University of British Columbia
CPSC 414 Computer Graphics

Advanced Rendering
Final Review
Week 13, Wed 26 Nov 2003

News
• proj 3 demo signup continues
  – record carefully, do not miss your demo slot!
• policy clarification
  – cannot use code from web for anything except low-level utilities like file loading (image files, object files)
  – must cite in writeup all web pages that were major inspiration

Schedule: Lab Hours for P3
• Mon Dec 1
  – AG 10-12, AW 12-2
• Tue Dec 2
  – AG 10-12, AW 12-2, TM 2-4
• Wed Dec 3
  – AW 1-2, PZ 2-4
• Thu Dec 4
  – AG 11-1
• Fri Dec 5
  – AG 10-11, PZ 11-1

Schedule: Lectures
• Wed (today)
  – advanced rendering, final review
• Fri
  – (finish review?), other graphics courses
  – evaluations, graphics in movies
  • Pixar shorts

Ray Tracing
• cast a ray from the viewer’s eye through each pixel
• compute intersection of ray with objects from scene
• closest intersecting object determines color

University of British Columbia
CPSC 414 Computer Graphics

Advanced Rendering

© Tamara Munzner

© Tamara Munzner

© Tamara Munzner

© Tamara Munzner

© Tamara Munzner

© Tamara Munzner
Recursive Ray Tracing

- cast ray from intersected object to light sources and determine shadow/lighting conditions
- also spawn secondary rays
  - reflection rays and refraction rays
  - use surface normal as guide (angle of incidence equals angle of reflection)
  - if another object is hit, determine the light it illuminates by recursing through ray tracing
- view-dependent

Radiosity

- rate at which energy emitted or reflected by a surface
  - conserve light energy in a volume
  - model light transport until convergence
  - solution captures diffuse-diffuse bouncing of light
- recall radiative heat transfer

Comparison

- ray-tracing: great specular, approx diffuse
- radiosity: great diffuse, ignore specular
- advanced hybrids: combine them

Non-Photorealistic Rendering

- look of hand-drawn sketches or paintings

www.red3d.com/cwr/npr/

NPRQuake

www.cs.wisc.edu/graphics/Gallery/NPRQuake
Image-Based Rendering

- store and access only pixels
  - no geometry, no light simulation, etc!
  - massive compression possible (120:1)

- display time not tied to scene complexity
  - expensive rendering or real photographs

Logistics

- LSK 200, noon-3pm Tue Dec 9
- policies
  - must have student photo ID face up on desk
  - cannot take exam without photo ID
  - one piece of 8.5"x11" paper allowed
  - both sides handwritten
  - no other books or notes
  - nonprogrammable calculator OK
  - if you finish in last 15 minutes, stay put

Topics Covered

- rendering pipeline
- modelling transformations
- viewing transformations
- projections
- display lists
- picking
- lighting/shading
- rasterization/scan conversion
- sampling/anti-aliasing
- animation
- texturing
- clipping
- quaternions
- visibility
- color
- visualization
- displays
- procedural approaches
- curves
- advanced rendering

Midterm Covered, Less Emphasis

- rendering pipeline
- modelling transformations
- viewing transformations
- projections
- display lists
- picking
- lighting/shading
- rasterization/scan conversion
- sampling/anti-aliasing
- animation
- texturing
- clipping
- quaternions
- visibility
- color
- visualization
- displays
- procedural approaches
- curves
- advanced rendering

- for these, look at midterm review notes
  - Week 7 Wed (16 Oct)

Sampling and Antialiasing
**Point Sampling**

- multiply sample grid by image intensity to obtain a discrete set of points, or samples.

