Color (wrap up)
Shadows

Wolfgang Heidrich

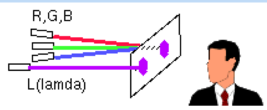
Course News

Assignment 3 (project)
- Due April 1

Reading
- Chapter 11.8, 10

Color Matching Experiments

Performed in the 1930s



Idea: perceptually based measurement
- Shine given wavelength (λ) on a screen
- User must control three pure lights producing three other wavelengths (say R=700 nm, G=546 nm, and B=438 nm)
- Adjust intensity of RGB until colors are identical

Color Matching Experiment

Results
- It was found that any color S(λ) could be matched with three suitable primaries A(λ), B(λ), and C(λ)
  - *Used monochromatic light at 438, 546, and 700 nanometers*
- Also found the space is linear, I.e. if

$$R(\lambda) \equiv S(\lambda)$$

then

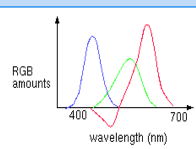
$$R(\lambda) + M(\lambda) \equiv S(\lambda) + M(\lambda)$$

and

$$k \cdot R(\lambda) \equiv k \cdot S(\lambda)$$

Negative Lobes

Actually:
- Exact target match possible sometimes requires “negative light”



- Some red has to be added to target color to permit exact match using “knobs” on RGB intensity output
- Equivalent mathematically to removing red from RGB output

Notation

Don’t confuse:
- Primaries: the spectra of the three different light sources: **R**, **G**, **B**
  - *For the matching experiments, these were **monochromatic** (I.e. single wavelength) light!*
  - *Primaries for displays usually have a wider spectrum*
- Coefficients *R, G, B*
  - *Specify how much of **R**, **G**, **B** is in a given color*
- Color matching functions: r(λ), g (λ), b (λ)
  - *Specify how much of **R**, **G**, **B** is needed to produce a color that is a metamer for pure monochromatic light of wavelength λ*

Negative Lobes

So:

- Can't generate all other wavelengths with any set of three positive monochromatic lights!

Solution:

- Convert to new synthetic "primaries" to make the color matching easy

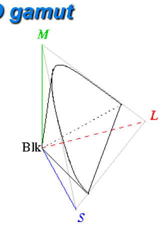
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2.36460 & -0.51515 & 0.00520 \\ -0.89653 & 1.42640 & -0.01441 \\ -0.46807 & 0.08875 & 1.00921 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Note:

- **R, G, B** are the same monochromatic primaries as before
- The corresponding matching functions $x(\lambda)$, $y(\lambda)$, $z(\lambda)$ are now positive everywhere
- But the primaries contain "negative" light contributions, and are therefore not physically realizable

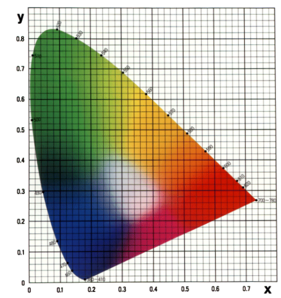
CIE Gamut and λ Chromaticity Diagram

3D gamut

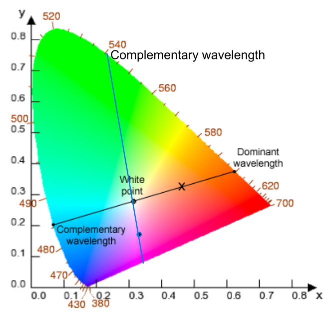


Chromaticity diagram

- Hue only, no intensity



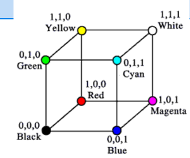
Color Interpolation, Dominant & Opponent Wavelength



RGB Color Space (Color Cube)

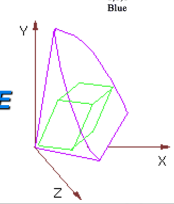
Define colors with (r, g, b) amounts of red, green, and blue

- Used by OpenGL
- Hardware-centric
- Describes the colors that can be generated with specific RGB light sources



RGB color cube sits within CIE

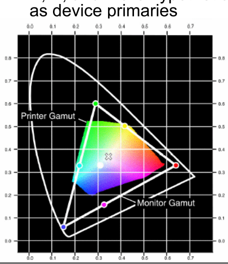
- Subset of perceivable colors
- Scaled, rotated, sheared cube

