let v_y be the vertical change in the limb. The new center of projection is $u_0 + v_y$ (or $v_0 + v_y$), and the adjusted translation is $T - sv_y\vec{R}$ (or $T - sv_y\vec{V}$).

REFERENCES

1. Jim Blinn. Where am i? what am i looking at? In *IEEE Computer Graphics and Applications*, volume 22, pages 179–188, 1988.
2. Rex Vicat Cole. *Perspective for Artists*. Dover Publications, 1976.
3. Steven M. Drucker and David Zeltzer. Camdroid: A system for implementing intelligent camera control. In *1995 Symp. on Interactive 3D Graphics*, pages 139–144. April 1995.
4. Michael Gleicher and Andrew Witkin. Through-the-lens camera control. *Siggraph*, 26(2):331–340, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.
5. Cindy Grimm, Karan Singh, and Nisha Sudarsanan. The ibar: A perspective-based camera widget. Technical Report WUCSE-2004-32, Wash. univ. in St. Louis, 2004.
6. Jeff Hultquist. A virtual trackball. In *Graphics Gems*, pages 462–463. 1990.
7. J. C. Michener and I. B. Carlbom. Natural and efficient viewing parameters. *Computer Graphics (Proceedings of SIGGRAPH 80)*, 14(3):238–245, July 1980.
8. Charles o'Connor Jr., Thomas Kier, and David Burghy. *Perspective Drawing and Application*. Prentice Hall, 1998.