**Spatial Domain**

- image as spatial signal

**Spatial Domain: Summing Waves**

- represent spatial signal as sum of sine waves (varying frequency and phase shift)
- very commonly used to represent sound “spectrum”

**Frequencies: Summing Spikes**

- $x$: wavelength, $y$: strength
- sine wave: impulse

**Fourier Transform Examples**

- square wave: infinite train of impulses

**Nyquist Rate**

- the lower bound on the sampling rate equals twice the highest frequency component in the image’s spectrum
- this lower bound is the Nyquist Rate
FallingBelowNyquistRate

- when sampling below Nyquist Rate, resulting signal looks like a lower-frequency one
  - this is aliasing!

• texels are specified by texture’s (u,v) space
  - a t each screen pixel, texel can be used to substitute a polygon’s surface property (color)
• we must map (u,v) space to polygon’s (s,t) space

Example Texture Map


texels are specified by texture’s (u,v) space

Texture Mapping

- texture map is an image, two-dimensional array of color values (texels)

Animation

algorithm:
simulating physics, parameterizing models

user:
keyframing
data:
motion capture

Texture Mapping
Texture Coordinate Transforms

\[ \text{glVertex3d} (s, s, s) \]
\[ \text{glTexCoord2d}(5, 5); \]
\[ \text{glVertex3d} (s, s, s) \]
\[ \text{glTexCoord2d}(1, 1); \]

Texture Mapping and Filtering

- in practice: 2 cases

Texture Minification Filters

- solution: precomputation
  - MIP-Mapping (Multum In Parvo)
  - “many things in a small place”
  - store not one texture image, but whole pyramid
  - resolution from level to level varies by factor of two
    (original resolution … 1x1)
  - every level is correctly filtered for its resolution

Clipping

- analytically calculating the portions of primitives within the viewport

Clipping Lines To Viewport

- combining trivial accepts/rejects
  - trivially accept lines with both endpoints inside all edges of the viewport
  - trivially reject lines with both endpoints outside the same edge of the viewport
  - otherwise, reduce to trivial cases by splitting into two segments
Cohen-Sutherland Line Clipping

- outcodes
  - 4 flags encoding position of a point relative to top, bottom, left, and right boundary
  - $OC(p1)=0010$
  - $OC(p2)=0000$
  - $OC(p3)=1001$

Sutherland-Hodgeman Polygon Clipping

- basic idea:
  - consider each edge of the viewport individually
  - clip the polygon against the edge equation
  - after doing all edges, the polygon is fully clipped

Sutherland-Hodgeman Clipping

- basic idea:
  - consider each edge of the viewport individually
  - clip the polygon against the edge equation
  - after doing all edges, the polygon is fully clipped

Rotation Methods

- 3x3 matrices
  - good: simple. bad: drifting, can’t interpolate
- Euler angles
  - good: can interpolate, no drift
  - bad: gimbal lock
- axis-angle
  - good: no gimbal lock, can interpolate
  - bad: can’t concatenate
- quaternions
  - good: solve all problems. bad: complex
Quaternions
• quaternion is a 4-D unit vector \( q = [x \ y \ z \ w] \)
  • lies on the unit hypersphere \( x^2 + y^2 + z^2 + w^2 = 1 \)
• for rotation about (unit) axis \( v \) by angle \( \theta \)
  • vector part = \( (\sin \frac{\theta}{2}) v = [x \ y \ z] \)
  • scalar part = \( (\cos \frac{\theta}{2}) = w \)
• rotation matrix
  \[
  \begin{bmatrix}
  1 - 2y^2 - 2z^2 & 2xy + 2wz & 2xz - 2wy \\
  2xy - 2wz & 1 - 2x^2 - 2z^2 & 2yz + 2wx \\
  2xz + 2wy & 2yz - 2wx & 1 - 2x^2 - 2y^2 \\
  \end{bmatrix}
  \]
• quaternion multiplication \( q_1 \cdot q_2 = [v_1, w_1] \cdot [v_2, w_2] = \left( \left[ (w_1v_2 + w_2v_1 + (v_1 \cdot v_2)), w_1w_2 - v_1 \cdot v_2 \right] \right) \)

Invisible Primitives
• why might a polygon be invisible?
  • polygon outside the field of view / frustum
    • clipping
  • polygon is back-facing
  • backface culling
  • polygon is occluded by object(s) nearer the viewpoint
    • hidden surface removal

Back-Face Culling
• on the surface of a closed manifold, polygons whose normals point away from
  the camera are always occluded:

Note: backface culling alone doesn’t solve the hidden-surface problem!

