QuickDraw 3D Preliminary User Interface Guidelines

1.1 Introduction

The QuickDraw 3D user interface toolkit strives to furnish developers with the functionality that will allow their application users to interact with 3D objects in a natural, facile way. This chapter describes how the user centered design process applies to the design of 3D interaction in QuickDraw 3D, some information on users and their tasks, and details on specific interaction techniques, as well as the contexts in which each is appropriate. This document offers **preliminary** guidelines; as the user interface design in QuickDraw 3D evolves these guidelines will be expanded and finalized in subsequent releases. Information in document is subject to change, and should be taken as an indication of future directions rather than finished work.

1.2 User Centered Design in QuickDraw3D

Since one of the primary motivations behind QuickDraw 3D is to broaden the market for 3D graphics applications from the high-powered-workstation-based technical user, to the wider audience of personal computer users, it is critical to design the human interface toolkit for QuickDraw 3D with the user in mind. User centered design simply means taking the user perspective during design, and including user-oriented considerations as equal to architectural or technical considerations when making design tradeoffs. It is not adequate to design and develop a product, then think about the user interface as a layer that can be applied at the end; at that point the design space may be too restricted to deliver functionality to the user in a way that is easy to learn and to use.

User centered design simply requires:

- early focus on users and their tasks
- user participation in the design
- design with guidelines for consistency
- user testing
- iteration of the design based on test outcomes

1.2.1 QuickDraw 3D Users

Strictly speaking, QuickDraw 3D "users" are application developers—QuickDraw 3D is a graphics library and user interface toolkit that Apple is providing to application developers to build innovative, new 3D graphics applications. So, in that sense, end users will not be "using QuickDraw 3D", but rather the applications built with it.

So, end-users will still be at your mercy to provide them with an easy to use human interface. However, providing an extensive user interface toolkit with the QuickDraw 3D APIs, will make your task easier by giving you ready-made HI components—you don't have to invent anything new. The result for the end user is that if the toolkit is designed well, the applications they use should be easier and more consistent.

Because an intent of QuickDraw 3D is to make 3D graphics more available and accessible, end-users of QuickDraw 3D applications can include users at all levels of expertise. At one extreme would be just about anyone who produces documents with graphics in them, anyone who might want to have the added visual appeal of 3D. At the other extreme, applications built with QuickDraw 3D have to provide the power and flexibility that also appeal to the high-end, graphics professional. No single application can be expected to span this entire range. Rather, an application should be designed with a specific user model in mind.

1.2.1.1 Casual Users

This group includes just about anyone who currently uses 2D graphics applications and wants to add create and add some simple 3D object (a logo, perhaps) into a document, newsletter, report, or presentation for added visual appeal and impact. Their computer expertise probably is restricted to conventional office or home applications, (e.g. word-processing, spreadsheets, and graphics).

Such users probably have no formal background in computer graphics and may only have a dim understanding of the differences between text and graphics processing. As such these users can not be expected to understand anything about how 3D graphics objects are created, stored, represented, or manipulated; they should be assumed not to understand any specialized terminology.

Another distinguishing characteristic of such users is that they probably will use 3D graphics application only infrequently. Because of infrequency of use, such users will almost always be novices trying to relearn the application.

1.2.1.2 Intermediate Users

This group of users probably has graphics background, but not primarily 3D. Their tasks would include developing 3D scenes of moderate complexity, and perhaps including simple animations. For example, they might be doing package design in 3D, or rendering an imported architectural drawing, or creating a simple flying logo animation.

This group of users is more sophisticated in their understanding of the fundamentals of graphics, and perhaps of 3D. These users would be expected to spend more of their time working in a 3D application and as such would tend to be more in need of accelerators.

1.2.1.3 Professional Users

This user is a full time 3D designer producing photorealistic 3D images and animation, visualization, special effects for film and video. This class of user probably spends most or all of their time using a variety of 3D applications for modeling, rendering, and animation. These users clearly would have a firm understanding of the foundations and terminology inherent in 3D graphics.

1.2.2 User Tasks

The following user task description will provide a useful context for later description of the details of the QuickDraw 3D user interface toolkit. These user tasks will not always be used by all users for all tasks, but form a useful subset for further discussion.

In general the following steps are used in the production of a 3D scene. The scene is generated by either importing or creating the desired objects; these objects are then modified as needed to their final configuration. Objects are then moved and arranged in the scene to reflect the desired scene. The objects appearance is then controlled by applying materials, and lighting. Finally, the viewpoint is arranged to yield the desired final result and a final rendering of the scene is made. Then, since most 3D graphics are destined for some existence other than on the machine they were created on, the scene is exported to some final medium such as print, film, video, CD-ROM, etc. Actual workflow involves intricate iteration through these design steps.

1.3 QuickDraw 3D User Interface Overview

1.3.1 Conceptual Model

There are many ways to think about 3D interface. One way to begin is to emulate the philosophy behind the Apple Lisa and Macintosh: Use the power of the CPU to improve the user's quality of life. This tenet lead to the widespread acceptance of the graphical user interface (GUI) as the preferred way to interact with a computer. Underlying the GUI is the concept of direct manipulation—the use of a pointing device to choose items and operations. Traditional direct manipulation GUI's enable the user to perform a 2D task. These concepts can be translated and extended to facilitate the performance of 3D tasks.

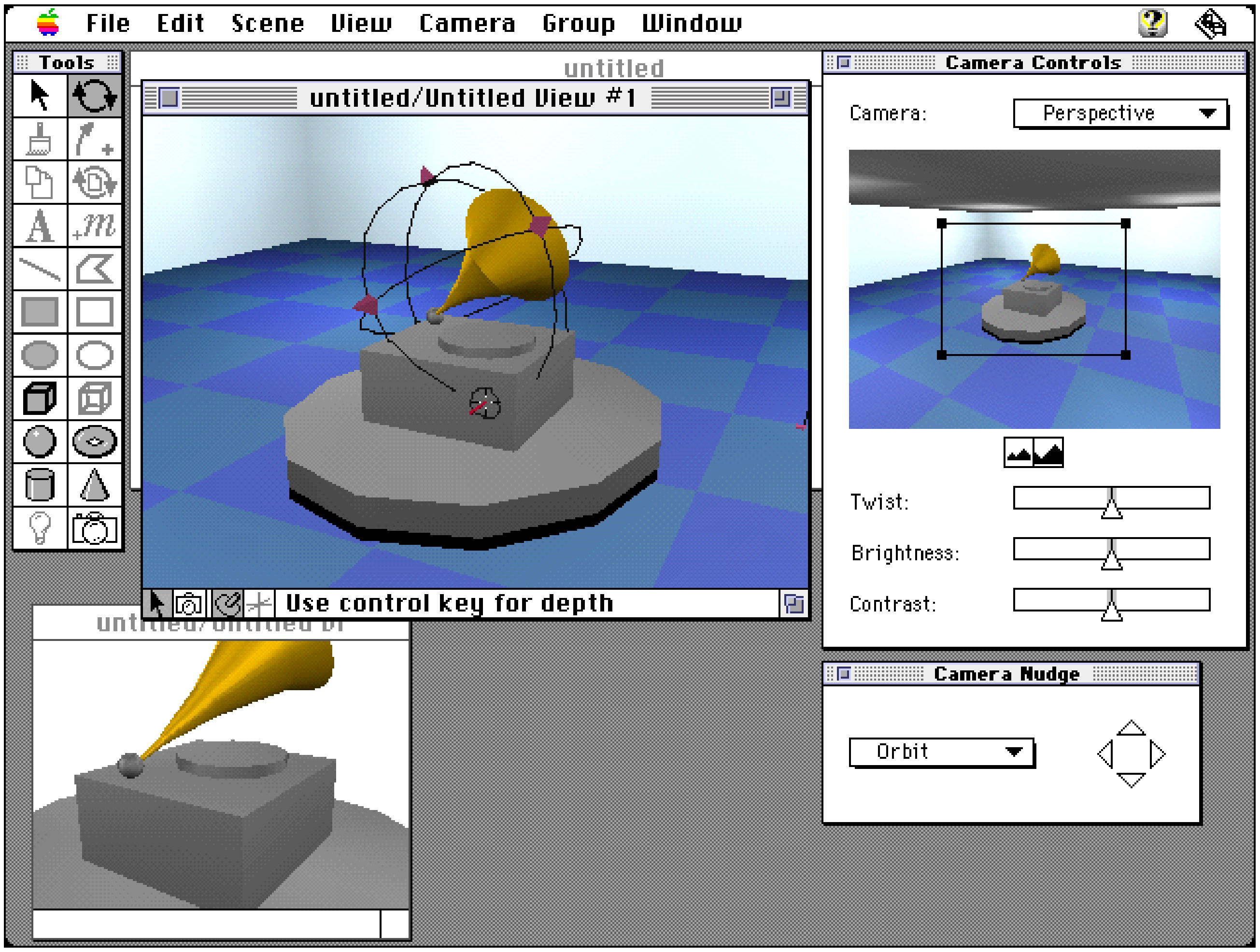
To work in 3D, the user must understand the 3D shapes and their arrangement. Users gain this understanding through interaction and exploration, a dynamic process accomplished over time. It is a cooperative venture between the user and the application being used. To support interaction, the application must complement every user action with appropriate feedback. To enable exploration, the application must, through its interface, encourage the user to look around, to move things, and to change things, all without penalty. When users are allowed to explore and interact with their 3D information, their greater understanding of the 3D configurations allows them to confidently achieve the task at hand.

