Color

CPSC 314

The Rendering Pipeline
**Modified Pipeline**

**Vertex shader**
- Replaces model/view, lighting, and perspective
- Have to implement these yourself
- But can also implement much more

**Fragment/pixel shader**
- Replaces texture mapping
- Fragment shader must do texturing
- But can do other things

**Vertex Program Properties**

*Run for every vertex, independently*
- Access to all per-vertex properties
  - Position, color, normal, texture coords, other custom properties
- Access to read/write registers for temporary results
  - Value is reset for every vertex
  - i.e. cannot pass information from one vertex to the next
- Access to read-only registers
  - Global variables, like light position, transformation matrices
- Write output to a specific register for the resulting color
Vertex Shaders/Programs

**Concept:**
- Programmable pipeline stage
  - *Floating-point operations on 4 vectors*
    - Points, vectors, and colors!
- Replace all of
  - *Model/View Transformation*
  - *Lighting*
  - *Perspective projection*

Vertex Programs – Instruction Set

**Arithmetic Operations on 4-vectors:**
- ADD, MUL, MAD, MIN, MAX, DP3, DP4

**Operations on Scalars**
- RCP (1/x), RSQ (1/√x), EXP, LOG

**Specialty Instructions**
- DST (distance: computes length of vector)
- LIT (quadratic falloff term for lighting)

**Very latest generation:**
- Loops and conditional jumps
**Vertex Programming Example**

*Example (from Stephen Cheney)*

- Morph between a cube and sphere while doing lighting with a directional light source (gray output)
- Cube position and normal in attributes (input) 0,1
- Sphere position and normal in attributes 2,3
- Blend factor in attribute 15
- Inverse transpose model/view matrix in constants 12-14
  - *Used to transform normal vectors into eye space*
- Composite matrix is in 4-7
  - *Used to convert from object to homogeneous screen space*
- Light dir in 20, half-angle vector in 22, specular power, ambient, diffuse and specular coefficients all in 21

---

#blend normal and position
# v = α v₁ + (1-α) v₂ = α(v₁-v₂) + v₂

MOV R5, v[2] ;
ADD R8, v[1], -R3 ;
ADD R6, v[0], -R5 ;
MAD R8, v[15].x, R8, R3
MAD R6, v[15].x, R6, R5 ;

# transform normal to eye space
DP3 R9.x, R8, c[12] ;
DP3 R9.z, R8, c[14] ;

# transform position and output
DP4 o[HPOS].x, R6, c[4] ;
DP4 o[HPOS].y, R6, c[5] ;
DP4 o[HPOS].z, R6, c[6] ;
DP4 o[HPOS].w, R6, c[7] ;

# normalize normal
DP3 R9.w, R9, R9 ;
RSQ R9.w, R9.w ;
MUL R9, R9.w, R9 ;

# apply lighting and output color
DP3 R0.x, R9, c[20] ;
DP3 R0.y, R9, c[22] ;
MOV R0.sw, c[21] ;
LIT R1, R0 ;
DP3 o[COL0], c[21], R1 ;
Skinning

Example by NVIDIA

Fragment Shader Generic Structure

constant registers
clamped diffuse and specular
texture coordinates

temporary registers
arithmetic logic unit

addressing instructions

rgba output
z-depth output

Figure 6.20. Generalized pixel shader. Variants in the pixel shader language primarily affect the way texture address instructions work, where temporary results can be stored, and whether the z-depth can be modified and output.
Fragment Shaders

- *Fragment shaders* operate on fragments in place of the texturing hardware
  - After rasterization, before any fragment tests or blending
- Input: The fragment, with screen position, depth, color, and a set of texture coordinates
- Access to textures and some constant data and registers
- Compute RGBA values for the fragment, and depth
  - Can also “kill “a fragment, that is throw it away
- Two types of fragment shaders: register combiners (GeForce4) and fully programmable (GeForceFX, Radeon 9700)

Cg

*Cg is a high-level language developed by NVIDIA*

- It looks like C or C++
- Actually a language and a runtime environment
  - Can compile ahead of time, or compile on the fly
  - Why compile on the fly?
- What it can do is tightly tied to the hardware
  - How does it know which hardware, and how to use it?
Vertex Program Example

```c
void CSE2v_fragmentLighting(float4 position : POSITION, 
    float3 normal : NORMAL, 
    out float4 oPosition : POSITION, 
    out float3 objectPos : TEXCOORD0, 
    out float3 oNormal : TEXCOORD1, 
    uniform float4x4 modelViewProj)
{
    oPosition = mul(modelViewProj, position);
    objectPos = position.xyz;
    oNormal = normal;
}
```

