Color
Week 10, Wed 5 Nov 2003

News
• yet more extra office hours
  – Tue 11-1 (AW xtra)
  – Wed 1-2 (AW lab), 2-3 (PZ lab)
  – Thu 11-1 (AW, AG xtra) 12-1 (AG lab)
  – Fri 10-11 (AG lab), 11:30-1:30 (AW, AG xtra)
• I’m at a conference Fri pm – Mon pm
  – guest lecture Monday: Abhijeet Ghosh
  – my personal mail response will be slow
  – use newsgroup or email to TAs
    • if can’t post remotely, try unsub/sub or port forward
• homework 1 pickup again end of class

Picking Hints
• use OpenGL picking to find correct face
• plane: vectors from face verts, construct normal
• 4 lines: gluUnProject
  – rect around pick xy point, z = 0 and z = 1
  – visual debugging: try drawing line in scene
  – print out matrices, see if look right
    • make sure to grab them when they’re correct
    • confusing glGetDouble params: MODELVIEW_MATRIX
• calculate line/plane intersection
  – nudge outwards along normal

Flying Hints
• spec: move wrt current camera coord sys
  – gluLookAt difficult
    • transform from roll/pitch/yaw/to eye/lookat/up
  – cumulative Euler angles difficult
    • transform from current axes (xy/z) to new basis vector
      set in world coords
    – not even just each mouse drag: each transformation!
    – roll/pitch/yaw: last one wrong no matter which order you pick
      – heading not same as direction of motion
  – incremental Euler angles easy
    • want to just use current camera coord sys axes!

Incremental Euler Approach
• assume you know current coord sys
  – drag means motion wrt simple axis (x, y, or z)
• storing roll/pitch/yaw/formward values
  – do not keep cumulative values!
  – do purely incremental
    • only nonzero during drag
      • all three axes won’t be active at once
  – apply new incremental motion so change to new coord sys

Readings
• Chapter 1.4: color
• plus supplemental reading:
    – pages 4-24 required
  – http://graphics.stanford.edu/courses/cs448b-02-spring/l4cdrom.pdf

©Tamara Munzner
Matrix Stack As Calculator, Storage

- if not saving cumulative values, how do you know where you are?
  - if careful to segregate modelling transforms with push/pop, current viewing transformation stored in matrix stack!
  - don’t just erase with glIdentity
  - reuse stack values from last frame instead

Matrix Stack As Calculator, Storage

- transformation order problem
  - stack only supports $p' = \text{Current Incr} \ p$
  - want $\text{p'} = \text{Incr Current} \ p$

- read out stack into temporary matrix
  - glGetDoublev, just like when you unproject
  - then wipe stack, issue incr, issue current
  - now stack has correct new value, life is good

- uses stack to both calculate and to store

University of British Columbia
CPSC 414 Computer Graphics

Visibility recap

The Z-Buffer Algorithm

- augment color framebuffer with z-buffer or depth buffer which stores Z value at each pixel
  - at frame beginning, initialize all pixel depths to $\infty$
  - when rasterizing, interpolate depth ($Z$) across polygon and store in pixel of Z-buffer
  - suppress writing to a pixel if its Z value is more distant than the Z value already stored there

  - depth-buffer essentially stores $1/Z$, rather than $Z$

Z-Buffer Pros

- simple!!!
- easy to implement in hardware
- polygons can be processed in arbitrary order
- easily handles polygon interpenetration

Z-Buffer Cons

- lots of memory (e.g. $1280 \times 1024 \times 32$ bits)
  - with 16 bits cannot discern millimeter differences in objects at 1 km distance
- Read-Modify-Write in inner loop requires fast memory
- hard to do analytic antialiasing
  - we don’t know which polygon to map pixel back to
- hard to simulate translucent polygons
  - we throw away color of polygons behind closest one
The A-Buffer
- antialiased, area-averaged accumulation buffer
  - z-buffer: one visible surface per pixel
  - A-buffer: linked list of surfaces

Hidden Surface Removal
- image-space algorithms
  - Z-buffer, Warnock's
  - perform visibility test for every pixel independently
  - performed late in rendering pipeline, resolution dependent
- object-space algorithms
  - painter's algorithm: depth-sorting, BSP trees
  - determine visibility on a polygon level in camera coordinates
  - early in rendering pipeline (after clipping)
  - resolution independent
  - expensive

University of British Columbia
CPSC 414 Computer Graphics

Color
To understand how to make realistic images, we need a basic understanding of the physics and physiology of vision. Here we step away from the code and math for a bit to talk about basic principles.

Basics Of Color
- elements of color:
  - Perception
  - Illumination

Basics of Color
- Physics:
  - Illumination
    - Electromagnetic spectra
  - Reflection
    - Material properties
    - Surface geometry and microgeometry (i.e., polished versus matte versus brushed)
- Perception
  - Physiology and neurophysiology
  - Perceptual psychology
Electromagnetic Spectrum

- Sun or light bulbs emit all frequencies within the visible range to produce what we perceive as the "white light"

White Light and Color

- when white light is incident upon an object, some frequencies are reflected and some are absorbed by the object
- combination of frequencies present in the reflected light that determines what we perceive as the color of the object

Sunlight Spectrum

- hue (or simply, "color") is dominant wavelength
- integration of energy for all visible wavelengths is proportional to intensity of color

Saturation or Purity of Light

- how washed out or how pure the color of the light appears
- contribution of dominant light vs. other frequencies producing white light
Intensity, Brightness

- intensity: radiant energy emitted per unit of time, per unit solid angle, and per unit projected area of the source (related to the luminance of the source)
- brightness: perceived intensity of light

Humans and Light

- when we view a source of light, our eyes respond to:
  - hue: the color we see (red, green, purple)
    - dominant frequency
  - saturation: how far is color from grey
    - how far is the color from grey (pink is less saturated than red, sky blue is less saturated than royal blue)
  - brightness: how bright is the color
    - how bright are the lights illuminating the object?