1.3.2 Working In Context

Many things can be learned from examining existing 2D graphical editors. One is that an object is never isolated from the other objects. In MacDraw, for example, operations such as scaling a circle or reshaping a polygon are done in the window with all the other objects, rather than in a separate window. Many 3D applications, on the other hand, arrange in one window and model in another. This is an artificial and harmful schism, since the changes in the model may depend directly on the way the shape interacts with other shapes in the scene. As much as possible, editing operations should be done in-context.

People are accustomed to working from a frame of reference. In a 3D interface, it is often useful to provide a frame of reference as part of the environment. For example, a modelling program might place the model in a room that contains the model at the center and that contains the camera. The walls of the room can be used to show shadows that themselves are directly manipulable objects (see Figure 1 on page 4).

FIGURE 1 3D Room as a frame of reference



1.3.3 3D Pointing

These techniques are driven first and foremost off of the 2D cursor. When pointing, the shape in the image at the cursor is indicated; when dragging, the dragged point is moved so its image is placed at the cursor (to the extent possible). These techniques allow scalability from a simple mouse to a high-end 3D pointing device.

There are four pointing modes, each of which is accessed and indicated differently in the user interface:

- pick
- guide
- free
- align

1.3.3.1 Pick Pointing

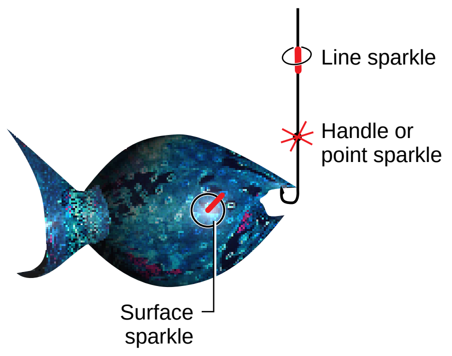
Pick pointing chooses a shape in the scene, or a position that is on a shape. It is typically used for indicating a target for selection (the normal “idle” mode), as well as for aligning a dragged object to some preexisting shape in the scene.

Pick pointing casts a line-of-sight ray into the scene (using window point picking) to determine what is pointed at. Things that are pickable depend on the particular use of pick pointing. For example, when the scale tool is active, the scale widget handle points are pickable, as well as all the shapes in the scene (the selected shape to translate it, the other shapes to select them). Pick pointing can indicate nothing at all if no shape is at the cursor position.

Pick pointing makes use only of the 2D position of the cursor; it ignores any depth or orientation from the pointing device.

When pointing at a pickable feature, a “prehighlight” or “sparkle” is shown at the picked position in the geometry (which is directly under the cursor in the image). The shape of the sparkle depends on the type of feature indicated, whether surface, edge or point (see Figure 2 on page 5).

FIGURE 2 The shape of the sparkle depends on the kind of shape that is indicated.



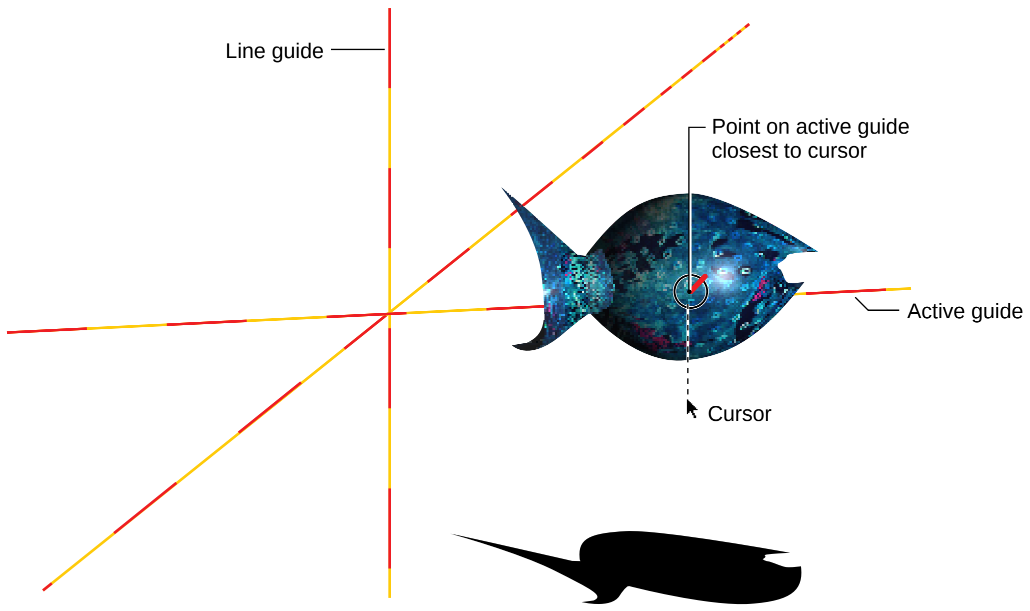
When pick pointing is used for real work (e. g. dragging a feature, as opposed to merely pointing around) it can be a problem when there is no shape at the cursor position. In this case, a temporary switch to free pointing (see below) is made; when the cursor again is over a shape, normal pick pointing resumes.

1.3.3.2 Guide Pointing

Guide pointing chooses a position that lies on some constraint shapes. It is typically used for shift-key constraints. It is also used in place of pick pointing in those operations that rely on pointing to a position on a hemisphere, as when dragging a handle of the rotation widget.

Guide pointing shows guide objects in the 3D scene, such as lines (see Figure 3 on page 6), points, planes, hemispheres, etc. The guide that is closest to the cursor (measured in screen space) is the active guide. The point on the active guide that is closest to the cursor indicates a 3D position in the scene.

FIGURE 3 During translation with the shift key down, line guides are shown to allow movement constrained to a single axis.



Guide pointing and pick pointing are similar: they both choose a position in the scene based on the cursor position and some 3D geometry. One difference is that guide pointing always defines a position, where pick only does so when the cursor is over a pickable shape.

Guide pointing makes use only of the 2D position of the cursor. It ignores any depth or orientation from the pointing device.

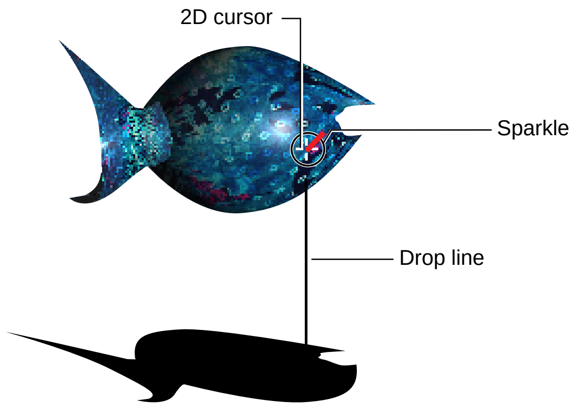
Some indicator should be shown at the position indicated by guide pointing. Typically this is the sparkle which is being dragged.

1.3.3.3 Free Pointing

Free pointing controls an arbitrary position in 3D space. It uses the “Z axis” control of the 3D pointing device to determine the depth along the line-of-sight ray through the cursor. It is used when true 3D positioning is demanded in the interaction, e. g. dragging an object to translate it.

Free pointing displays an indicator at the 3D position, with a line descending to the bottom of the image (see Figure 4 on page 7); typically this intersects objects or a “floor plane” to give a cue to the indicator’s depth. The indicator is commonly the sparkle which is being dragged.

FIGURE 4 While controlling a free position, as during translation, a line descends from the sparkle to the floor plane, providing an additional depth cue.



Some interactions, such as translation, may use the change in the pointing device orientation control to rotate the selected shape.

1.3.3.4 Align Pointing

Align pointing is a variation on free pointing, and is used when the manipulated feature has strong basis vectors, which can produce an unpleasant feeling of cross-coupling. This is true of the scale widget handles.

Where free pointing intersects the line-of-sight ray with a plane parallel to the image plane, align pointing uses an actual world-space plane to intersect against. This plane is derived from the feature being manipulated. Typically the normal to the plane is chosen to be as orthogonal to the image plane as possible.

Align pointing is visually identical to free pointing.

1.3.3.4.1 Pointing Devices

The four pointing modes may be controlled using a variety of pointing devices. If adevice with three positional axes, two axes, X and Y, are always devoted to moving the cursor. The third axis, Z, is used only during free or align pointing modes.

The mouse may emulate a 3D pointing device by the use of a keyboard modifier to switch the vertical axis between Y and Z axes. With the modifier key up, the mouse acts normally. When the modifier key is held down, and the application is in free or align pointing mode, horizontal movement of the mouse is normal, but vertical movement causes the controlled point to get farther and nearer.

Some devices control both position and rotation in 3D. The orientation may be used by some tools during free and align pointing modes, such as translation.

One of the advantages of this pointing scheme is that it scales gracefully from use with a mouse, to a true 3D pointing device, to high-end six-degree-of-freedom devices. The API for the pointing modes isolates the application from concerns at this level, so users can confidently plug-and-play with new devices.

This pointing model requires that the 3D pointing device be first and foremost a good 2D pointing device—you should be able to choose from menus and palettes without having to drop the 3D device and pick up the mouse.

1.3.3.5 Pointing Modes

Switching between the pointing modes uses a combination of user control and application control. A tool is designed to use a particular pointing mode (or modes) at a particular phase of its use. For example, the spot light creation tool starts (specifying the light-from position) in free pointing, with pick pointing enabled. When the light-from is specified by a click, the light-to location is again specified using free pointing, with pick pointing enabled. For the final setting of angle, guide pointing is used. In a second example, the scale tool starts in pick pointing, allowing the user to indicate a scale widget handle or an object in the scene. If a handle is clicked, then free pointing is used, pick pointing is disabled, and guide pointing is accessed via the shift key. If an object is clicked, free pointing is used, with pick and guide pointing enabled.

The user can control the particular pointing mode, within the enabled pointing modes. Constrained interactions, typically implemented with guide pointing, are accessed using the shift key.

