Color

CPSC 314

The Rendering Pipeline

Geometry Processing

Geometry Database

Model/View Transform. → Lighting → Perspective Transform. → Clipping

Fragment Processing

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

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/v^2), 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

---

**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 combinators (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

```cpp
void CUBE_fragmentLighting(float4 position : POSITION,
                          float3 normal : NORMAL,
                          float3 objectPos : OBJECTPOS,
                          float3 objectNorm : OBJECTNORM,
                          uniform float4x4 modelViewProj) :
{
    position = mul(modelViewProj, position);
    objectPos = position.xyz;
    normal = normal;
}
```

Pixel Program Example

```cpp
cull singlePixelProjection vertexPosition : POSITION,
uniform float4x4 modelViewProj : MODELVIEWPROJ:
{
    // Compute the model-view-projection matrix
    // and use it to transform the vertex position
    // into clip space.
    // ...
}
```

Shadow Maps

**Shadow maps using the alpha test**

**Alpha shadow map:**

- Computed in fragment shader

**Shadow map applied as projective texture**
**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

**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

---

**Color**

*CPSC 314*
**Simple Model of Color**

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

- Why does this work?

---

**Basics of Color**

**Elements of color:**

- Illumination
- Perception

**Physics**

- Illumination
  - Electromagnetic spectra
- Reflection
  - Material properties
    - Surface geometry and microgeometry (i.e., polished versus matte versus brushed)

**Perception**

- Physiology and neurophysiology
- Perceptual psychology

---

**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

---

**Electromagnetic Spectrum**

**The Electromagnetic Spectrum**

---

**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: reddish
  - Very hot: bluish
- Demo: [http://www.mhhe.com/physsci/astromapplets/Blackbody/Frame.html](http://www.mhhe.com/physsci/astromapplets/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 that 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

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

Perceptual | Colorimetric
--- | ---
Hue | Dominant wavelength
Saturation | Excitation purity
Lightness | Luminance
  - Reflecting objects | Brightness | Luminance
  - Light sources

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 Luca Ncerta
**Tristimulus Theory of Color Vision**

- Although light sources can have extremely complex spectra, it was empirically that colors could be described by only 3 **primitives**.
- Colors that look the same but have different spectra are called **metamers**.

**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 $R=700\text{nm}$, $G=546\text{nm}$, and $B=436\text{nm}$)
- Adjust intensity of RGB until colors are identical

**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 wavebands 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:

\[
\begin{align*}
X - R & \\
Y - G & \\
Z - B &
\end{align*}
\]

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
CIE Diagram

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

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 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

Additive vs. Subtractive Colors

Additive: light
- Monitors, LCDs
- RGB model

Subtractive: pigment
- Printers
- CMY(K) model
**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**
  - \( \text{displayedIntensity} = a^T \text{(maxIntensity)} \)

**Gamma correction**

- \( \text{displayedIntensity} = (\text{maxIntensity})^{\frac{1}{\gamma}} \)
- \( = a \text{(maxIntensity)} \)

**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