A User Interface for Interactive Cinematic Shadow Design

Fabio Pellacini Parag Tole Donald P. Greenberg
Program of Computer Graphics, Cornell University

ABSTRACT

Placing shadows is difficult task since shadows depend on the relative positions of lights and objects in an unintuitive manner. To simplify the task of the modeler, we present a user interface for designing shadows in 3d environments. In our interface, shadows are treated as first-class modeling primitives just like objects and lights. To transform a shadow, the user can simply move, rescale or rotate the shadow as if it was a 2d object on the scene's surfaces.

When the user transforms a shadow, the system moves lights or objects in the scene as required and updates the shadows in realtime during mouse movement. To facilitate interaction, the user can also specify constraints that the shadows must obey, such as never casting a shadow on the face of a character. These constraints are then verified in real-time, limiting mouse movement when necessary. We also integrate in our interface fake shadows typically used in computer animation. This allows the user to draw shadowed and non-shadowed regions directly on surfaces in the scene.

CR Categories: I.3.7 [**Computer Graphics**]: Three-Dimensional Computer Graphics and Realism; I.3.6 [**Computer Graphics**]: Methodologies and Techniques – Interactive Techniques.

Keywords: Human Computer Interaction, Illumination, User Interface Design, Lighting Design

1. INTRODUCTION

The patterns of light and shadows on the surfaces of objects are extremely important visual cues. In cinematic lighting design for movies, light designers carefully place lights and objects to specify the visual appearance of the scene and enhance storytelling [Calahan 2000]. Although so important, the task of designing shadows is extremely complex even for experienced users since shadows are affected by lights and objects positions in a very unintuitive manner.

To simplify the task of light designers, we propose a new user interface that lets the user manipulate the shadows in the scene in real-time, while the system automatically adjusts lights and objects transformations as necessary. In our interface, shadows are treated as first-class modeling primitives just like objects and light sources. A shadow can be moved, rotated or scaled as if it was a 2d objects on the scene's surfaces, and similar to other objects in the scene. The mouse interface used is same as that used for object transformations in familiar graphics packages such as Maya™ or 3dsMax™. For example, in order to place a shadow, the user simply clicks on the shadow, and then drags it to the desired location.

In order to further simplify the task of placing shadows in a complex environment, we let the user place constraints on the shadows using an intuitive painting interface; for instance, the modeler may want to ensure that there is never a shadow on the main character's face. These constraints are verified in realtime and enforced by limiting mouse movement when necessary. This allows the user to manipulate the shadows in complex environments more efficiently.

In addition to real shadows caused by occlusion of light sources, fake shadows are used in cinematic lighting to enhance the visual appearance of the scene. We integrate fake shadows in our interface providing the user with the ability to draw shadowed and non-shadowed areas directly on the surfaces of the objects. The system automatically creates 3d cucaloris [Barzel 1997] that are updated appropriately when the lights are moved.

We have implemented a prototype of our interface using shadow maps on commodity graphics hardware. Our prototype supports multiple lights, soft shadows and constraint validation in real-time. Since the only requirement to implement our interface is the support of hardware accelerated shadow maps, we can easily integrate our interface in relighting engines such as the one presented by Gershbein and Hanrahan [2000] to provide a complete light design solution for cinematic lighting.

We believe that our shadow manipulation interface, combined with shadow constraints and fake shadows, provides the user with a powerful tool for shadow design for cinematic lighting. Our user interface for shadow manipulation is very intuitive since it takes advantage of users' familiarity with standard interfaces for object transformations by treating the shadows as 2d primitives on the surfaces of the scene.

2. RELATED WORK

Researchers in computer animation have tried to simplify lighting design interfaces. Poulin and Fournier [1992] presented an interface where the user interacts with the shadows by transforming the shadow volumes in a wireframe view. Although interesting, we believe that this interface is far less intuitive than manipulating the shadows themselves; also the shadow volume wireframe clutters the scene display, making it hard for the user to appreciate the scene's final look. In a following work, Poulin et al. [1997] presented a sketching interface to specify constraints on the shadows, which are then solved to compute the light positions. Unfortunately, it is almost impossible for the user to specify solvable constraints for complex scenes, thus limiting the usability of their system. Our constraints are specified in the same way, but instead of being solved, they just need to be verified so as to limit mouse movement when necessary. In the context of global illumination, Schoneman et al. [1993] presented an optimization technique based on a sketching metaphor, while Kawai et al. [1993] automatically set the light parameters based on a subjective impression of the scene illumination. Although useful, these systems cannot move the light sources. Also, none of these previous approaches allow the user to adjust object positions, which we do with the same interface as lights.