The choice between pick and free/align pointing relies on indicating which are available at the moment, and providing a switch between the modes. In addition to menu items or buttons to switch between the modes, a keyboard shortcut must be provided, so that the switch may be thrown during a click-drag interaction.

The New Era application provides both menus and buttons to indicate and control the pick vs. free/align pointing modes. The menu items are in the Scene menu, “Point at Shape” for pick mode, “Point in Space” for free, and “Change Pointing Mode” with a command-T equivalent to toggle between the two modes. This takes care of the control part of the problem, but leaves the indication of which modes are available and which is in use, somewhat under-supported. To provide this indication, a pair of icon buttons in the bottom of an interactive window represent pick and free/align pointing. These icon buttons are highlighted (inverted) or dimmed as appropriate. They can be clicked to change the pointing mode. This arrangement should be used by any application that doesn’t have a compelling reason to deviate from it.

The choice between free and align pointing is typically one made by the tool designer, not the user. At the user interface, the two modes are presented identically.

There are times when guide pointing is used in lieu of pick pointing, for example when dragging a handle of the rotate widget. It is presented in the user interface as Point at Shape, even though it is actually implemented using guide pointing.

1.3.3.6 Scalability

In a simple application, it may be best to not provide a switch between pick and free/align pointing. This reduces complexity at the cost of some functionality. The menu items, buttons and keyboard shortcuts for switching between pick and free/align pointing modes can be eliminated.

1.3.4 Widgets

In a 2D graphical editor, the “handle box” indicates selection and provides features that the user can activate to control spatial parameters, like position and size. When spatial parameters are manipulated in a 3D scene, shapes must be added to the scene to indicate selection and to provide features that the user can activate to control the parameters. The combination of 3D graphical shapes and behavior when the user activates them is called a widget. The application may use the widgets provided by QuickDraw 3D, or design new ones to support new tasks.

The 3D graphical shapes that a widget displays indicate several things. They indicate which object(s) are affected by the widget. The shapes indicate where to click to operate the widget. They also indicate the effect of widget on the selected object—which parameters that the widget will effect. The 3D graphical shapes are visual affordances that show the potential for action on an object.

Because they are displayed in the scene along with the objects that compose the scene, widgets must look different from the scene objects. However, the typical application can’t restrict kinds of objects that may be in the scene, in shape, color or rendering style.

One way to make widgets look different is to base them off a color scheme that the user can choose to be different from the scene objects. Portions of the widget may be colored to indicate a color that the widget controls. For example, a widget that represents a blue spot light source would be partially blue. Other colors would be determined by the color scheme chosen by the user.

The color scheme determines colors that can be used for parts of a widget or other 3D user interface object. Two colors are given for “foreground” parts—those that are prominent and manipulable. The colors might be used directly, or as the basis for generating other colors, e. g. as two points on a line in a color space. Two other colors are given for “background” parts—those that are recessive or structural. For example, the walls of a room might be made from tiles of the “background” colors. The non-manipulable edges of a bounding box might also be of a background color. Another color in the color scheme indicates the state of persistent selection. A selected widget should indicate its state by having patches of the selection color, rather than being completely turned to the selection color. A widget such as the spot light source might look like when selected. The color palette allows widgets to harmonize visually with one another, to contrast with the user’s objects, and to indicate their function and state.

Many widgets are box-shaped, based on the bounding box of the shape that they operate on. The shape’s bounding box should be found in its local coordinate reference frame, and then transformed to the world coordinate frame.

1.3.4.1 Widget Design Process

This section describes the process of designing widgets, using the point light as an example.

1. Understand the parameters that need to be adjusted. For a point light, it's just the parameters from the QuickDraw 3D API: on/off switch, brightness, color, shadow switch, attenuation type, location. Consider which of these will be presented to the user, and which can be set to fixed values. The on/off switch may be set on always, for example, eliminating it from the user interface.
2. Try various interpretations of the parameters. In the point light, brightness and the color's intensity are redundant. It's typically not a good idea to present redundant controls when no value is added. The UI could present the color picker to choose an RGB value for the color, and fix the brightness at one. Alternately, a color picker that ignored brightness could be used, in conjunction with a brightness slider. The alternative that better matches the user model should be chosen—perhaps with informal user testing. In this case the second alternative is chosen since light color and intensity "feel" like independent variables.
3. Interpret parameters spatially. Another pair of point light parameters that interact are brightness and attenuation type. Specifically, when attenuation is not flat, brightness must be increased (to values over 1.0 typically) by an amount that is related to the scale of the scene. Consider a point light with inverse-linear attenuation, and a sphere 10 units away and one unit in diameter. The intensity of the light is attenuated by 0.1 when it hits the sphere. Then consider changing the scale of the scene—keeping its appearance constant—so the sphere is 1000 units away and two in diameter. Now the intensity of the light is attenuated by 0.001. The interaction between scene scale and brightness is more pronounced with inverse-square attenuation.

   The wrong solution is to provide a brightness slider that can crank the brightness to huge values, and let the user compensate for the effect himself. It is clearly more desirable to provide an interpretation of the parameters that masks this interaction. One way to do this is to measure the scene in some way (e. g. diameter of scene bounding sphere) and scale the brightness parameter by this measure, adjusted by the attenuation type.

   Alternately, we can provide a spatial control for each point light to measure some relevant distance in the scene. We chose to provide a "light-to" position that the user can place in the scene relative to the light location. The distance between the light location and the light-to position is used to scale the brightness. Providing a spatial control that contributes to the brightness parameter presents a confusing aspect of lighting in a sensible way.

   Parameters that don't have a spatial interpretation should be placed in the 2D portion of the user interface, typically floating palettes and menus. The point light widget has a corresponding palette to access the brightness slider, color picker, attenuation type and shadows switch. The location is clearly a spatial parameter, as is the light-to position.
4. Reuse common forms between widgets. Certain design elements repeat throughout the widget designs in the HI guidelines, providing a smoother learning curve. For example, the light-to position appears in the widgets for the point, spot and directional lights; the spot light functions much like the camera widget.

5. Consider splitting functionality between multiple widgets. For example, the functions of translating, rotating and scaling could have been combined into a single widget. Such a widget design results in many handles to access all those functions. We split the widget into two simpler widgets, scale-translate and rotate-translate. Whether to combine or split functions depends on the task breakdown (how intermixed the functions are in typical use) and the complexity of the widgets.
6. Use the pointing modes to control the spatial affordances. Determine for each affordance on the widget, the pointing modes that will be used for the application modes "point at shape," "point in space" and with the shift key held down. Which of the application modes will be enabled? What pointing mode is used for each? For pick pointing, what shapes are pickable? For guide pointing, what guides are presented, and which are visible?
7. Design the appearance of the widget. In general, a positional handle is represented by an octahedron and a rotational handle by a sphere. Anything in the widget that is structural is drawn with lines, which are not selectable; a widget's elements should usually be joined by structural elements, and appear to surround the selected shape. The color scheme defines the colors of the handles and structural elements.

At this point, several design alternatives have been examined. We are now at the midpoint of an iterative design cycle. Now discard the obvious failures, ending up with a few potential designs. It may be best to set them aside, until other related widgets take form. Prototype the few designs, and perform informal user testing to evaluate them. Use the user feedback to select a starting point for the next stage of refinement, and to identify problems with the design.

1.3.5 Views and Scenes

One of the key user needs in 3D graphics is to understand the arrangement and relationships of objects. To help the user understand 3D shapes and their arrangement it is useful to distinguish between *scenes* and *views*. A scene contains a collection of shapes, lights, cameras, shaders, etc. In a sense a scene is like a document, and can be operated on in similar fashions, e.g. a scene can be created, an existing scene can be opened, and a scene can be closed. The application may portray the same set of objects, or scene, in multiple views. A scene is composed of one or more views and a view is a view onto the scene from a given camera position.

A view encompasses more than just a camera position: it also includes image settings and rendering settings. Image settings include such things as image size and resolution, anti-aliasing settings, backdrop image or color, overlay image, atmospheric effects, etc. that may vary from view to view. Rendering settings include the choice of final renderer, optimization settings for the renderer, and custom renderer controls. Because views include all this state information, they are persistent in the scene/document.

Some settings apply scene-wide—they affect all views onto the scene. These include the color scheme used for the interactive widgets, the ambient light level, etc.

The typical modelling/arrangement application would allow multiple views (and scenes) to be open at the same time.

A final rendering is called a snapshot. A snapshot of a view is taken by a menu command, “Take Snapshot,” or by printing the view. A snapshot rendering may take a while, so it is presented as a background process that operates on a (logical) copy of the scene. A snapshot contains only the scene elements, without the widgets that allow interaction with the scene.

So that the user may better anticipate the appearance of a final rendering, a view can be switched between the normal “interactive” mode and a “preview” mode. In interactive mode, widgets show selection and allow interaction, and the “interactive renderer” is used. No widgets are shown in a preview view, interaction is not possible, and the view’s chosen renderer is used.

1.4 Tools and Techniques

This section discusses many different techniques and widgets for interacting with the various properties of shapes. A typical application would choose between these techniques by providing a tool palette. One of the tools in the palette is active at any time. Objects that are selected are indicated using the widget for the current tool. Changing the tool changes the widgets used to indicate selection. Some tools may be accompanied by a menu to change characteristics of the tool or to change attributes of the selected shape.

1.4.1 Tool Model

A typical graphic editor, whether 2D or 3D, uses a tool model that allows the user to chose the function of clicking and dragging in the document. Some tools create new objects, such as MacDraw’s rectangle tool or oval tool. Other tools edit existing objects, e. g. in MacDraw, the arrow tool, the reshape mode and the rotate mode.

