Shadow Mapping

Casting curved shadows on curved surfaces

Lance Williams, SIGGRAPH '78

CSE 291, January 15, 2015

Presented by Ailie Fraser

Some slides and photos adapted from Mark Kilgard

Why are shadows important?

Overview

- About the author
- Background
- Shadow Mapping
- Dealing with imprecision and limitations
- Later improvements and extensions

Lance Williams

- PhD, University of Utah, Graphics & Animation
    - worked with Ivan Sutherland and David Evans
- New York Institute of Technology, ‘76-’86
    - worked with Ed Catmull, Jim Clark, Alvy Ray Smith
    - developed mip-mapping & shadow mapping techniques here
- Apple Advanced Technology Group, ‘87-’95
- Dreamworks, Disney, Google, Nokia, and now NVIDIA

MIP mapping

- A.K.A. image pyramids
- Reduced-resolution versions of an image
- Increases rendering speed and reduces aliasing!

=> Important for real-time rendering

Shadow Mapping - Background

Previous algorithms for adding shadows to scenes only worked for planar surfaces and polygons

- Can compute shadows for flat surfaces by projecting the object onto the surface
- Not so straightforward for curved surfaces

Shadow Mapping - Background

Williams' technique is based on "z-buffer visible surface computation" (Ed Catmull's PhD thesis, 1974)

- Technique for determining which parts of objects are visible in a scene
- Builds a z-buffer (depth map) containing the z-value of the closest object at each pixel
- Still widely used today

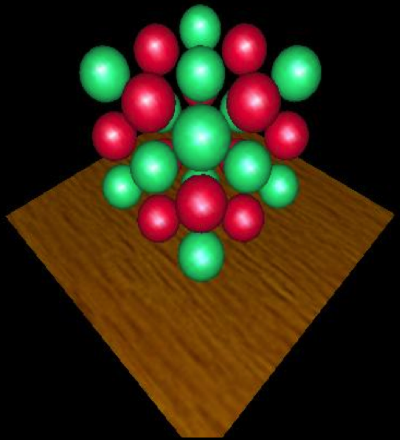
Shadow Mapping - Background

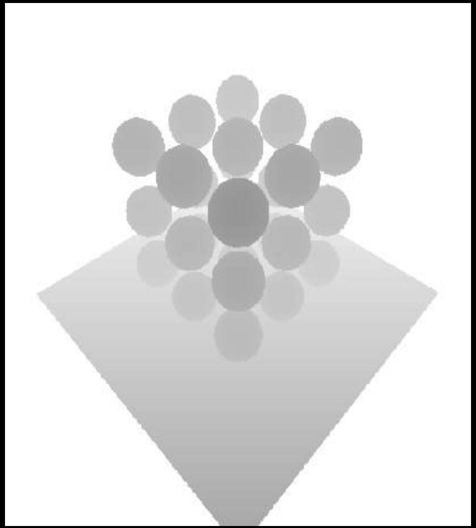
Advantages:

- Objects do not need to be sorted => works for very complex scenes
- Cost depends only on the screen resolution, not depth complexity
- All processing then happens in image space

Shadow Mapping

1. Construct a view of the scene as seen from the light source. Then calculate the Z-values at every point and store this depth map.





Shadow Mapping

2. Render the scene from the observer's point of view. For each point:

- Transform the point into the light source view
- Compare Z-value of transformed point (Z_t) with value in the depth map (Z_d) at the corresponding (X, Y) point
- $Z_t > Z_d$ => point not visible by the light
- $Z_t \approx Z_d$ => point is visible by the light

Shadow Mapping

"Correct" method:

- Transform each point into light space *as it is generated* when rendering the observer's view

Problem?

- This method's efficiency does depend on the scene's complexity. It transforms *all* points, even those not visible in the final image.

Shadow Mapping

“Modified” method:

Render the whole scene first, then do the transformation and shadowing as a *post-process*

This method does not depend on scene complexity, as only points visible to the observer are computed. BUT:

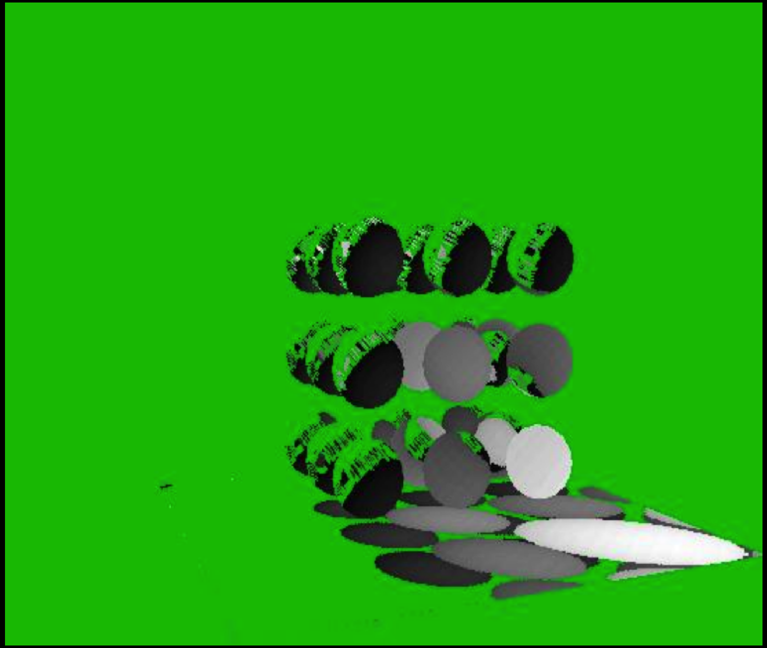
- highlights get shaded incorrectly
- more room for quantization problems

Shadow Mapping

Green = points where transformed Z-value and depth map value are approximately equal

Non-green = transformed

Z-value is greater than depth map value => should be in shadow



Dealing with imprecision

Problem 1: self shadowing

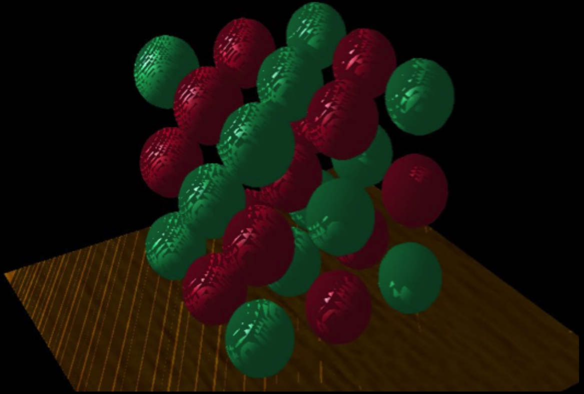
- Imprecise quantization means that the transformed Z-value will likely not be *exactly equal* to its corresponding depth map value
- If it ends up slightly larger, the point will incorrectly appear in shadow

Dealing with imprecision

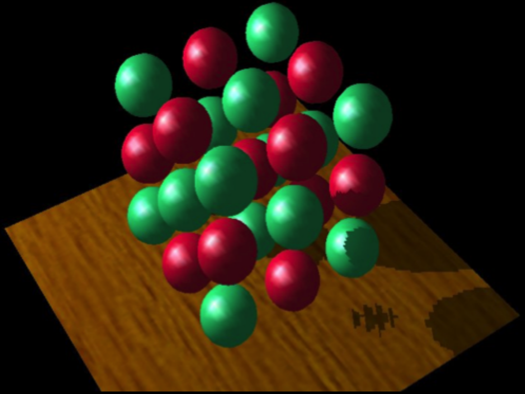
Solution: subtract a "bias" from the transformed Z-value

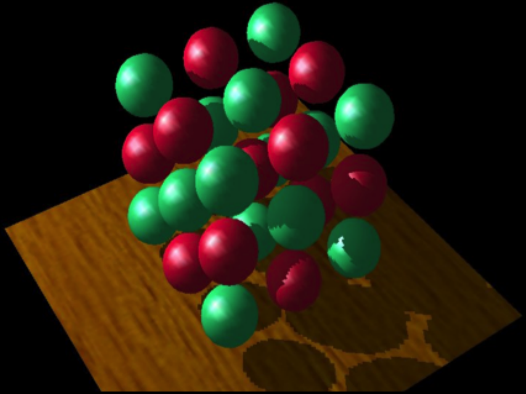
- Need to choose the right value for bias