3. USER INTERFACE

This section describes the user interface from the point of view of the modeler, leaving implementation details for the next section. In the rest of the paper, we will discuss only spotlights. Since omni lights and directional lights have fewer degrees of freedoms, they can be integrated using a subset of the user interface commands used for the spotlight case.

3.1 User interface paradigm

Our physical user interface is based upon a traditional 2D input device such as a standard three-button mouse or a tablet. The interaction mode of the user with the scene is selected using modifier keys. Although important, the particular interface paradigm is not the main contribution of this paper; our algorithm can be integrated in other interface paradigms such as gestural systems, button widgets or gizmo widgets [Conner et al. 1992].

3.2 Lighting operations

3.2.1 Overview

When direct lighting illuminates a scene, hotspots and shadows define the patterns of the light on the surfaces of the objects. To completely specify the light patterns, the user has to be able to modify the position and size of the hotspot of each spotlight and the position, size and orientation of each shadow. While the hotspots depend only on the lights parameters, the shadows depend on the positions of lights as well as objects in the scene. When interacting with the shadows, the user chooses to either modify the light source or the object transformation by selecting the specific mode. Figure 1a shows the lighting operations provided in our interface together with the equations the system uses to update corresponding light and object transformations.

3.2.2 Shadow operations

At any time during the light design process, our system presents the user with a rendering of the scene from an arbitrarily selectable camera position. By clicking on a shadow, the user selects the light-object pair that casts that shadow. Depending on the interaction mode, the user can move, scale or rotate the shadow as if it was a 2d object on the surfaces of the scene. The system figures out the correct transformation for either the light or the object. In case more than one object is casting the selected shadow, the system automatically chooses the one closest to the light; while in the presence of more than one light, the system selects the closest light as for the hotspot operations' case. While this automatic selection seems to work well, the users have the option of overriding the automatic behavior and selecting the light-object pair.

When changing the light parameters, the user can either move or rescale shadows. The user moves a shadow by dragging it on the surfaces of the objects in the scene. During mouse movement, the system moves the light source by rotating the ray connecting the picked point and the light around the closest point of the shadow-casting object to the light. To rescale a shadow, the user drags the mouse vertically while the system moves the light position along the line connecting the light position and the object center proportional to the amount of mouse movement.

To simplify the light design task, it is important that the operations on shadows affect the appearance of the hotspots as little as possible. To ensure this, when the light position is being changed, we also update the direction of the light so that the position of the hotspot on the picked surface remains the same. We also would like to keep the size of the hotspot (as projected on the surfaces) the same. Unfortunately, this is not always possible on general surfaces; nonetheless we try to correct most of the size mismatch by resizing the hotspot angle as if the picked surface was a plane.

The user can also change the position, size and rotation of the shadow by changing the position and rotation of the object casting the shadow. The interface to move and scale the shadow is the same as above. When moving the shadow, the system moves the object by rotating it around the light position. While resizing the shadow, the system moves the object along the line connecting the center of the object to the light position. To rotate the shadow, the user drags the mouse vertically, similarly to object rotation in commercial animation packages. The object is rotated around the line connecting the center of the object and the light position. The shadow operations on an object specify only four degrees of freedom. The remaining two are object rotations along two axes orthogonal to the line joining the light and the object. For a general object, these rotations affect the shape of the shadow in a completely unintuitive manner. We believe that adding these non-intuitive shape transformations as a shadow manipulation command would not help the users more than simply using standard object rotations (also provided in our prototype).

3.2.3 Additional shadow operations

In the rescaling operation specified above, the position of the mouse on the screen does not represent the size of the shadow being rescaled (see Figure 1a). When finely tweaking the shadow, it would be helpful if simply dragging the shadow silhouette to a new position could rescale the shadow by that amount. In general this is not possible for an arbitrary silhouette and an arbitrary mouse movement. Nonetheless for small mouse movements, we can approximately achieve it. Depending on the selected action, the system moves the light or the object along the line passing though the light position and the object center. It can be shown, that the interface will behave as expected provided that the picked point during mouse drag is roughly in the plane containing the light position, the object center and the silhouette point picked by the initial mouse click. While the system tries to find the best solution for every possible mouse position, the shadow may not be in the expected position for large mouse movements.