The editing tools collectively control all the spatial aspects of a shape. All those controls—position, scale, rotation and control points in MacDraw—aren’t available at the same time. Instead, they are divided into multiple tools. This divide and conquer strategy applies even more strongly to 3D user interfaces.

Tools should all be available from a single tool palette. The palette controls and indicates the current tool, which defines many things in the user interface:

- The pointing modes
- Which application pointing modes (point-at-shape vs. point-in-space) are enabled
- Which application pointing mode is in use by default
- Which objects are selectable
- The effect of clicking and dragging in the scene
- Whether the selection is maintained or emptied
- The appearance of the selection indicator (widget)
- Any relevant menu

The tool model is appropriate for a sufficiently complex editing environment. A simpler application may have few enough parameters to present them all at once. An application that is simpler still may not even require the notion of selection.

1.4.1.1 Tools and Selection

Different editing tools may change the meaning of what is selectable. For example, the scale tool allows selection of any shape, the mesh tool selects any mesh face, vertex or edge, and the decal rotate tool selects only decals.

Editing tools should keep the selection unchanged, to the extent possible. When changing between the scale and rotate tools, the selection should remain unchanged. When changing from decal rotate to scale, the shapes that host the selected decals should be selected.

When a creation tool is chosen, the selection is typically emptied. When the tool finishes, it should leave the new object selected, typically switching to the scale tool. The MacDraw “sticky tool” thing can be lifted—double-click on a creation tool in the tool palette locks the tool “on.” Instead of reverting to the scale tool after creating the shape, the creation tool is kept active. The created shapes are left unselected by this process.

1.4.1.2 Tool Menu

Some tools have a menu to provide access to commands and non-spatial controls. For example, the mesh tool or the decal tools. This menu should be appended to the menu bar while the tool is chosen.

1.4.2 Transform Tools

The transform tools allow the user to move, scale and rotate a shape. There is no explicit translation tool; translation is available from many tools by click-dragging on the shape. The exceptions are tools that need to use the shape as an affordance (e. g. the mesh tool, or the orbit tool below) for other interactions. The scale, rotate and orbit tools are typically selected via a tool palette.

1.4.2.1 Translate

Translation is one of the few (or the only) place where orientation from a pointing device is used.

In tools that support translation, click-dragging on the shape moves the shape in space. When the user does so, translation defaults to free-space pointing mode. If the dragging device reports orientation then the orientation is changed as well as the position.

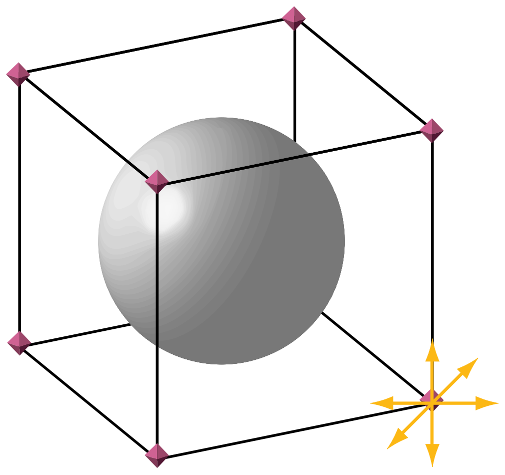
While the shift key is pressed, three guide lines appear through original click position, aligned to the shape’s axes at click time. Orientation from the device is ignored.

The point-at-shape pointing mode is enabled during translation. When the user toggles to point-at-shape mode, the dragged point moves to the indicated position. In addition, the dragged shape is rotated so that the back side of the dragged surface is normal-matched to the front side of the indicated position. This allows a quick and easy form of alignment. If the dragging device reports orientation, then the twist about the normal can be controlled; otherwise, the twist is minimized. When nothing is pointed at, the effect is the same as free-space pointing.

1.4.2.2 Scale

The scale widget consists of a wireframe bounding box, with a handle at each corner for scaling (see Figure 5 on page 14). The bounding box is aligned to the shapes “local coordinate frame.” The box itself can’t be grabbed. The shape itself affords translation (above).

FIGURE 5 The scale widget provides eight handles for scaling. Dragging the object itself translates it.



Click-dragging on a handle scales the shape within the bounding box, relative to the opposite handle, using the aligned free-space pointing mode, aligned to the plane of the bounding box that is most facing the camera. Point-at-shape is disabled.

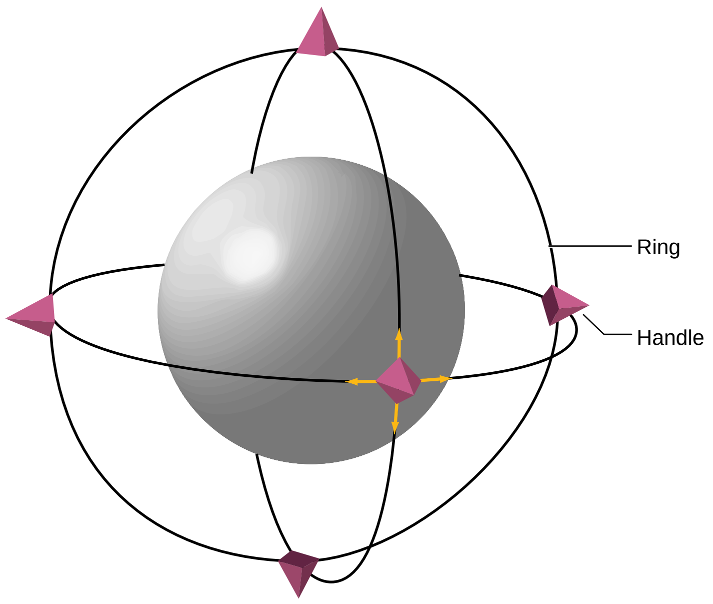
The shift key accesses both uniform and single-axis scaling. While the shift key is down, three guide lines pass through the original handle position, aligned to the shape’s axes. A fourth, visually-distinguished guide passes through the same position and through the opposite handle, affording uniform scaling.

If multiple shapes are selected, each is indicated by its own scale widget, and they all travel the same amount in the same direction as the dragged one.

1.4.2.3 Rotate

The rotate widget consists of three round, mutually orthogonal rings encircling the shape (see Figure 6 on page 15). At the six ring intersections, a handle. The rings are aligned with the shape’s local coordinate frame. The handles afford free rotation; the rings afford constrained rotation. The shape itself affords translation (above).

FIGURE 6 The rotation widget provides six handles to rotate the object. Dragging the object itself translates it.



Click-dragging on a handle rotates the object to keep the handle under the cursor. Point-at-shape is indicated, even though the implementation uses guide pointing; free-space pointing is disabled. If the handle starts on the visible part of the sphere, it is kept there; if it starts on the far side of the sphere, then it stays there. If the handle is dragged outside the sphere then the handle is kept on the sphere, but as close to the dragged position as possible. While the shift key is down, two orthogonal guide circles appear through the original handle position.

Click-dragging on a ring rotates the object about the axis of the ring. Point-at-shape is indicated, even though the implementation uses guide pointing; free-space pointing is disabled. The grabbed point rides along the ring, as close as possible to the cursor. While the shift key is down, guide points at every 30° and 45° from the grabbed point are shown.

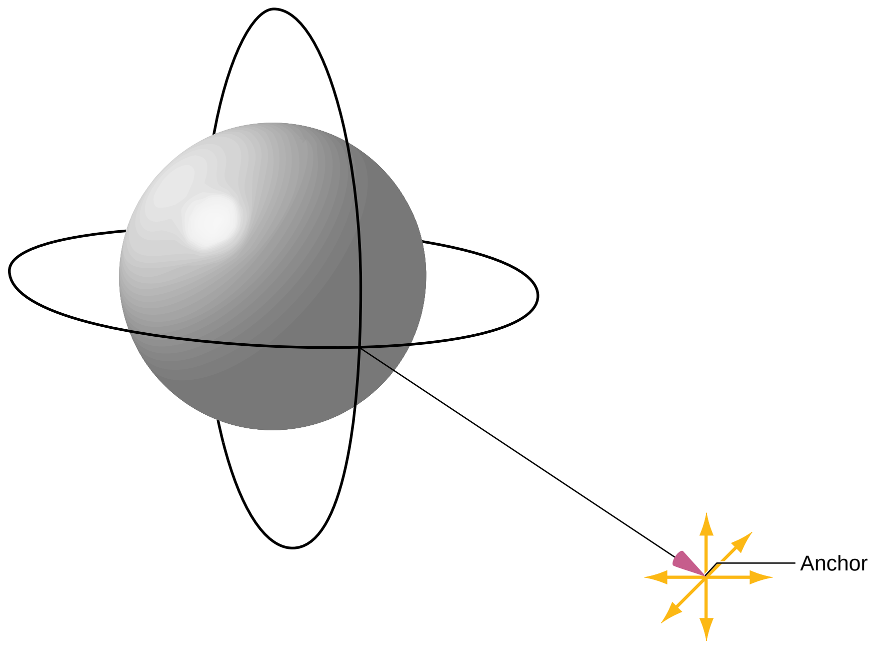
If multiple shapes are selected, each is indicated by its own rotate widget, and they are all rotated about the same world-space axis, by the same amount.

1.4.2.4 Orbit

The orbit tool provides access to constrained motion about an anchor.

The orbit widget consists of an anchor point and an indication of which shape is selected (see Figure 7 on page 16). The shape is its own affordance for constrained motion, unlike scale and rotate.

FIGURE 7 The orbit widget provides an anchor point that is the center of the constrained motion.



