University of British Columbia
CPSC 314 Computer Graphics
May-June 2005

Tamara Munzner

Animation, Advanced Rendering,
Final Review

Week 6, Tue Jun 14

http://www.ugrad.cs.ubc.ca/~cs314/Vmay2005
News

- P4 grading
  - 4:30-5:45 Wed Jun 22
Review: Volume Graphics

- for some data, difficult to create polygonal mesh
- voxels: discrete representation of 3D object
  - volume rendering: create 2D image from 3D object
- translate raw densities into colors and transparencies
  - different aspects of the dataset can be emphasized via changes in transfer functions
Review: Volume Graphics

- **pros**
  - formidable technique for data exploration

- **cons**
  - rendering algorithm has high complexity!
  - special purpose hardware costly (~$3K-$10K)

volumetric human head (CT scan)
Review: Isosurfaces

- 2D scalar fields: isolines
  - contour plots, level sets
  - topographic maps
- 3D scalar fields: isosurfaces
Review: Isosurface Extraction

- array of discrete point samples at grid points
  - 3D array: voxels
- find contours
  - closed, continuous
  - determined by iso-value
- several methods
  - marching cubes is most common

Iso-value = 5
Review: Marching Cubes

- create cube
- classify each voxel
- binary labeling of each voxel to create index
- use in array storing edge list
  - all 256 cases can be derived from 15 base cases
- interpolate triangle vertex
- calculate the normal at each cube vertex
- render by standard methods
Review: Direct Volume Rendering Pipeline

- do not compute surface
Review: Transfer Functions To Classify

- map data value to color and opacity
  - can be difficult, unintuitive, and slow

Gordon Kindlmann
Review: Volume Rendering Algorithms

- ray casting
  - image order, forward viewing

- splatting
  - object order, backward viewing

- texture mapping
  - object order
  - back-to-front compositing
Review: Ray Casting Traversal Schemes

Intensity
Max

Average
Accumulate
First

Depth
Review: Information Visualization

- interactive visual representation of abstract data
  - help human perform some task more effectively
- bridging many fields
  - graphics: interacting in realtime
  - cognitive psych: finding appropriate representation
  - HCI: using task to guide design and evaluation
- external representation
  - reduces load on working memory
  - offload cognition
- familiar example: multiplication/division
- infovis example: topic graphs
Review: Shneiderman mantra

- overview, zoom and filter, details-on-demand
Review: Overviews - SeeSoft

- colored lines of code: lines one pixel high
Review: Focus+Context

- integrate overview and details into single view
  - H3: 3D fisheye
  - TreeJuxtaposer: stretch and squish
  - SpaceTree: collapse/expand
Review: 3D Extrusion vs. Linking

- perspective interferes with comparison
  - daily, weekly patterns hard to see
- linked cluster/calendar view more effective

[van Wijk and van Selow, Cluster and Calendar based Visualization of Time Series Data, InfoVis99, citeseer.nj.nec.com/vanwijk99cluster.html]
Review: Preattentive Visual Channels: Popout

- single channel processed in parallel for popout
  - visual attentional system not invoked
  - speed independent of distractor count
  - hue, shape, texture, length, width, size, orientation, curvature, intersection, intensity, flicker, direction of motion, stereoscopic depth, lighting direction,...

- multiple channels not parallel
  - search linear in number of distractor objects

[Chris Healey, Preattentive Processing, www.csc.ncsu.edu/faculty/healey/PP]
Review: Data Type Affects Channel Ranking

- spatial position best for all types
  - accuracy at judging magnitudes, from best to worst

[Mackinlay, Automating the Design of Graphical Presentations of Relational Information, ACM TOG 5:2, 1986]
Review: Coloring Categorical Data

- discrete small patches separated in space
- limited distinguishability: around 8-14
  - channel dynamic range: low
  - choose bins explicitly for maximum mileage
- maximally discriminable colors from Ware
  - maximal saturation for small areas
  - vs. minimal saturation for large areas

