A User Interface for Interactive Cinematic Shadow Design

Fabio Pellacini
Parag Tole
Donald P. Greenberg

Program of Computer Graphics
Cornell University
Importance of Shadows
Shadow placement

• Shadow placement by directly transforming lights/objects is hard
  – Shadows depend
    • on lights positions
    • on objects positions
    • in a very unintuitive manner
  – Need to determine which light/object pair cast the shadow
Previous work

- Interactive systems
  - Shadow volume manipulation [Poul92]
    - Not as intuitive as manipulating shadows
- Optimization-based systems
  - What you paint is only approximately what you get
Our approach

- Shadows are treated as first class entities
- Shadows transformations
  - displayed in realtime
    - quick user feedback
    - performed by a “click-and-drag” interface
      - mouse click: select shadow
      - mouse drag: move/scale/rotate shadow
        - on the surfaces of the scene
- All shadows are real!
Our approach - VIDEO
Shadow movement example

- Click on a shadow to select it
  - Light-object pair is selected
Shadow movement example

• Drag the shadow to a new position
  – Constrained on the surfaces of the scenes
Shadow movement example

- System rotates the light around the object
Light vs. Object transform

Move Light

Move Object
Shadow scaling example

- System moves the light on the axis passing through the object center and the light
Light vs. Object transform

Move Light

Move Object
Shadow rotation example

- System rotates the object around the axis passing through the object’s center and the light
Handling multiple lights/objects

• Automatically select light/object pair
Handling multiple lights/objects

- Automatically select light/object pair
Constraints

- Complex environments remain hard
  - Transforming a shadow affects other shadows
- Solution: apply constraints to mouse motion
  - Intuitive specification of constraints
Constraints

• Painting metaphor for constraint specification
Constraints

- Shadows are updated when constraints are valid
Constraints

- User is informed when constraints are invalid
Constraints

- Constraints for shadow regions
Shadow/Light Cookies

- Invisible “objects” used in cinematic lighting to add/remove shadows
  - Painting interface
  - First class objects
Shadow Cookies
Shadow Cookies

- Same painting interface used for constraints
Light Cookies
Light Cookies

- Same painting interface used for constraints
Light Cookies

Attach to the light

Attach to the world
Implementation details

• Requirements
  – Interactive update of shadows
  – Interactive validation of constraints

• Rendering
  – Hardware-assisted shadow maps
  – Multi-pass algorithm for multiple lights

• Constraints validation
  – Constraints represented as array of 3d points
  – Read back data from hardware for validation
Conclusion - VIDEO
Conclusion

- Shadows as first class entities
  - Interactive feedback to the user
  - Shadow transformations same as object ones
- Intuitive constraints specification
  - Interactive constraint validation
  - Limit mouse interaction when necessary
- Shadow cookies as first class objects
Future work

• Different input devices and UI metaphors
• Test scalability for complex environments
• More complex constraints
  – Already supported by the validation system
• Extensions to animated sequences
  – Supports only keyframing now
Acknowledgements

• Thanks to Randy Fernando, James Ferwerda, Moreno Piccolotto and Bruce Walter for useful 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.

• Modeling performed in 3DSMax. Models based on models from 3DSMax and 3DCafe.
References


Shadow precise scaling example

- Drag edge of the shadow to precisely rescale
  - Only for small mouse movements
**Light vs. Object transform**

Move Light

Move Object