Click-dragging on the shape translates the shape on the sphere centered on the anchor. Point-at-shape is indicated, even though the implementation uses guide pointing; free-space pointing is disabled. If the shape started on the visible half, then it is kept there; if it started on the back half, then it stays there. If the point is dragged outside of the sphere then it is kept on the sphere, as close to the dragged position as possible. If the shift key is up, then the shape is rotated so the same face pointing toward the anchor. Twist is minimized. If the shift key is down then the orientation remains unchanged from the click-time orientation, so the orbit is merely a translation.

Click-dragging on the anchor moves the anchor using free-space pointing. Point-at-shape is enabled, allowing the placement of the handle on surfaces in the scene. The shift key shows three guide lines through the center of the shape, parallel to the local coordinate axes.

If multiple shapes are selected, each is indicated by its own orbit widget, they all have the anchor displaced by the same world-space vector from the centers of the selected shapes, and each travels relative to its own anchor.

The orbit tool is for a sophisticated user model, and could be eliminated from a simpler application.

1.4.3 Mesh Editing

The underlying representation for some kinds of 3D models is the polygonal mesh.A mesh is a representation of a free form shape. It is presented as a surface, rather than a volume or solid. The mesh may be open or closed, and may be topologically complex. Meshes are generally smooth, but can contain sharp edges

In some circumstances it is useful to expose this representation, and to allow the user to change it directly. The down-side of this is that the application must expose the user to such (typically) unfamiliar concepts as vertices, edges and faces. Mesh editing may be broken into two tasks: geometrical editing and topological editing. Both require that faces, edges and vertices be selectable. They must therefore be represented graphically.

Geometrical editing involves the selection of vertices or faces, the application of material properties (e. g. color) to them, and their placement. The application's material picker controls the material characteristics for the selected faces and/or vertices. The selected vertices and faces may be moved by click-dragging.

Topological editing involves creating and deleting vertices and faces.

1.4.3.1 Mesh Tool

The mesh tool provides basic editing functions for a mesh object's elements. Ultimately, this is much more primitive than what we would like to see applications provide: the ability to describe and control shape at a high level.

The mesh tool can be used as a model for other complex-surface modelling representations, such as NURBs. We do not expect to provide guidelines for NURBs editing until a later release of QuickDraw 3D.

The mesh tool provides low-level access to the faces, vertices and edges that comprise a mesh. These elements are selectable, individually or in combination. The shape of the mesh may be altered by changing the positions of the vertices. The topology of the mesh may be altered by dividing or coalescing faces, or by joining boundary edges, either within a single mesh or between meshes.

1.4.3.2 Meshes and Shapes

A menu command is provided to convert a shape (e. g. a cone or an extrusion) to a mesh. "Convert to Editable Form." For wireframe primitives, such as the oval frame, the editable form is a poly-line instead of a mesh. This is a one-way conversion—there's no chance of using the primitive controls (e. g. endcap settings or the extrusion path) once the conversion has been made.

Some functions in an application will force the conversion to mesh. Since this is a one-way street, this should be noted, typically with a warning dialog.

A mesh shape consists of a single, connected fabric of faces. If a mesh is broken into disconnected pieces, then multiple shapes result. (The new shapes inherit the local coordinate frame of the original.)

Two meshes may be coalesced into a single mesh by stitching their boundaries together. (The resulting mesh has the local coordinate frame of the first-selected of the two pieces.)

1.4.3.3 Mesh Tool

The mesh tool is accessed by working "inside" a mesh shape. Using the mesh tool, the user can select faces, vertices and edges of the mesh shape(s) that s/he is working inside. The correct things happen for click, shift-click, box selection, etc.

The mesh tool's Shape Controls Palette consists of the a radio pair, Edge: Smooth / Sharp, in addition to the usual precision slider, lock radio buttons and other elements. The mesh tool's menu contains the following commands:

- Show/Hide Structure
- Blend (may be checked)
- Select More
- Divide
- Extrude Face
- Extend Boundary
- Connect Boundary(ies)
- Stitch Boundaries

Many of the operations treat interior edges (those with two adjacent faces) and boundary edges (those with only one) as different.

1.4.3.4 Selection

The user can switch whether the structure of the mesh is visible or not. The structure is shown by drawing the wireframe version of the mesh over the solid version.

As the user moves the cursor over any mesh object in the scene (both the selected one(s) and those not selected) the indicated feature is prehighlighted with a sparkle that corresponds whether it is a face, edge or vertex. A face is prehighlighted with a sphere handle at the face center, and its edges if the structure is not shown. An edge is shown with a cylinder or box lying along the edge (depending on whether the edge is smooth or sharp), and the edge line if the structure is not shown. A vertex is shown with a octahedron handle. The prehighlight geometries are shown in the prehighlight color.

Selection of a feature is indicated with the prehighlight geometries, except in the normal handle colors. When a selected feature is prehighlighted, the colors change again.

1.4.3.5 Materials

Materials may be applied to or adjusted for the selected features. If the selection is "blended" then the material change is 100% for the most central feature, and blends toward 0% for features closer toward the unselected part of the mesh. While the QuickDraw 3D API provides the notion of shader and attribute inheritance from the geometry level, and sharing may happen at the element levels, that model is not promoted to the user. Instead, the user model is that each face, vertex and edge may have a different material applied to it. The application can (and should) use the sharing and inheritance to implement this, though.

Since the QuickDraw 3D API applies shaders at the face level only, an attempt to change the material class for a vertex or edge is denied (using a staged alert). The same applies when attempting to change the material class for faces while blending.

1.4.3.6 Reshaping

Click-dragging on a face, vertex or edge selects it, then moves the adjacent vertices using free-space pointing. Point-at-shape is disabled. If the selection is “blended” then the movement is 100% for the dragged feature, and blends toward 0% for features closer toward the unselected part of the mesh. If the pointing device provides orientation, then the selected feature(s) are rotated in their rigid configuration. With the shift key down, guide lines appear through the click point, along the normal(s) to the surface at the point.

1.4.3.7 Extend Selection

The Extend Selection command selects more of the same kind of features as are already selected. For each selected face, the edge-adjacent faces are also selected. For each selected interior edge, the vertex-adjacent edges are added to the selection. For each selected boundary edge, the adjacent boundary edges are selected. For each selected vertex, the edge-adjacent vertices are added.

1.4.3.8 Smooth/Sharp Edge

Meshes generally appear smooth. An edge may be made sharp or smooth by selecting it and changing the “smooth edge” state. Smooth edges are represented with shared vertex normals. Sharp edges are represented using normals assigned to the mesh corners to express the discontinuity.

1.4.3.9 Delete Feature

A feature may be deleted by selecting it, and using the Edit menu “Clear” command, or the Delete or Clear key. A face is deleted by making a hole where the face once was.A vertex or interior edge is deleted by coalescing the adjacent faces (delete all faces, then recreate a single face). An exterior edge is deleted by removing the adjacent face.

1.4.3.10 Divide Feature

A command is provided to subdivide the selected feature(s). For each selected triangular face, a vertex is created in the middle of the face, offset to accommodate the curvature of the surface, and the face is replaced by triangles that connect the new vertex to the face’s edge.For each selected face with more than three edges, the face replaced by equivalent triangles.For each pair of vertices that share a face (but not an edge), the face is replaced by two that share an edge between the vertices

For each selected edge, a vertex is created in the middle of the edge, offset to accommodate the curvature of the edge. The two adjacent faces (or one, if it’s a boundary edge) are replaced by a faces that use the new vertex.The new edges are left selected.

1.4.3.11 Extrude Face

A selected face(s) can be pulled out of the mesh, creating a bunch of new quadrilateral faces. For each selected face, delete the face, create a new face displaced from the original, with the same shape, and create quad faces between the old and new edges. The direction to pull out is along the normal, reflected to point toward the camera. The amount of displacement is computed from the edge lengths. Adjacent faces should be dealt with correctly. The new faces (not the quads) are left selected.

1.4.3.12 Extend Boundary

A command is provided to create new faces outward from the boundary edges of a mesh. For each selected boundary edge, a new, quadrilateral face is created, in the same plane as the adjacent face.

1.4.3.13 Connect Boundary(ies)

This command closes a boundary with faces. If only one boundary edge sequence is selected then the whole boundary is capped. If two boundary edge sequences are selected then faces are made that connect the sequences. When one boundary edge sequence is selected, that boundary is capped off. The new faces are left selected.

When two boundary edge sequences are selected, this command creates faces to join two sequences. This can join two shapes into a single mesh, or can join two boundaries of a single mesh, or can split a single boundary in two. If the two boundary sequences are on different meshes, then the second-selected mesh is gathered into the first-selected mesh, without changing its world-space position. The new faces are left selected.

1.4.3.14 Stitch Boundaries

Pulls two boundaries together, joining them. Very similar to the above, except that instead of making faces, it actually moves vertices. The stitched edges are left selected.

1.4.4 Appearance

The preceding section discussed controlling the shape of an object. This section discusses controlling the way an object looks, including its surface characteristics and how light hits it. Because the two are very linked, both in interaction style—selection followed by operation—and in real effect, many issues have already been covered.

Any object in the real world is made out of some material: wood, plastic, glass, fabric, etc. To create a desired effect, the user must be able to change the materials of a whole object or just parts of it. Traditional 2D painting and drawing tools have provided colors, patterns and sometimes gradients as material characteristics. Materials in 3D tend to be much more complicated than in 2D, since they mimic the characteristics of physical materials. Rather than choosing the many parameters individually, it is often desirable to bundle them into a *material*.

Materials may be stored in a palette, keeping common materials handy. Materials in a palette must be identified in some way. A name is one good way. Also, a material swatch should be provided—a view of the material, applied to some standard object such as a sphere, set in some standard environment. The swatch identifies the material in an abstract way; it may be desirable to be able to browse materials by looking at material previews that show the selected object in the current scene configuration, as it would appear with the material applied to the selected portion. The material palette may be a simple scrolling list, or it may be broken into categories such as woods, rocks, metals, plastics, etc.