too small



too big

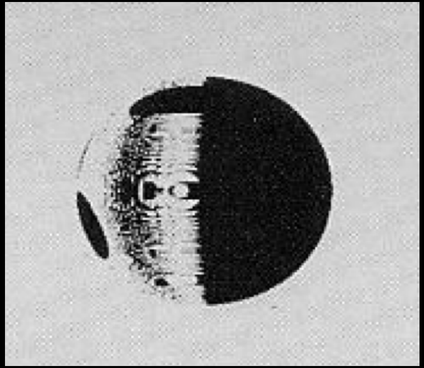




Dealing with imprecision

Problem 2: curved surfaces shadowing themselves

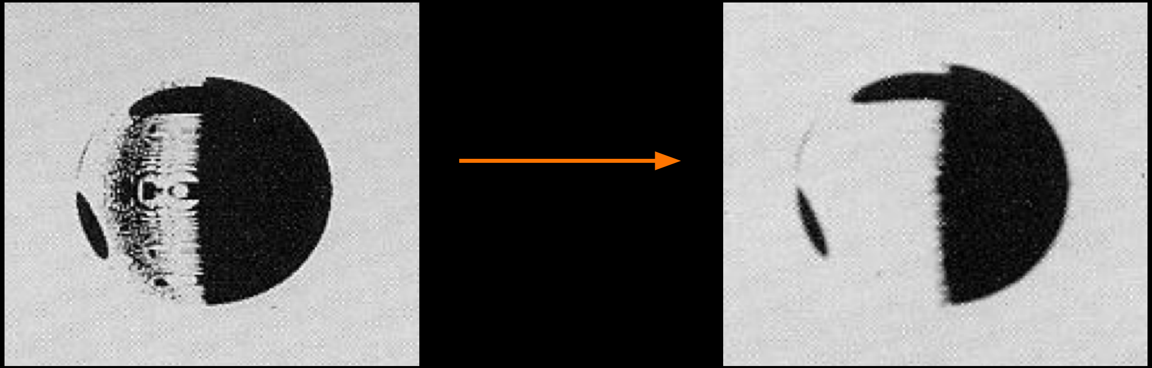
- Objects such as spheres should shadow themselves
- Point where shadow begins is not sharply defined
- May switch back and forth between shaded and not shaded



Dealing with imprecision

Solution:

- Add random dithering to transformed Z-values
- Apply an “edge dequantizing filter”
- Apply a low-pass filter to smooth contours



Limitations of shadow mapping

- Only objects within the viewing volume of the light source can cast shadows
- Multiple light sources (or omnidirectional light sources) require multiple depth maps and transformations => method is no longer efficient
- Resolution of the depth map determines quality of shadows (too low can cause aliasing and jagged edges)

Later Improvements

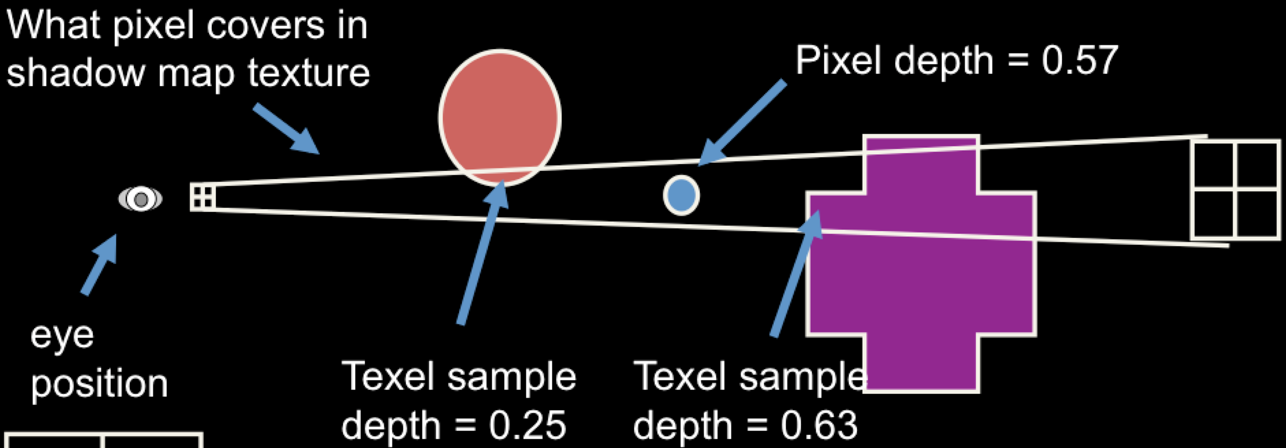
Percentage Closer Filtering

- Reduces aliasing and allows for soft shadow effects
- Apply averaging filter to comparison *results*, not depth values themselves

(Reeves, Salesin, Cook, 1987)