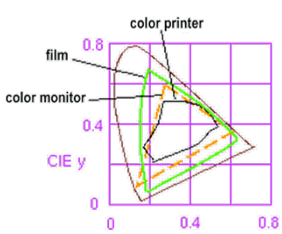


Device Color Gamuts

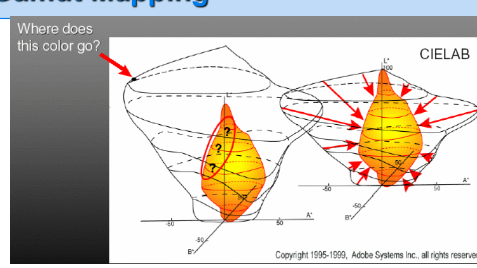
Use CIE chromaticity diagram to compare the gamuts of various devices

- X, Y, and Z are hypothetical light sources, not used in practice as device primaries





Gamut Mapping

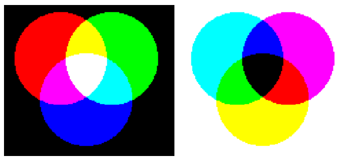


Additive vs. Subtractive Colors

Additive: light
- Monitors, LCDs
- RGB model

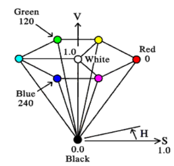
Subtractive: pigment
- Printers
- CMY(K) model

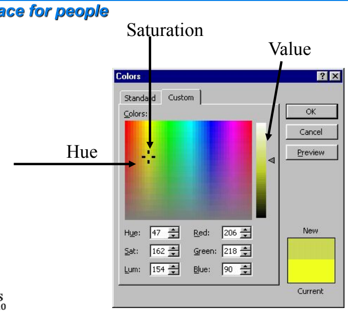
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



HSV Color Space

More intuitive color space for people
- H = Hue
- S = Saturation
- V = Value
  - *Or brightness B*
  - *Or intensity I*





Monitors

Monitors have nonlinear response to input
- Characterize by gamma

$$\text{displayedIntensity} = a^{\gamma} \cdot \text{maxIntensity}$$

Gamma correction

$$\text{displayedIntensity} = \left(a^{1/\gamma}\right)^{\gamma} \cdot \text{maxIntensity}$$

Gamma for CRTs:
- Around 2.4

Shadows

Wolfgang Heidrich

Shadows

What are shadows?
- What distinguishes a point in shadow from a lit point?

Shadows

Types of light sources
- Point, directional
- Area lights and generally shaped lights
  - *Not considered here*
  - *Later: ray-tracing for such light sources*

Problem statement
- A shadow algorithm for point and directional lights determines which scene points are
  - *Visible from the light source (I.e. illuminated)*
  - *Invisible from the light source (I.e. in shadow)*
- Thus: shadow casting is a visibility problem!

Types of Shadow Algorithms

Object Space

- Like object space visibility algorithms, the method computes in object space which polygon parts that are illuminated and which are in shadow
  - *Individual parts are then drawn with different intensity*
- Typically slow, O(n^2), not for dynamic scenes

Image Space

- Determine visibility per pixel in the final image
  - *Sort of like depth buffer*
  - *Shadow maps*
  - *Shadow volumes*

Wolfgang Heidrich

Credits

- The following shadow mapping slides are taken from Mark Kilgard's OpenGL course at Siggraph 2002.

Wolfgang Heidrich

Shadow Mapping Concept (1)

Depth testing from the light's point-of-view

- Two pass algorithm
- First, render depth buffer from the light's point-of-view
  - The result is a "depth map" or "shadow map"
  - Essentially a 2D function indicating the depth of the closest pixels to the light
- This depth map is used in the second pass

Wolfgang Heidrich

Shadow Mapping Concept (2)

Shadow determination with the depth map

- Second, render scene from the eye's point-of-view
- For each rasterized fragment
  - Determine fragment's XYZ position relative to the light
  - This light position should be setup to match the frustum used to create the depth map
  - Compare the depth value at light position XY in the depth map to fragment's light position Z