Our interface also supports soft shadows from area lights. The user can choose the shadow softness with the same mouse interface as rescaling the shadow. The system will rescale the size of the area light by the appropriate amount.

3.2.4 Hotspot operations

The user can change the size and the position of the hotspot of the spotlights in the scene, by clicking on a hotspot and moving or rescaling it with the same mouse interface used for shadows. In the presence of multiple lights, the system automatically selects the light whose hotspot direction is closest to the line connecting the light position and the picked point. The user can choose to override the automatic behavior. When moving the hotspot, the system will compute an appropriate spotlight direction and while rescaling it, the hotspot angle will be modified.

The manipulation of the hotspot is very similar to the one used by commercial animation packages, except that the hotspot is moved on the surface of objects, making it very simple for the user to "shine a light on a specific location on a surface".

Shadow and hotspot operations completely specify the light source's parameters. A spotlight has six degrees of freedom: 3 for position, 2 for direction and 1 for hotspot angle. The light and hotspot operations in our interface map these onto shadow position and size (3) and hotspot position and size (3).

Since shadows and hotspots operations have the same interface as object transformations in commercial packages, the learning curve for new users is very short. In addition, since the hotspot operations do not affect the shadows, the design is simplified by making the shadow and hotspot interactions orthogonal.

3.3 Constraints

When light designers are interacting with shadows in complex scenes, they often want to specify constraints that shadows have to obey during interaction. In our interface, the user can specify that an area of a surface should not change its "shadow state" by drawing the contour of that region on the objects. Drawing a constraint on a shadow has the effect of enforcing that the marked area will always remain in shadow by at least one object; while drawing the constraint on a lit area will make sure that no shadows will be in that area. While manipulating the shadows in the scene, the constraints are verified and the positions of objects and lights are updated only if all of the constraints are valid. Figure 1b illustrates the interface for constraints.

We believe that our interface to set constraints by painting is an intuitive way of specifying complex constraint behavior. Also, since we define a constraint as a set of points that cannot change their "shadow state", we implicitly let the user specify only solvable constraints, thus avoiding the confusion of other interfaces where unsolvable constraints can be specified.

3.4 Shadow Cookies

Shadow cookie [Barzel 1997] is the colloquial term for cucaloris, the technical term for an opaque card with cutouts used to block a light. In special effects production environments, cookies are used to add fake shadows from non-existent objects.

Our interface allows the user to paint shadow cookies with the same painting metaphor used for constraints, shown in Figure 1c. The user can "attach" the cookie to the light source or to the world selecting the appropriate interaction mode. When the cookie is attached to the light, moving the light will not move the position of the cookie shadow on the surfaces of the objects. On the other hand, if the cookie is attached to the world, then moving the light will move the shadow. Also world-attached shadow cookies are first-class shadow objects and their shadows can be individually manipulated using the shadow-object mode.

We also introduce light cookies that are used to subtract any existing shadows from their region (Figure 1c). They are drawn in the same way as shadow cookies and can also be attached to the light or to the world. World-attached light cookies can also be used to cut parts of the shadows cast by specific objects.

4. IMPLEMENTATION DETAILS

Our lighting interface requires realtime updates of the rendered scene with shadows. We achieve this by using hardware shadow maps for computing shadows and hardware lighting for the light contributions. Our prototype runs at about 20 frames per second on a Pentium III 500 MHz with a GeForce 3 graphics board for scenes with 4 light sources using a multipass tecnique. When computing shadows from area lights, the number of samples is reduced to 4 during mouse interaction, and the final image is computed when the interaction stops.

During interaction, the system determines the 3d position of the point picked by the mouse by querying the depth buffer of the camera view. Also the system needs to determine which light-object pair is casting a shadow on the picked point. This is computed by querying the depth and id buffers of the given light's view. Since the two read back operations are required only at mouse click, there is no slowdown when dragging the mouse.