Physiology of Vision

- The eye:
- The retina
  - Rods
  - Cones
    - Color!

Trichromacy

- three types of cones
  - L or R, most sensitive to red light (610 nm)
  - M or G, most sensitive to green light (560 nm)
  - S or B, most sensitive to blue light (430 nm)

- color blindness results from missing cone type(s)

Physiology of Vision

- The center of the retina is a densely packed region called the fovea.
  - Cones much denser here than the periphery

Metamers

- a given perceptual sensation of color derives from the stimulus of all three cone types

- identical perceptions of color can thus be caused by very different spectra
Metamer Demo


Adaptation, Surrounding Color

- color perception is also affected by
  - adaptation (stare at a light bulb... don't)
  - surrounding color/intensity:
    - simultaneous contrast effect

Combining Colors

Additive (RGB)
Shining colored lights on a white ball

Subtractive (CMYK)
Mixing paint colors and illuminating with white light

Color Spaces

- Three types of cones suggests color is a 3D quantity. How to define 3D color space?
- Idea:
  - Shine given wavelength (\(\lambda\)) on a screen
  - User must control three pure lights producing three other wavelengths (say R=700nm, G=546nm, and B=436nm)
  - Adjust intensity of RGB until colors are identical

Negative Lobes

- Exact target match with phosphors not possible
  - Some red had to be added to target color to permit exact match using “knobs” on RGB intensity output of CRT
  - Equivalently (theoretically), some red could have been removed from CRT output
  - Figure shows that red phosphor must remove some cyan for perfect match
  - CRT phosphors cannot remove cyan, so 500 nm cannot be generated
Negative Lobes

- can’t generate all other wavelengths with any set of three monochromatic lights!
- solution: convert to new synthetic coordinate system to make the job easy

CIE Color Space

- Target spectrum matched by finding corresponding X, Y, and Z quantities
  - Integrate product of spectral power and each of the three matching curves over all wavelengths

CIE Color Space

- CIE defined three “imaginary” lights X, Y, and Z, any wavelength \( \lambda \) can be matched perceptually by positive combinations
- \[ X' = X / (X + Y + Z), \quad Y' = Y / (X + Y + Z), \quad Z' = 1 - X' - Y' \]
- The gamut of all colors perceivable is thus a three-dimensional shape in X, Y, Z
- \( \text{Color} = X'X + Y'Y + Z'Z \)

CIE Chromaticity Diagram (1931)

For simplicity, we often project to the 2D plane \( X' + Y' + Z' = 1 \)
- \( X' = X / (X + Y + Z) \)
- \( Y' = Y / (X + Y + Z) \)
- \( Z' = 1 - X' - Y' \)

Device Color Gamuts

- Since X, Y, and Z are hypothetical light sources, no real device can produce the entire gamut of perceivable color
- Example: CRT monitor

For simplicity, we often project to the 2D plane \( X' + Y' + Z' = 1 \)
- \( X' = X / (X + Y + Z) \)
- \( Y' = Y / (X + Y + Z) \)
- \( Z' = 1 - X' - Y' \)
Device Color Gamuts

- We can use the CIE chromaticity diagram to compare the gamuts of various devices:
- Note, for example, that a color printer cannot reproduce all shades available on a color monitor

RGB Color Space (Color Cube)

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

**RGB Color Gamuts**

- The RGB color cube sits within CIE color space something like this:

**Converting Color Spaces**

- Simple matrix operation:

\[
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} = \begin{bmatrix}
X_R & X_G & X_B \\
Y_R & Y_G & Y_B \\
Z_R & Z_G & Z_B
\end{bmatrix} \begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

- The transformation \( C_2 = M_1^{-1} M_1 C_1 \) yields RGB on monitor 2 that is equivalent to a given RGB on monitor 1

**YIQ Color Space**

- YIQ is the color model used for color TV in America. Y is brightness, I & Q are color
  - Note: Y is the same as CIE’s Y
  - Result: Use the Y alone and backwards compatibility with B/W TV!
  - These days when you convert RGB image to B/W image, the green and blue components are thrown away and red is used to control shades of grey (usually)

**Converting Color Spaces**

- Converting between color models can also be expressed as such a matrix transform:

\[
\begin{bmatrix}
Y \\
I \\
Q
\end{bmatrix} = \begin{bmatrix}
0.30 & 0.59 & 0.11 \\
0.60 & -0.28 & -0.32 \\
0.21 & -0.52 & 0.31
\end{bmatrix} \begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

- Note the relative unimportance of blue in computing the Y
HSV Color Space

- A more intuitive color space
  - \( H = \) Hue
  - \( S = \) Saturation
  - \( V = \) Value (or brightness)

Perceptually Uniform Color Space

- Color space in which Euclidean distance between two colors in space is proportional to the perceived distance
  - CIE, RGB, not perceptually uniform
  - Example with RGB

Pick up Homework 1

- take 2