Percentage Closer Filtering

- **Traditional filtering is inappropriate**

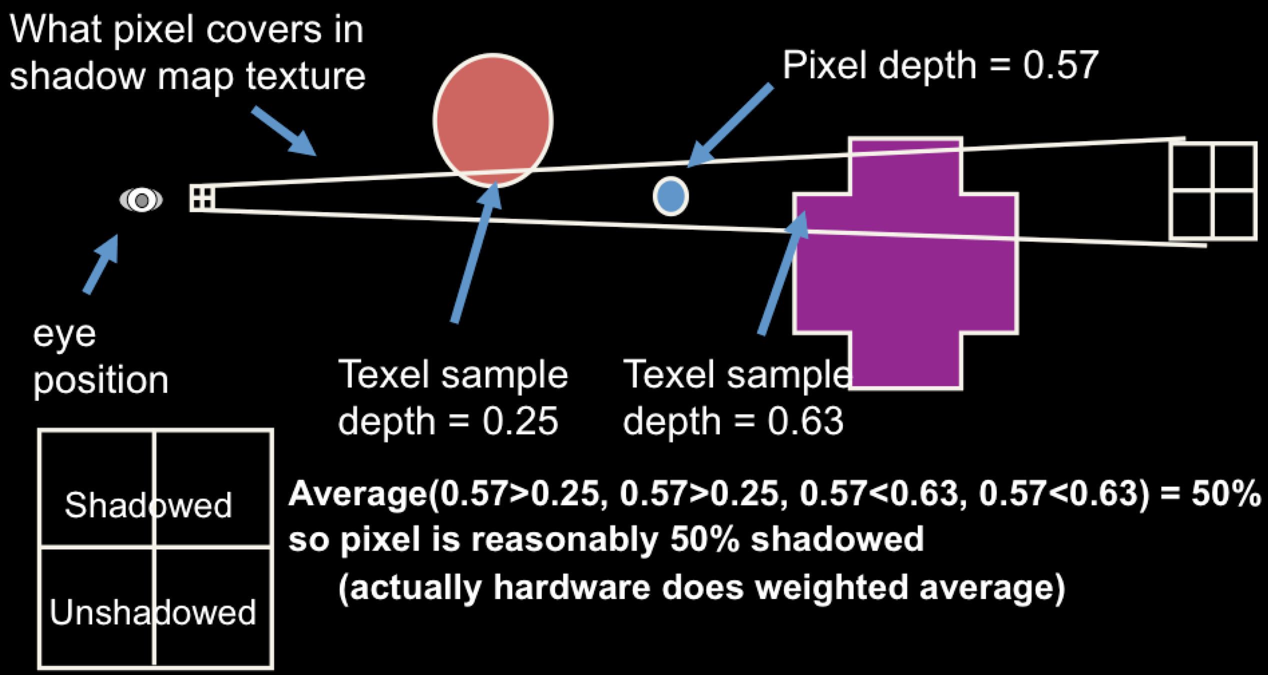


0.25	0.25
0.63	0.63

Average(0.25, 0.25, 0.63, 0.63) = 0.44
0.57 > 0.44 so pixel is **wrongly** "in shadow"
Truth: nothing is at 0.44, just 0.25 and 0.57

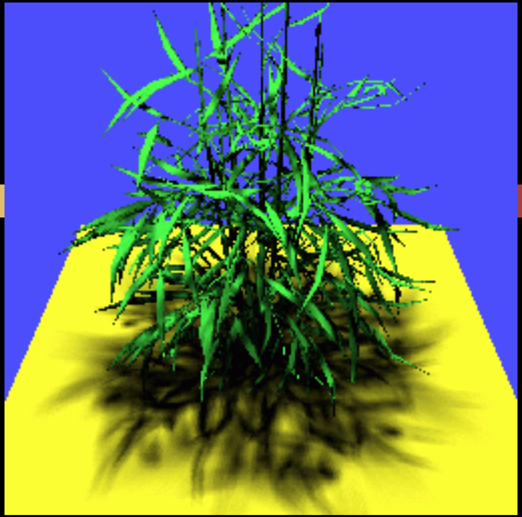
Percentage Closer Filtering

- **Average comparison *results*, not depth values**



Average(0.57>0.25, 0.57>0.25, 0.57<0.63, 0.57<0.63) = 50%
so pixel is reasonably 50% shadowed
(actually hardware does weighted average)