If the user can’t find the desired material in the palette, or the selected material is almost, but not quite, right then facilities must be accessed to adjust the material parameters. The user may pick one or more materials as a basis. If one is chosen then controls for the parameters must be presented. A swatch or preview must be visible so that the effect of the abstract settings may be viewed. It is often desirable to provide multiple ways to change the same set of parameters. For example, the brightness and sharpness of the specular highlight might be controlled by handles displayed in the swatch; other handles may control the contributions of ambient and diffuse light. To make the values of the parameter sliders more relevant, each end could be accompanied by a swatch showing the value at the extremes.

QuickDraw 3D’s material parameters are very abstract. For example, the “specular color” value determines the shininess of the material for both specular highlights and reflections, as well as any color imparted by the surface for highlights and reflections. This might be more logically represented by a slider for “shininess” or “reflectivity” and a color (and picker) for “reflective color.”

Some values are naturally bounded, such as between zero and one; a simple slider may be used for such values. Others may take on a much wider range. For example, index of refraction may be between zero and ∞, but is typically between zero and one. A non-linear slider may be used for these values. Such a slider may allow fine selection of small values, and coarse selection of large values. One kind of behavior can be implemented using a traditional slider to control the power to which a constant is raised. Alternately, a piece-wise linear function could relate the position to the value.

A slider should usually be accompanied by a numeric field, so the user may choose or type a specific value. If several of sliders control related functions, then a swatch showing the currently chosen values should be shown.

1.4.4.1 Surface Maps

Texture mapping, bump mapping and displacement mapping are rendering techniques that have been used to serve two functions. First, they represent visual complexity in the absence of the model’s geometrical complexity. Second, they are a way of incorporating a 2D image into a 3D model. Both are valuable functions from a user’s perspective, however the interface to them may not be the same thing. For example, a texture and displacement map may be created without the user’s knowledge to represent fine surface detail by a modelling application. That same modelling application may allow a texture map to be imported and placed manually on the object’s surface. Since the former use of texture maps is primarily geometrical, it will not be considered here, separately from the modelling section. The latter application—incorporating a 2D image—is the subject of this section.

Traditional techniques map an image onto the entirety of a shape. To do this, a parametrization of the shape must be generated, assigning to each point on the shape a coordinate in the image plane. Some geometrical representations, such as NURBs, have an implicit parameterization that is used. For representations that have no implicit parameterization, one is constructed using a control shape such as a sphere or a cylinder. The mapping of the image onto the control shape is predefined by the application, and the placement of the control shape determines how the texture projects onto the target shape. Wrapping a single surface map around a whole object is rarely the user’s desired effect. Instead, a technique that doesn't expose the user to concepts like parameterization and control shapes is proposed here.

The simplest form of surface mapping is incorporating a planar image into a 3D scene, such as a billboard, or art to hang on a wall. A more complex operation is to wrap a surface map onto a shape, such as placing a label on a bottle, or embossing a logo. The image may be imported or pasted in, or it may come from an image or surface map library. Several things must be determined to place a surface map onto a shape:

- The portion of the shape to affect
- Whether to take the image's colors or to use its luminance as a bump or displacement value
- Whether the image is used singly or as a pattern
- How the planar image is wrapped onto the shape
- The scale of the image relative to the shape
- The scale of the bump or displacement relative to the shape

When an image is pasted or imported into a scene, it goes into the selected shape. If none is selected then the image is an independent, planar object, with some sensible position, orientation and scale. If a shape is selected then the surface map is attached to the shape. Whether the image is used as a texture, bump or displacement map may be controlled by menu selections. Other menu commands control whether the image is used singly or as a tile to repeat as a pattern.

Other menu commands control the way that the image plane is wrapped onto the shape. One mode, “Project Parallel,” causes the image to appear on the shape as if by a distant slide projector. A second mode, “Wrap as Decal,” performs a simple decal wrap of the image plane. The plane that cuts horizontally through the center of the image is intersected with the shape, creating a curve along which the horizontal centerline of the image is placed. The same happens for the vertical centerline. A third mode, “Custom Wrapping,” allows the user to select points on the image plane and pull them to points on the shape. Detents are provided that maintain the scale of the image.

The attached surface map widget controls the placement and scale of the image, much in the way that the translate/rotate/scale widget works, but with some additional constraints. The image may be moved on the surface so that its center point is tangent to the shape. It may be rotated freely within the tangent plane. The image may be scaled, and the displacement or bump effect is controlled by scaling orthogonal to the tangent plane.

1.4.4.2 Volume Maps

Where a surface map simulates a veneer laid on a surface, volume maps simulate the surface effects of a solid material. For example, a woodgrain cylinder may be made by wrapping a cylinder with a veneer, or by carving a block of solid wood. A volume map is applied to a shape much in the way that a surface map is. The position, orientation and scale of the volume map is controlled by an attached volume map widget, which functions like the translate/rotate/scale widget.

1.4.5 Lighting

One component of the appearance of a scene is the way it is lit. Typically lighting is controlled by specifying several idealized light sources. These may include spot lights, point source lights, directional lights, and an ambient light. While the effects of the lights are visible in the scene, the lights themselves are not.

Lights are represented in the scene by widgets. The widgets provide control of spatial parameters, and enable selection of lights. Widgets are visible in an Interactive View if chosen, but are not visible in a Preview.

Ambient lights are not represented in the scene, as they have no spatial component. Conceptually, there is a single ambient light.

A light control palette controls the non-spatial parameters of the selected lights. There is a single palette, whose contents depend on the type of the selected light

1.4.5.1 Light Control Palette

The light control palette provides control of the non-spatial parameters of the selected light.The controls available in the palette depend on the type of the selected light—directional, point or spot. If no light is selected then the palette provides control of the ambient light (see Figure 2 on page 21). If the selected lights are of different types then the light control palette is empty, while if they are of the same type then the appropriate one is shown.

FIGURE 8 The ambient light control panel changes the color and brightness of the scene’s ambient lighting.

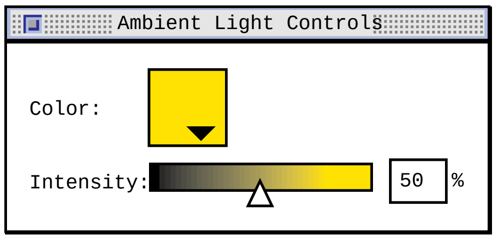


FIGURE 9 The directional light palette additionally controls whether the light casts a shadow.

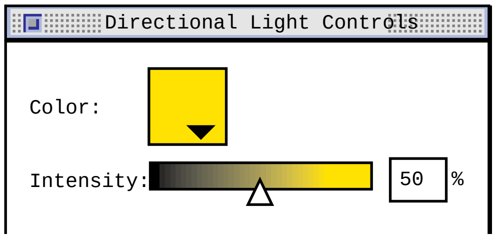


FIGURE 10 For a point light, the distance attenuation, or falloff, is controlled.

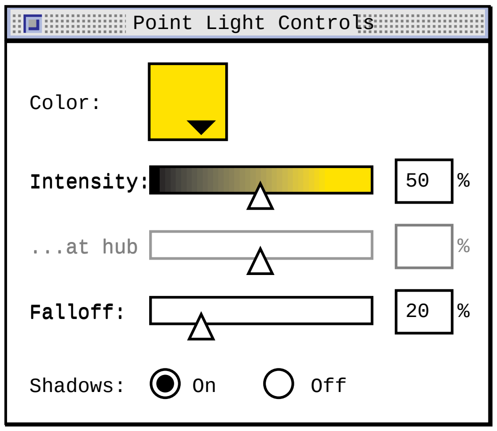
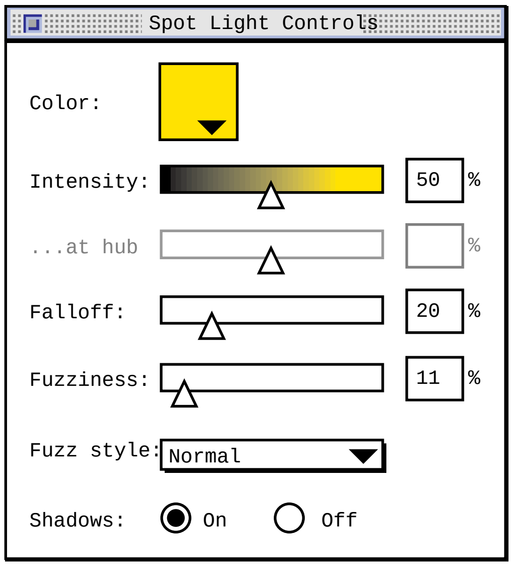


FIGURE 11 A point light also has characteristics that control the fuzziness of its edges.



Clicking on color field brings up the “2D color picker” which accesses hue and saturation, but leaves value at its maximum setting.

1.4.5.1.1 Intensity

The light intensity slider in the light control palette controls ranges from no light through “unit” light to “over-lit.” Over-lighting is useful for some visual effects. The white range of the slider is linear between zero (left) and one (right) intensity. The hatched range of the slider is for over-lighting. The left side of the hatched range is one intensity, the right side should be some reasonable maximum, such as five, and the intensity should be linear between them.

Ambient intensity is not allowed to be over-lit. The hatched area should be grayed out, and the slider position should occupy only the white range of the slider.

When a light includes a distance attenuation function, it becomes difficult to talk about the brightness of the light without taking into account the scale of the scene (that is, the measure of distance). For this reason, we include in the design of the point and spot lights a second position in space, the "light-to" position (the first is the position of the light, the "light-from" position). It is at the light-to position where the intensity value in the slider is interpreted. (In addition, the light-to position controls the spot light direction, and makes it very analogous to the camera widget.)