Wolfgang Heidrich

The Shadow Mapping Concept (3)

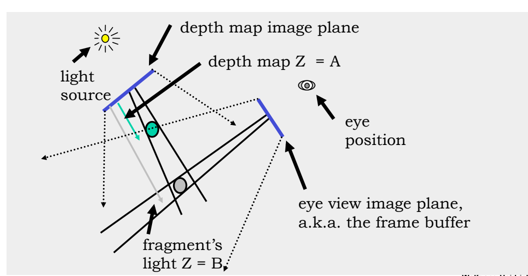
The Shadow Map Comparison

- Two values
  - A = Z value from depth map at fragment's light XY position
  - B = Z value of fragment's XYZ light position
- If B is greater than A, then there must be something closer to the light than the fragment
  - Then the fragment is shadowed
- If A and B are approximately equal, the fragment is lit

Wolfgang Heidrich

Shadow Mapping with a Picture in 2D (1)

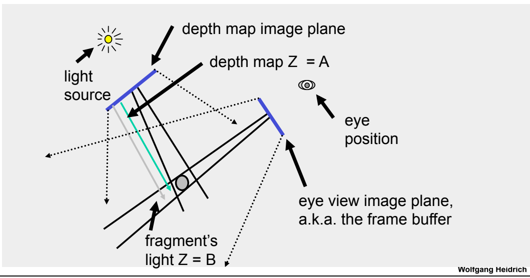
The A < B shadowed fragment case



Wolfgang Heidrich

Shadow Mapping with a Picture in 2D (2)

The A = B lit fragment case

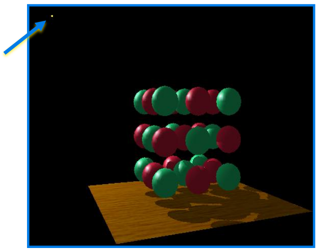


Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (1)

A scene with fairly complex shadows

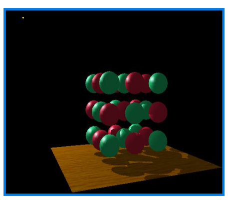
the point light source



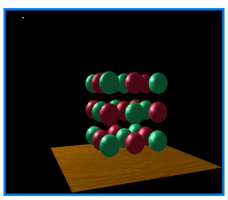
Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (2)

Compare with and without shadows



with shadows

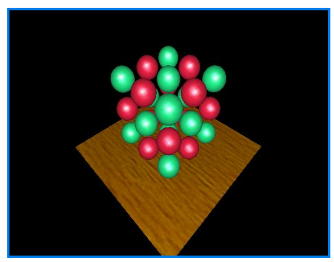


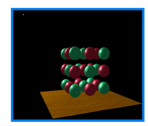
without shadows

Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (3)

The scene from the light's point-of-view



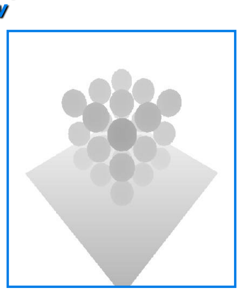


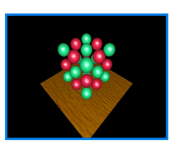
FYI: from the eye's point-of-view again

Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (4)

The depth buffer from the light's point-of-view



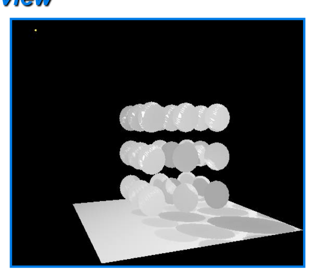


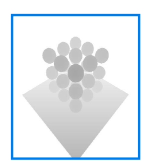
FYI: from the light's point-of-view again

Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (5)

Projecting the depth map onto the eye's view



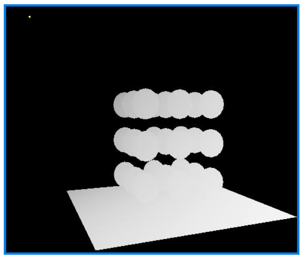


FYI: depth map for light's point-of-view again

Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (6)