Layered Attenuation Maps



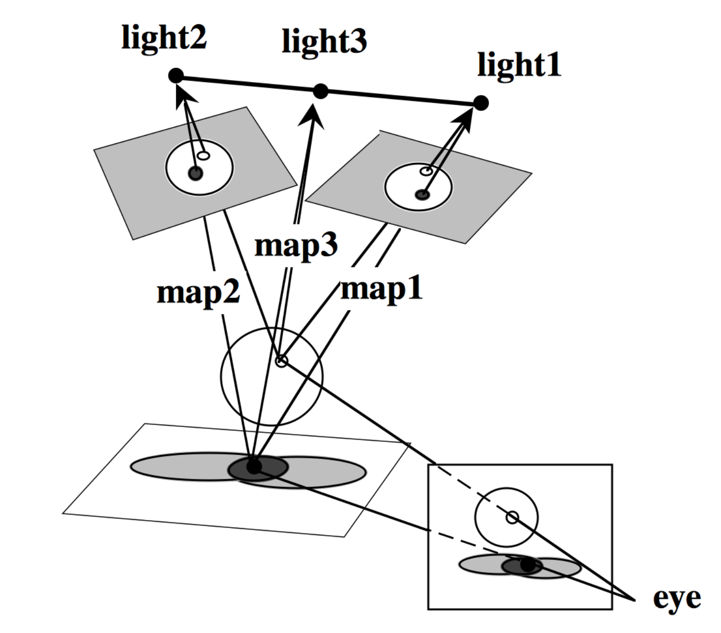
- Extends shadow mapping to produce soft shadows from area light sources **in real-time**
- Build views from several sample points on the light
- Warp these views into a central reference frame
- Build a *layered-depth image*:
  - Stores one layer for each depth value, along with what fraction of the views this pixel is visible in

(Agrawala, Ramamoorthi, Heirich, Moll, 2000)

View interpolation

Another method for soft shadows from area light sources

- Calculate depth maps for a small number of views from points on the light source
- Interpolate between them using morphing



(Chen, Williams, 1993)

Combine with shadow volumes

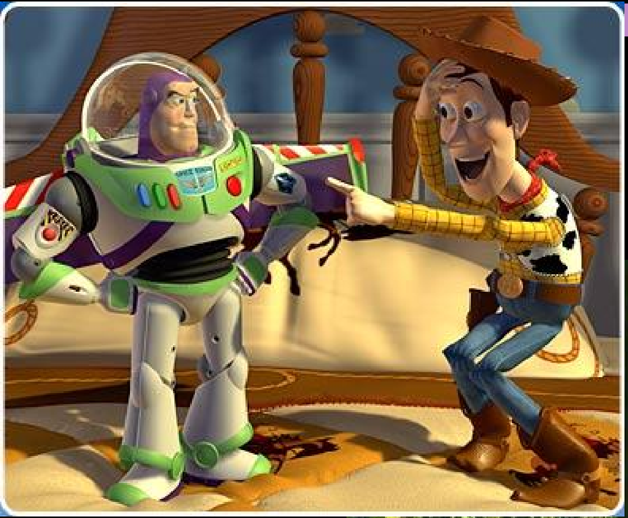
- Shadow volumes: compute the 3D areas in the scene that are occluded
- Can be CPU-intensive as it relies on scene geometry
- Hybrid method: create a depth map like in shadow mapping, use this instead of scene geometry to compute shadow volumes

(McCool, 2000) 

Conclusion

Shadow mapping is an influential and extensible method for efficiently producing realistic shadows.

Questions?



References

Agrawala, M., Ramamoorthi, R., Heirich, A., & Moll, L. (2000). Efficient Image-based Methods for Rendering Soft Shadows. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 375–384). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Chen, S. E., & Williams, L. (1993). View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 279–288). New York, NY, USA: ACM.

Kilgard, M. (2002). *Shadow Mapping with Today's OpenGL Hardware.* Powerpoint presented at SIGGRAPH 2002 Course 31.

Lance Williams. Retrieved from http://5dinstitute.org/people/lance-williams

Lance Williams. Retrieved from https://research.nvidia.com/users/lance-williams

Lance Williams. Retrieved from https://www.linkedin.com/pub/lance-williams/3/746/591

McCool, M. D. (2000). Shadow Volume Reconstruction from Depth Maps. *ACM Trans. Graph., 19*(1), 1–26.

Reeves, W. T., Salesin, D. H., & Cook, R. L. (1987). Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 283–291). New York, NY, USA: ACM.

Shadow mapping. (2014, November 29). In *Wikipedia, the free encyclopedia.* Retrieved from http://en.wikipedia.org/w/index.php?title=Shadow_mapping&oldid=632600511

Shadow volume. (2014, December 11). In *Wikipedia, the free encyclopedia.* Retrieved from http://en.wikipedia.org/w/index.php?title=Shadow_volume&oldid=630190017

Williams, L. (1978). Casting Curved Shadows on Curved Surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 270–274). New York, NY, USA: ACM.