The fall-off slider in the control panel pertains to features in the QuickDraw 3D 1.5 API, namely the ability to set the distance attenuation as an inverse quadratic function. For 1.0 applications, a simple pop-up with "none," "medium," and "full" will suffice.

For 1.5 applications, the fall-off slider controls the inverse quadratic by the following functions, given the slider position, *f*, as ranging from 0 at the left to 1 at the right.

let $k = 1 / (1–0.75f)$

let p be the distance between the light and the light-at position

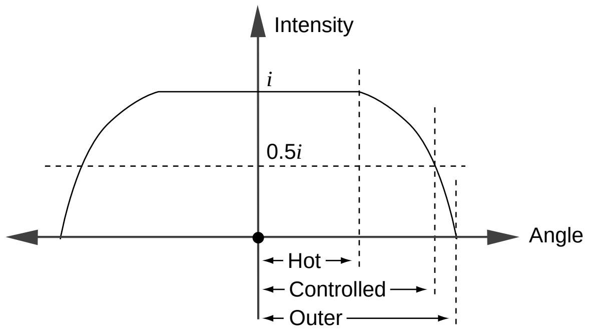
<u>if $1 \le k \le 2$</u>	<u>if $2 \le k \le 4$</u>
$c_0 = 2–k$	$c_0 = 0$
$c_1 = (k–1) / p$	$c_1 = (4–k) / 2p$
$c_2 = 0$	$c_2 = (k–2) / 2p^2$

This gives a nicely behaved family of curves (nonintersecting, positive, negatively sloped) that interpolate between constant, inverse linear and inverse squared falloff curves. The distance component is normalized by the use of *p*.

1.4.5.1.2 Spot Light Angular Attenuation

The fuzziness control for the spot light controls the light's angular attenuation. The light's hot angle and outer angle are functions of the fuzziness and the angle that is controlled by the widget. The widget-controlled angle controls the angle at which the brightness is half of the full brightness (see Figure 2 on page 22). The fuzziness slider controls the fraction of that angle that is occupied by the angular fall-off between full brightness and half brightness—with the slider to the left most, the hot angle is the same as the widget-controlled angle; at the right most, the hot angle is zero.

FIGURE 12 The width of a spotlight is controlled by its widget. The degree of angular attenuation is controlled by a slider in the spotlight control panel, which controls the ratio of the hot angle to the controlled angle.



The pop-up for controlling the fuzz style should be omitted by all but the most sophisticated users. If omitted, then linear should be used. The items in the pop-up should be “normal,” “smooth,” and “sharp” for linear, cosine and exponential.

1.4.5.2 Light Widgets

The ambient light has no light widget. The widgets for directional, point and spot lights are based on three common affordances: a light-from position, a light-to position, and a handle to move both positions together. Preliminary graphic designs for these widgets are attached. When selected, the widget appears with full graphics. When unselected, only the light-from affordance is shown. The light-from geometry is shown in the light’s color.

FIGURE 13 The selected spotlight widget has affordances for moving the source of the light, the center of the spot, or both together. Other handles control the angular width of the spot.

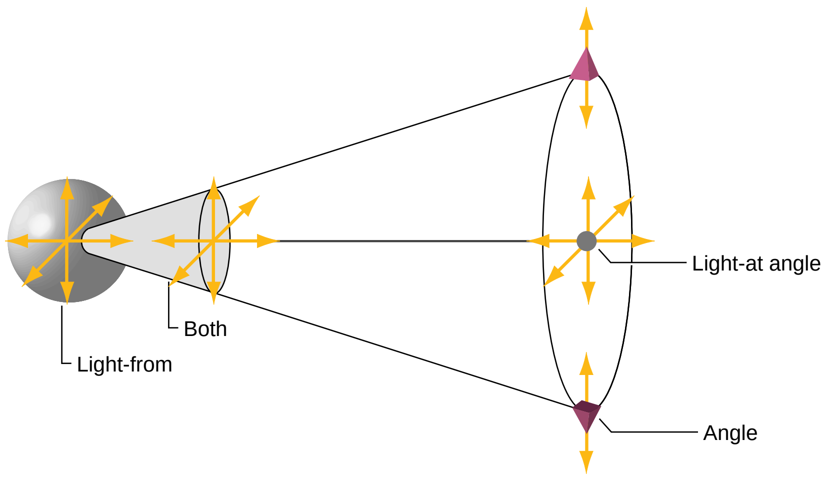


FIGURE 14 The point light selected point light widget has similar controls to the spot light.

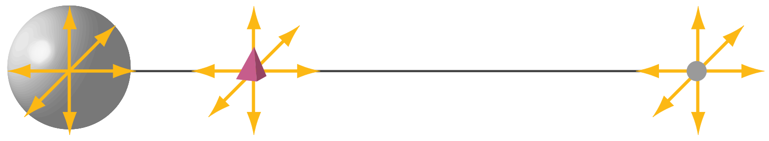


FIGURE 15 The selected directional light is shown with a widget.

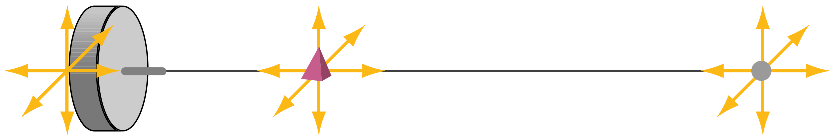
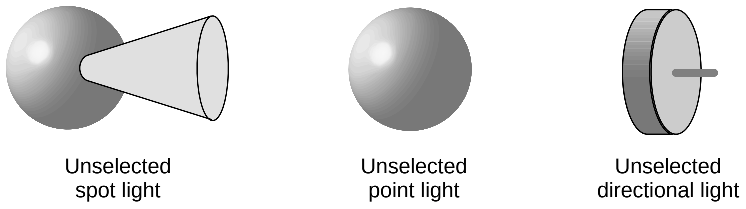


FIGURE 16 When unselected, the light widgets appear as just their light-from affordances.



All the affordances of the light widgets are highlighted with the asterisk sparkle.

Click-dragging on the light-from affordance should move it using free-space pointing mode, leaving the light-to point unchanged. The point-at-shape pointing mode is enabled, allowing the light to be aligned with a surface. Shift key keeps the light-from position on a sphere centered at the light-at position, with a radius of the pre-drag distance between the light-from and light-at positions. The sphere is divided in two by its horizon line. The light-from is kept on the hemisphere (not mathematically a hemisphere, because of perspective) that it was closer to (measured in world space) when the shift key was pressed.

Click-dragging on the light-to affordance is the same as the light-from, except that the constraint sphere is centered on the light-from.

Click-dragging on the "both" affordance moves both the light-from and light-to positions by the same amount, in free-space pointing mode. Point-at-shape is disabled. Shift key constrains to the plane through the original grab point, perpendicular to the light-from / light-to line.

Clicking on an unselected light selects it, and then behaves exactly like the light-to drag.

The usual shift-click behavior for multiple selection. When multiple lights are selected, they are moved together. When a mixture of lights and shapes are selected: moving the both affordance also moves the shapes; moving a shape moves the lights without rotating them.

The point light widget’s angle affordances work identically to one another. They are shown at the widest diameter (measured in the image plane) of the ellipse. Click-dragging on an angle affordance constrains to that diameter. There is no shift key behavior.

1.4.5.3 Light Creation

All lights should be created by a user interface that is similar to creating geometric objects; this typically, consists of a tool palette of geometries and lights.

• Default color:	white
• Default intensity:	80%
• Default shadows:	off
• Default falloff:	inverse linear
• Default fuzziness:	sharpest
• Default fuzz style:	linear

- Pointing defaults to point-in-space, with point-at-shape enabled.

A click-release establishes light-from position, and sticks the light-to to the pointer. Click-drag then establishes light-to position and adjusts it. The light is having an effect during this time. Alternately, starting with a click-drag establishes the light-from position at the click position, and the drag adjusts the light-to position.

For spot light, the angle is some default width during the light-from and light-to positioning. Then a constraint line appears through the widest ellipse diameter, and the angle affordance tracks on the line until click-drag-release completes.

1.4.5.4 Scalability

The interface described here assumes a very sophisticated user, who is attempting to express a visual concept using precise lighting controls. If this is not the user model for the application, then some expressiveness can be eliminated for the sake of simplicity.

Removing control over light attenuation simplifies the user interface. For point and spot lights, inverse-linear attenuation would be the only type available. Certain light types may be unnecessary for the purposes of the application. Lighting consisting of only inverse-linear point lights may be completely sufficient for the user’s task.

To further simplify lighting, the scale factor on attenuation could be computed from the overall bounds of the scene, eliminating the light-to position for point lights.

An extreme level of simplicity is to eliminate provisions for editable lights, giving instead preset groups of lights. The user would choose from a set of standard light configurations that looked best. Alternately, several lights in fixed positions, and an on/off switch for each could be provided.

1.4.6 Viewing & Cameras

As discussed above the user must be able to understand the 3D shapes and their configuration to achieve the desired effect. This understanding comes from the user's exploration of the scene, in concert with the application's presentation of the 3D space. A still picture is of limited use for both understanding and interacting with shapes and their arrangement. The user must be able to direct the interactive viewpoint to examine or interact with a desired part of the scene.

Scrollbars in a 2D application require the correct point of view to understand their operation. The user may desire to scroll the document up in the view, or equivalently to slide the view down on the document. The former is document-centered scrolling, and the latter is view-centered scrolling. Macintosh has solved this problem by making scrolling view-centered: Pressing the "down" button moves the view down, and the document up. Interestingly, some "paint" applications provide a "hand" mode that performs document-centered scrolling: click-dragging the hand down moves the view up, and the document down within the view.