Constraints are represented as the array of the 3d positions of the points on the contour of the constraint. This data structure makes it simple to specify constraints on surfaces of any shape. We verify a constraint by checking if any of the points in the constraint has changed its shadow state. To do this, we query the depth buffer of each light with the position of each point in the constraint.

We represent cookies as 3d polygons created halfway between the light position and the picked point. Shadow cookies are drawn only in the shadow map pass, while light cookies are drawn in the stencil buffer, which is then used to set the stencil test to fail in these regions. Representing the cookies as 3d objects makes it very easy to use the rest of our interface to manipulate the cookies themselves. It would be much harder to modify the cookie later if we were to directly paint a matte in the shadow as is normally done in computer animation [Barzel 1997].

5. CONCLUSIONS AND FUTURE WORK

We presented a user interface for interactive cinematic shadow design consisting of an intuitive interface for shadow manipulation and a painting interface for shadow constraints and fake shadows. Our interface can be easily integrated in relighting engines such as [Gershbein and Hanraham 2000] to provide a more complete interactive lighting design solution. We believe that our interface to manipulate shadows is intuitive to use; treating shadows as 2d objects on the surfaces of the scene allows us to leverage the user's familiarity with object transformations. We also believe that the ability to draw intuitive constraints simplifies the design process for accurate shadows in complex lighting situations.

We would like to extend our interface in several directions. First, we would like to try to integrate our shadow manipulation interface in other paradigms such as widgets [Conner 1992] and sketching that have proven to be very fruitful in surface modeling. We would also like to extend our shadow interface to support more complex light parameters such as the one presented in [Barzel 1997] to provide a better user interface for a more complete light design solution. For the fake shadow components, we would like to investigate if the behavior of more complex cookie representations such as [Petrovic et al. 2000] can also be made more intuitive to use for the user.

We would also like to test how our system scales with the complexity of the scene. First, updating the shadows in realtime may not be possible in some cases; integrating the use of simplified geometry for shadow computation might make the system more usable. Furthermore, an extremely complex environment could have very complicated overlapping shadow patterns that could make the selection of light-object pair cumbersome. While our informal tests suggest that our interface scales well with up to four lights and hundreds of objects, more formal user studies are needed to verify the scalability when the complexity is dramatically increased. Also, in the presence of very complex shadow patterns, it might be useful to specify constraints relatively to specific light-object pairs. While our constraint validation system currently supports these constraints, we would like to extend our constraint painting metaphor to support these more complex constraint specifications.

Finally, we would like to extend our interface to support animated scenes. Currently the user can design the shadows for animated sequences by lighting a few selected frames separately and using keyframe interpolation for the remaining frames; fruitful extensions could be the use of shadow movements to define the animated parameters of the lights directly and the specification and validation of constraints for the entire animation sequence.

ACKNOWLEDGMENTS

We would like to thank Randy Fernando, James Ferwerda, Moreno Piccolotto and Bruce Walter for their comments. This work was supported by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and performed using equipment generously donated by Intel Corporation and NVidia Corporation.

BIBLIOGRAPHY

BARZEL, R. 1997. Lighting controls for computer cinematography. *Journal of Graphics Tools*, 1997.

CALAHAN, S., 2000. Storytelling through Lighting, a Computer Graphics Perspective. In *Advanced RenderMan*, 2000.

CONNER, D.B., SNIBBE, S.S., HERNDON, K.P., ROBBINS, D.C., ZELEZNIK, R.C., AND VAN DAMN, A., 1992. Three-Dimensional Widgets. In *Proceedings of SIGGRAPH 1992*.

GERSHBEIN, R. AND HANRAHAN P., 2000. A Fast Relighting Engine for Interative Cinematic Lighting Design. In *Proc. of SIGGRAPH 2000*.

KAWAI, J.K., PAINTER, J.S. AND COHEN, M.F., 1993. Radioptimiation – Goal Based Rendering. In *Proceedings of SIGGRAPH 1993*.

PETROVIC, L., FUJITO, B., WILLIAMS, L. AND FINKELSTEIN, A., 2000. Shadows for Cel Animation. In *Proceedings of SIGGRAPH 2000*.

POULIN, P. AND FOURNIER, A., 1992. Lights from Highlights and Shadows. In *Proc. of Symposium of 3D Interactive Graphics, 1992*.