Projecting light's planar distance onto eye's view

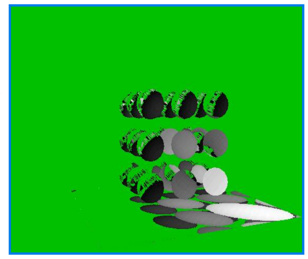


Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (6)

Comparing light distance to light depth map

Green is where the light planar distance and the light depth map are approximately equal



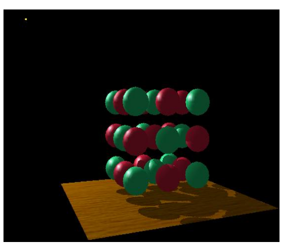
Non-green is where shadows should be

Wolfgang Heidrich

Visualizing the Shadow Mapping Technique (7)

Complete scene with shadows

Notice how specular highlights never appear in shadows

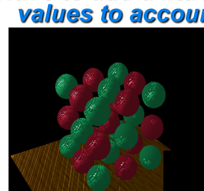


Notice how curved surfaces cast shadows on each other

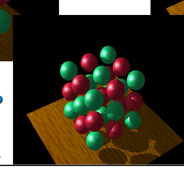
Wolfgang Heidrich

In Practice: Depth Map Precision Issues

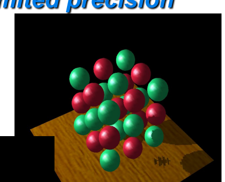
Have to add a little offset to depth map values to account for limited precision



Too little bias, everything begins to shadow



Just right



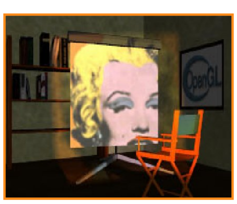
Too much bias, shadow starts too far back

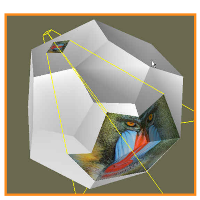
Wolfgang Heidrich

What is Projective Texturing?

An intuition for projective texturing

- The slide projector analogy





Wolfgang Heidrich

About Projective Texturing (1)

First, what is perspective-correct texturing?

- Normal 2D texture mapping uses (s, t) coordinates
- 2D perspective-correct texture mapping
  - *Means (s, t) should be interpolated linearly in eye-space*
  - *So compute per-vertex s/w, t/w, and 1/w*
  - *Linearly interpolated these three parameters over polygon*
  - *Per-fragment compute s' = (s/w) / (1/w) and t' = (t/w) / (1/w)*
  - *Results in per-fragment perspective correct (s', t')*

Wolfgang Heidrich

About Projective Texturing (2)

So what is projective texturing?

- Now consider homogeneous texture coordinates
  - *(s, t, r, q) --> (s/q, t/q, r/q)*
  - *Similar to homogeneous clip coordinates where (x, y, z, w) = (x/w, y/w, z/w)*
- Idea is to have (s/q, t/q, r/q) be projected per-fragment

Back to the Shadow Mapping Discussion . . .

Assign light-space texture coordinates to polygon vertices

- Transform eye-space (x, y, z, w) coordinates to the light's view frustum (match how the light's depth map is generated)
- Further transform these coordinates to map directly into the light view's depth map
  - *Expressible as a projective transform*
- (s/q, t/q) will map to light's depth map texture

Shadow Map Operation

Next Step:

- Compare depth map value to distance of fragment from light source
- Different GPU generations support different means of implementing this
  - *Today's GPUs: pixel shader!*
  - *Earlier: special hardware for implmenting this feature (e.g. SGI), or just using alpha blending [Heidrich'99]*

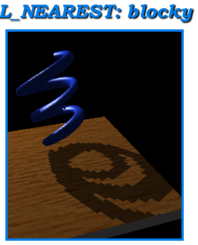
Issues with Shadow Mapping (1)

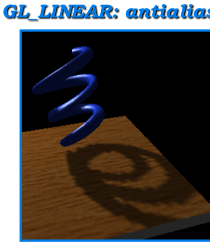
Not without its problems

- Prone to aliasing artifacts
  - *"percentage closer" filtering helps this*
  - *normal color filtering does not work well*
- Depth bias is not completely foolproof
- Requires extra shadow map rendering pass and texture loading
- Higher resolution shadow map reduces blockiness
  - *but also increase texture copying expense*