Pixel Program Example

```c
void CSE2f_basicLight(float4 position : TEXCOORD0, 
    float3 normal : TEXCOORD1, 
    out float4 color : COLOR, 
    uniform float3 globalAmbient, 
    uniform float3 lightColor, 
    uniform float3 lightPosition, 
    uniform float3 eyePosition, 
    uniform float3 Kd, 
    uniform float3 Ke, 
    uniform float3 Ka, 
    uniform float shininess)
{
    float3 P = position.xyz;
    float3 N = normalize(normal);
    // Compute the diffuse term
    float3 L = normalize(lightPosition - P);
    float diffuseLight = max(dot(N, L), 0.0);
    float3 diffuse = L * lightColor * diffuseLight;
    // Compute the specular term
    float3 V = normalize(position - P);
    float3 H = normalize(L + V);
    float specularLight = pow(max(dot(N, H), 0.0), shininess);
    if (diffuseLight < 0) specularLight = 0;
    float3 specular = Ke * lightColor * specularLight;
    color.xyz = max(0.0, diffuse + specular);
    color.w = 1;
}
```
Shadow Maps

*Shadow maps using the alpha test*

---

**Alpha shadow map:**

---
Shadow Maps

*Shadow map applied as projective texture*

- *Light source z*
  - Computed in fragment shader
Shadow Maps

*Subtract shadow map from computed depth*

- Also in fragment shader
- Wherever the result is not 0, we have shadow!

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

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

Color

CPSC 314
Simple Model of Color

• Simple model based on RGB triples
• Component-wise multiplication of colors
  \((a_0, a_1, a_2) \times (b_0, b_1, b_2) = (a_0b_0, a_1b_1, a_2b_2)\)

Why does this work?

Basics Of Color

Elements of color:

Illumination → Perception → Reflectance
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
Light Sources

Common light sources differ in the kind of spectrum they emit:

- Continuous spectrum
  - Energy is emitted at all wavelengths
    - Blackbody radiation
    - Tungsten light bulbs
    - Certain fluorescent lights
    - Sunlight
    - Electrical arcs
- Line spectrum
  - Energy is emitted at certain discrete frequencies

Blackbody Radiation

**Black body**

- Dark material, so that reflection can be neglected
- Spectrum of emitted light changes with temperature
  - This is the origin of the term “color temperature”
    - E.g. when setting a white point for your monitor
  - Cold: mostly infrared
  - Hot: redish
  - Very hot: bluish
- Demo:
  
  [Demo Link](http://www.mhhe.com/physsci/astronomy/applets/Blackbody/frame.html)
White Light

- Sun or light bulbs emit all frequencies within the visible range to produce what we perceive as the "white light"
- But the exact tone depends on the emitted spectrum

Sunlight Spectrum
Continuous Spectrum

**Example:**
- Sunlight
- Various “daylight” lamps

Line Spectrum

**Examples:**
- Ionized gases
- Lasers
- Some fluorescent lamps
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 determines what we perceive as the color of the object.

Physiology of Vision

The retina

- Rods
  - B/w, edges
- Cones
  - Color!
Physiology of Vision

Center of retina is densely packed region called the fovea.

- Cones much denser here than the periphery

Hue

Hue (or simply, "color") is dominant wavelength/frequency

- 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
- Saturation: how far is color from grey
  - *Pink is less saturated than red, sky blue is less saturated than royal blue*

**Intensity vs. Brightness**

**Intensity** : physical term

- **Measured** 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)

**Lightness/brightness: perceived intensity of light**

- Nonlinear
# Perceptual vs. Colorimetric Terms

<table>
<thead>
<tr>
<th>Perceptual</th>
<th>Colorimetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hue</td>
<td>Dominant wavelength</td>
</tr>
<tr>
<td>Saturation</td>
<td>Excitation purity</td>
</tr>
<tr>
<td>Lightness</td>
<td>Luminance</td>
</tr>
<tr>
<td>- Reflecting objects</td>
<td></td>
</tr>
<tr>
<td>Brightness</td>
<td>Luminance</td>
</tr>
<tr>
<td>- Light sources</td>
<td></td>
</tr>
</tbody>
</table>

# Color/Lightness Constancy

*Color perception also depends on surrounding*

- Colors in close proximity
- Illumination under which the scene is viewed
Adaptation, Surrounding Color

Color perception is also affected by

- Adaptation (move from sunlight to dark room)
- Surrounding color/intensity:
  - Simultaneous contrast effect

Color/Lightness Constancy

Do they match?