POULIN, P., RATIB, K. AND JACQUES, M., 1997. Sketching Shadows and Highlights to Position Lights. In *Proceedings of Computer Graphics Internation 97, 1997*.

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J. AND GREENBERG, D.P., 1993. Painting with Light. In *Proceedings of SIGGRAPH 1993*.

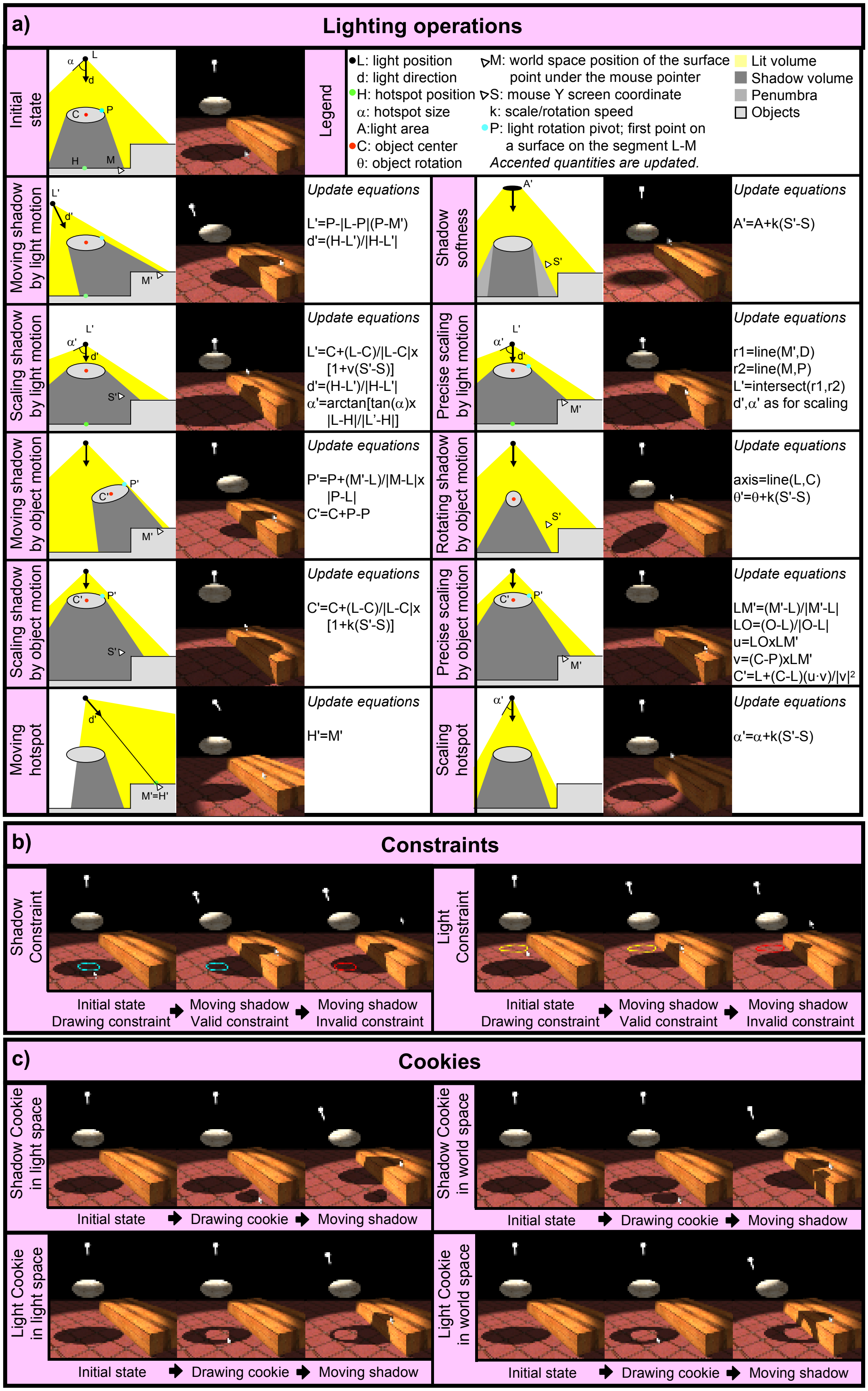

Figure 1: a) Lighting interaction modes. b) Constraints. c) Cookies.