Hardware Shadow Map Filtering Example

GL_NEAREST: blocky ***GL_LINEAR: antialiased edges***





Low shadow map resolution used to heightens filtering artifacts

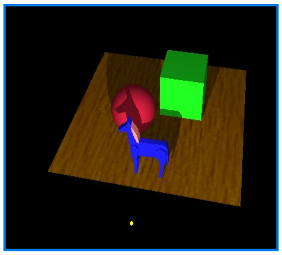
Issues with Shadow Mapping (2)

Not without its problems

- Shadows are limited to view frustums
  - *could use six view frustums for omni-directional light*
- Objects outside or crossing the near and far clip planes are not properly accounted for by shadowing
  - *move near plane in as close as possible*
  - *but too close throws away valuable depth map precision when using a projective frustum*

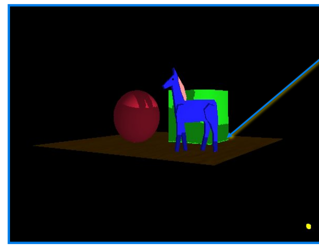
More Examples

Complex objects all shadow



More Examples

Even the floor casts shadow

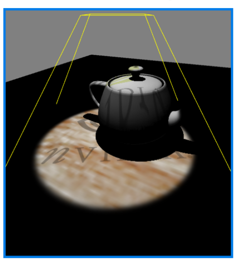


Note shadow leakage due to infinitely thin floor

Could be fixed by giving floor thickness

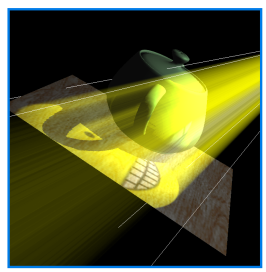
Combining Projective Texturing for Spotlights

Use a spotlight-style projected texture to give shadow maps a spotlight falloff



Combining Shadows with Atmospherics

Shadows in a dusty room



Simulate atmospheric effects such as suspended dust

1) *Construct shadow map*
2) *Draw scene with shadow map*
3) *Modulate projected texture image with projected shadow map*
4) *Blend back-to-front shadowed slicing planes also modulated by projected texture image*

Credit: Cass Everitt

Shadow Maps

Approach for shadows from point light sources

- Surface point is in shadow if it is not visible from the light source
- Use depth buffer to test visibility:
  - *Render scene from the point light source*
  - *Store resulting depth buffer as texture map*
  - *For every fragment generated while rendering from the camera position, project the fragment into the depth texture taken from the camera, and check if it passes the depth test.*

Shadow Volumes

Use new buffer: stencil buffer

- Just another channel of the framebuffer
- Can count how often a pixel is drawn

Algorithm (1):

- Generate silhouette polygons for all objects
  - *Polygons starting at silhouette edges of object*
  - *Extending away from light source towards infinity*
  - *These can be computed in vertex programs*

Shadow Volumes

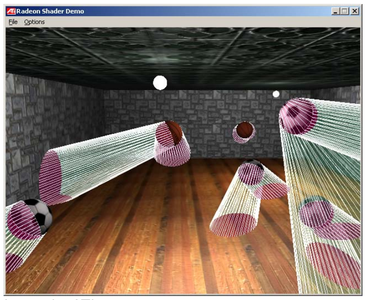


Image by ATI

Shadow Volumes

Algorithm (2):

- Render all original geometry into the depth buffer
  - *I.e. do not draw any colors (or only draw ambient illumination term)*
- Render front-facing silhouette polygons while incrementing the stencil buffer for every rendered fragment
- Render back-facing silhouette polygons while decrementing the stencil buffer for every rendered fragment
- Draw illuminated geometry where the stencil buffer is 0, shadow elsewhere

Shadow Volumes

Image by ATI

Shadow Volumes

Discussion:

- Object space method therefore no precision issues
- Lots of large polygons: can be slow
  - *High geometry count*
  - *Large number of pixels rendered*

Coming Up:

Friday

- Ray-tracing

Next Week:

- Global illumination