Back-face Culling: NDCS

Painter’s Algorithm
• draw objects from back to front
• problems: no valid visibility order for
  – intersecting polygons
  – cycles of non-intersecting polygons possible
BSP Trees

- preprocess: create binary tree
  - recursive spatial partition

- runtime: traverse tree
  - check node’s plane vs. eyepoint
  - recurse on far, draw object, recurse on near

Warnock’s Algorithm

- recursive
  - start at root viewport of entire window
  - clip objects to current viewport
  - trivial if 0 or 1 objects
  - subdivide if more
  - stop if trivial or down to one pixel

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 \( z \)

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
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?

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)

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

Color Constancy

- measured basis
  - monochromatic lights
  - physical observations
  - negative lobes

- transformed basis
  - "imaginary" lights
  - all positive, unit area
  - Y is luminance
Device Color Gamuts

- use CIE chromaticity diagram to compare the gamuts of various devices

The Gamma Problem

- device gamma
  - monitor: $I = A(k_1 D + k_2 V)^\gamma$
  - typical monitor $\gamma = 2.5$
  - LCD: nearly linear
- OS gamma
  - defined by operating system
  - inverse gamma curve $I^{1/\gamma}$
  - “gamma correction”

The Gamma Problem

- monitor:
  - back-to-front compositing
  - inverse gamma curve

Volume Rendering Algorithms

- ray casting
  - image order, forward viewing
- splatting
  - object order, backward viewing
- texture mapping hardware
  - object order
  - back-to-front compositing
Overview and Detail

- problem: getting lost
  - 1 window: lose track of navigation history
  - 2 windows: see overview and detail both
  - Focus+Context
    - 1 window combines overview and detail
    - carefully chosen distortion

Comparison

- array of small multiples
  - side by side comparison easier than temporal

Display Technologies

- CRT: Cathode Ray Tubes
- LCD: Liquid Crystal Displays
- plasma display panels
- DMD/DLP: micromirror array projectors
- display walls: tiled projector array
Displays and Devices
• mobile display with laser to retina
• stereo glasses/display
• 3D scanners
  – laser stripe + camera
  – laser time-of-flight
  – cameras only, depth from stereo
• Shape Tape
• haptics (Phantom)
• 3D printers

Procedural Approaches
• fractal landscapes

Procedural Approaches
• L-systems

Procedural Approaches
• particle systems

Procedural Approaches
• Perlin noise

Comparing Hermite and Bezier
Hermite

Comparing Hermite and Bezier
Bezier

Sub-Dividing Bezier Curves
• step 3: find the midpoint of the line joining $M_{012}$ $M_{123}$, call it $M_{0123}$
de Casteljau’s Algorithm

- can find the point on a Bezier curve for any parameter value \( t \) with similar algorithm
  - for \( t=0.25 \), instead of taking midpoints take points 0.25 of the way

\[ P_0, P_1, P_2, P_3 \]

\[ M_{01}, M_{12}, M_{23} \]

\( t=0.25 \)

Demo: [www.saltire.com/applets/advanced_geometry/spline/spline.htm](http://www.saltire.com/applets/advanced_geometry/spline/spline.htm)

Continuity

- piecewise Bezier: no continuity guarantees

- continuity definitions
  - \( C^0 \): share join point
  - \( C^1 \): share continuous derivatives
  - \( C^2 \): share continuous second derivatives

B-Spline

- \( C^0 \), \( C^1 \), and \( C^2 \) continuous
- piecewise: locality of control point influence