Image courtesy of John MCann

© Wolfgang Heidrich
Color/Lightness Constancy

Do they match?
Color/Lightness Constancy
Color/Lightness Constancy

Color Constancy

- Automatic “white balance” from change in illumination
- Vast amount of processing behind the scenes!
- Colorimetry vs. perception

From Color Appearance Models, fig 8-1

© Wolfgang Heidrich
Tristimulus Theory of Color Vision

- Although light sources can have extremely complex spectra, it was empirically that colors could be described by only 3 primaries.

- Colors that look the same but have different spectra are called metamers.

- Metamer demo:
  
  http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/color_theory.html

Color Matching Experiments

Performed in the 1930s

Idea: perceptually based measurement

- 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
Color Matching Experiment

Results

• It was found that any color \( S(\lambda) \) could be matched with three suitable primaries \( A(\lambda), B(\lambda), \) and \( C(\lambda) \)
  
  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 had to be added to target color to permit exact match using “knobs” on RGB intensity output
- Equivalently theoretically to removing red from RGB output
- Figure shows that red primary must remove some cyan for perfect match
- CRT phosphors cannot remove cyan, so 500 nm cannot be generated
**Negative Lobes**

*So:*

- Can’t generate all other wavelenths with any set of three positive monochromatic lights!
- Solution: convert to new synthetic coordinate system to make the job easy

---

**CIE Color Space**

- CIE defined three “imaginary” lights X, Y, and Z, any wavelength $\lambda$ can be matched perceptually by positive combinations

Note that:
- $X \sim R$
- $Y \sim G$
- $Z \sim B$
**Measured vs. CIE Color Spaces**

**Measured basis**
- Monochromatic lights
- Physical observations
- Negative lobes

**Transformed basis**
- "imaginary" lights
- All positive, unit area
- Y is luminance, no hue
- X,Z no luminance

---

**CIE Gamut and Chromaticity Diagram**

**3D gamut**

**Chromaticity diagram**
- Hue only, no intensity
Facts about the CIE “Horseshoe” Diagram

• All visible colors lie inside the horseshoe
  – Result from color matching experiments
• Spectral (monochromatic) colors lie around the border
  – The straight line between blue and red contains the purple tones
• Colors combine linearly (i.e. along lines), since the xy-plane is the projection of a linear space

Facts about the CIE “Horseshoe” Diagram (cont.)

A point C can be chosen as a white point corresponding to an illuminant

• Usually this point is of the curve swept out by the black body radiation spectra for different temperatures
• Relative to C, two colors are called complementary if they are located along a line segment through C, but on opposite sides (i.e. C is an affine combination of the two colors)
• The dominant wavelength of the color is found by extending the line from C through the color to the edge of the diagram
• Some colors (i.e. purples) do not have a dominant wavelength, but their complementary color does

© Wolfgang Heidrich
CIE Diagram

- Blackbody curve
- Illumination:
  - Candle 2000K
  - Light bulb 3000K (A)
  - Sunset/sunrise 3200K
  - Day light 6500K (D)
  - Overcast day 7000K
  - Lightning > 20,000K

Color Interpolation, Dominant & Opponent Wavelength

Complementary 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 color space*
- 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**

Where does this color go?

---

**Additive vs. Subtractive Colors**

**Additive: light**
- Monitors, LCDs
- RGB model

**Subtractive: pigment**
- Printers
- CMY(K) model

\[
\begin{align*}
C & = 1 \\
M & = 1 - G \\
Y & = 1 - B \\
\end{align*}
\]
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
  - displayedIntensity = a^γ (maxIntensity)

Gamma correction
- displayedIntensity = (a^{1/γ})^γ (maxIntensity)
  = a (maxIntensity)

Gamma for CRTs:
- Around 2.4
Coming Up...

Friday:
- Ray-tracing

Monday:
- Global illumination / curves and surfaces
- Last topics to be on the final

Wednesday:
- Current research topics in graphics
- Advanced graphics courses at UBC
- Lab tour