A similar problem exists in 3D: the user may wish to move a shape to the right in the view, or equivalently to move the viewpoint to the left. Worse, the user may wish to rotate the shape to the right, or to rotate the view to the left about the shape. When a single object is visible, with no other context, the natural thing to do is to perform scene-centered movement, analogous to document-centered scrolling. This is also the case when examining a single shape in a scene. When navigating through a scene of shapes, it is more natural to think about view-centered movement. A user may move about a scene, and then find an object to examine. At some point, the focus changes from view-centered to scene-centered. It's very difficult to determine when that transition point occurs. Except under very special circumstances, camera motions should be controlled in the view-centered model.

Camera position is the 3D analog of scrolling and zooming in a 2D drawing or painting application. Like the 2D application, the user pans and scrolls to work on some portion of the document at the appropriate level of detail. Unlike the 2D application, in 3D the camera position is also used to control the final rendering. The suite of camera controls must support both the interactive viewing and rendering setup tasks.

1.4.6.1 Camera vs. Interactive Viewpoint

Traditional 3D tools make it difficult to move the camera viewpoint. They often combine the concepts of the rendering camera with the interactive viewpoint. The combination of these concepts does the user a disservice, since it restricts the user from changing the viewpoint to better interact with the scene.

Applications should instead separate the concepts of the rendering camera(s) and the interactive viewpoint. The interactive viewpoint is the camera that is associated with a dynamic, interactive view, in which the user manipulates the scene. A rendering camera is associated with a static snapshot of the scene, or with an animated movie. One way to provide this distinction is to allow the user to note good vantage points, as in MacDraw Pro's "Set View…" menu items. Another is to have two kinds of windows: interactive and preview. These are accessed in New Era using the buttons at the bottom-left of a view window.

Movement of the viewpoint may be controlled by the user in a few ways: dragging a special widget in the scene, using the camera tool, manipulating controls outside the scene, or using a special input device.

1.4.6.1.1 Camera Widget

The camera widget represents in the scene both cameras associated with views and viewpoints. They are used to select a camera so an operation can be performed on it, and to control the look-from and look-at positions of the camera. The operations include deleting the camera (and corresponding view or viewpoint), to open the corresponding view (or make one if no view exists already), and to move the active view's camera to the position of the selected camera.

The camera widget in New Era is a very simple affair. It consists of a look-from point, a look-at point, and a handle that moves both together. Its primary use is to position the camera at a position relative to some existing object in the scene, e. g. at the model of a head, looking at a specific feature.

The look-at and look-from points can be dragged in both free and pick pointing modes. The camera retains the state of both endpoints, and uses the prior state for subsequent drags. The handle that moves both endpoints is only used in free pointing mode.

When unselected, the camera widget appears as a frame in the plane that contains the look-at position and is parallel to the image plane. When selected, affordances appear at the look-at and look-from points, and a frame appears to indicate the viewing volume.

Click-dragging on the look-at or -from point moves it using point-in-space mode. Point-at-shape is enabled so the point may be placed on an existing surface. With the shift key down, the point is constrained to a fixed distance from the other point, with the usual hemisphere solution.

1.4.6.1.2 Rotate Camera Tool

The Rotate Camera tool is a variation on the Rotate Shape tool, butinstead of moving the shape, moves the camera. It shows the same six handles (joined with lines instead of circles, just to look different). Instead of translating the object in response to dragging on the shape, it moves the camera to keep the grabbed point under the cursor. Instead of rotating the object when a handle is dragged, it moves the camera to keep the handle under the cursor. When the tool is active, Point-at-Shape is enabled.

Note that the current implementation of the handle rotation is unpredictable when dragged outside of the horizon of the Rotate Camera tool widget.

Click-dragging on a handle moves the camera so that the center of the sphere doesn't move visually, the handle stays (as close as possible) to the cursor, and the camera doesn't twist off of vertical. The handle stays on the visible or far side of the sphere, depending on which side it started. The camera is kept from looking within a few degrees of straight down or up. With the shift key down, the handle travels on the closer of the two ellipses that passed through it when the drag started.

Click-dragging on the shape moves the camera so that the grabbed point stays under the cursor, using Point-in-Space mode. The camera stays untwisted with respect to vertical.

With the shift key up, the camera gaze direction stays parallel to its starting direction. With the shift key down, the camera look-from position is unchanged, using sphere-constrained pointing, centered on the look-from, with radius determined by the distance to the grabbed point.

Since it is useful to interleave this tool with other tool use, a modifier key chord will temporarily activate the Camera Rotate tool.

1.4.6.1.3 Close-Up

Whenever the cursor is in the active view, holding the command and option keys down zooms into the details under the cursor (not yet implemented in New Era). This is true if the mouse button is up, or even while dragging a handle. The zoom centers on the cursor, and as the cursor moves, the zoom center moves accordingly. That is to say, close-up doesn't change what is being pointed at, but merely the display of it. The zoom returns to normal when the keys are released.

A suitable zoom factor, e. g. 4x, should be chosen. If camera transitions are "on" then the zoom should ease in and ease out.

1.4.6.1.4 Animated Transitions

Camera moves may be animated to make them more understandable to the user. An application parameter (accessed under Preferences…) controls whether the transitions are immediate, or animated quickly (a third of a second) or slowly (a second). Complex camera transitions, such as happens when using the Rotate Camera tool with a rotate handle, the shift key down, and jump between the two constraint circles, should be animated through "legal" positions, rather than just interpolated.

1.4.6.1.5 Camera Controls Palette

The camera controls palette provides non-spatial controls for the active view. These controls include the type of camera (perspective, orthographic, viewplane), image-space controls for field-of-view, and sliders for twist, brightness and contrast.

Controlling field-of-view for a perspective camera while looking through it can be rather confusing—it is difficult to distinguish field-of-view changes from front-back movement. Orthographic cameras have a scale factor, similar to field-of-view, which must also be controlled. Viewplane cameras are difficult to understand only by looking through the camera. These controls are presented as image-space controls, which select a portion of a larger image space. This is portrayed as an image with a box that frames the actual view. The box can be scaled and moved about on the image. The image can be zoomed in or out.

The box is always completely contained by the visible part of the image. It can't be moved or scaled out of view. Zooming in is limited so that the box is visible. Zooming in is limited so that the box doesn't become too small.

Scaling the box controls the field-of-view of the perspective camera, the scale of the orthographic view, and the scale of the viewplane camera. The box is uniformly scaled, since proportions are controlled by an image settings dialog and the window zoom. Scaling is about the center of the box to keep from inadvertently moving the look-at position.

Moving the box changes the camera so that it best matches the box. For a perspective camera the look-at position is changed, while keeping the same distance from the look-from position, so that it corresponds to the world-space ray through the new box center in the image. For an orthographic camera, both the look-at and look-from positions are moved parallel to the image plane so that the world-space ray is matched. After the box is moved for either perspective or orthographic cameras, the image is re-rendered so the box is again at the center. Moving the box on a viewplane camera shifts the camera's image offset. For the viewplane camera, the box may be placed off-center.

To the extent possible, moving and scaling the box should be "live," updating the underlying view during the drag.

Zooming the image goes by some reasonable steps, e. g. by scales of 3/4 and 4/3. Zooming a perspective camera changes the image field-of-view, up to some reasonable maximum (e. g. 120°).

The twist control changes the degree of twist off of vertical (with respect to the room) for the current view, ranging over ±180° with a detent at zero. The camera's "up vector" is computed to be the orthogonal projection of the room up vector onto the image plane, then is twisted in the image plane by the twist value.

The brightness and contrast settings provide overall lighting adjustment for the scene when rendered in the active view. The ambient light brightness is multiplied by the brightness setting; all other light brightness values are multiplied by the contrast setting. The brightness should range [-1..1] and the contrast should range [0..2]. Both should have a detent at their center.

1.4.6.1.6 Camera Nudge Palette

The camera nudge palette provides incremental, orthogonal control of the camera parameters. A pop-up menu provides choice of which parameter (or pair of parameters) is controlled; two (or four) buttons increment the chosen parameter(s). The parameters are:

- pan-tilt (look-at)
- orbit (look-from)
- dolly
- truck
- zoom (for orthographic or viewplane, this is the image range/scale)
- vertigo zoom (perspective only)
- twist
- offset (viewplane only)

It may be useful to provide a thumbnail image per nudge button to show the effect of many nudges of that button.

1.4.6.1.7 Undo/Redo

A special undo queue should be maintained for the interactive viewpoint, separate from the genereal undo queue. This provides a light-weight way to look at a configuration of shapes from a variety of positions, and to check how changes look from those positions.

1.4.6.1.8 Viewpoints

Views correspond to windows, and are useful for saving very important viewing positions and snapshot image setups. It is useful to provide lighter-weight viewing positions, which we call *viewpoints*, which save only the most basic viewing information, e.g. look-from, look-at and field-of-view.

A viewpoint may be made by issuing a menu command, which “remembers” the current camera position. A camera widget is left in the scene indicating the viewpoint. The viewpoint may later be activated, moving the view to the remembered look-from, look-at and field-of-view.

1.4.6.1.9 Standard Views

It is often useful to provide some standard views onto a scene, e.g. front elevation, side elevation and plan views. This may be accomplished by having a scene always contain views with those positions, and disabling any camera motion (and name changes, deletes, etc.) for those views.

1.4.6.1.10Hither and Yon

The user should never be aware of hither and yon clipping. The hither and yon clipping planes should be set automatically. The yon distance should be chosen to encompass the farthest visible object. The hither distance should be a tiny fraction of the yon distance, e. g. yon/1,000,000.