[Colin Ware, Information Visualization: Perception for Design. Morgan Kaufmann 1999. Figure 4.21]
Review: Rainbow Colormap Disadvantages

- perceptually nonlinear segmentation, hue unordered

- (partial) solution perceptually isolinear map


Review: Color Deficiency – vischeck.com

- 10% of males have red/green deficit

original  protanope  deuteranope  tritanope

Review: Space vs. Time: Showing Change

literal
<-----------------------------
time for time

abstract
space for time

animation: show time using temporal change
  - good: show process
  - good: compare by flipping between two things
  - bad: compare between many things
  interference from intermediate frames

[Outside In excerpt. www.geom.uiuc.edu/docs/outreach/oi/evert.mpg]
[www.astroshow.com/ccdpho/pluto.gif]
[Edward Tufte. The Visual Display of Quantitative Information, p 172]
Review: Space vs. Time: Showing Change

**literal**

<-----------------------------

time for time

**abstract**

------------------------------>

space for time

**small multiples: show time using space**

- overview: show each time step in array
- compare: side-by-side easier than temporal external cognition instead of internal memory
- general technique, not just for temporal changes

[Edward Tufte. The Visual Display of Quantitative Information, p 172]
Animation

(slides based on Robert Bridson’s CPSC 426 preview)

www.ugrad.cs.ubc.ca/~cs426
Computer Animation

- offline: generate a film, play it back later
- long ago reached the point of being able to render anything an artist could model
- problem is: how to model?
  - tools/UI for directly specifying model+motion (the traditional technique)
  - procedural modeling (e.g. particle systems)
  - data-driven modeling (e.g. motion capture)
  - physics-based modeling (e.g. fluid simulation)
Real-Time Animation

- for example, games
- rendering limited, modeling even more limited
- “traditional” technique - replay scripted motions
  - but scalability/realism are becoming a problem
  - need to generate more new motion on the fly
Traditional CG Animation

- Grew out of traditional animation
- [Pixar]
- every detail of every model is parameterized
  - e.g. position and orientation of base of lamp, joint angles, lengths, light intensity, control points for spline curve of power cord, …
- associate a “motion curve” with each parameter - how it changes in time
- animating == designing motion curves
Motion Curves

- **keyframe approach:**
  - artist sets extreme values at important frames
  - computer fills in the rest with splines
  - artist adjusts spline controls, slopes, adds more points, adjusts, readjusts, re-readjusts, …

- **straight-ahead approach:**
  - artist simply sets parameters in each successive frame

- **layering approach:**
  - design the basic motion curves first, layer detail on afterwards
Motion Curve Tools

- retiming: keep the shape of the trajectory, but change how fast we go along it
  - add a new abstract motion curve controlling distance traveled along trajectory
- Inverse Kinematics (IK):
  - given a skeleton (specified by joint angles)
  - artist directly controls where parts of the skeleton go, computer solves for the angles that achieve that
Procedural Modeling

- write programs to automatically generate models and motion
- for example, “flocking behaviour”
- build a flock of birds by specifying simple rules of motion:
  - accelerate to avoid collisions
  - accelerate to fly at preferred distance to nearby birds
  - accelerate to fly at same velocity as nearby birds
  - accelerate to follow “migratory” impulse
- let it go, hope the results look good
Data-Driven Modeling

- measure the real world, use that data to synthesize models
  - laser scanners
  - camera systems for measuring reflectance properties
  - Image-Based Rendering - e.g. Spiderman
  - …
Data-Driven Motion

- record real motion (motion capture = mocap)
- then play it back
- but life is never that simple
  - real motion is hard to measure
  - measurements are noisy
  - won’t quite fit what you needed
  - not obviously adaptable to new environments, interactive control, etc.
Marker-Based Mocap

- stick performer in a tight black suit, stick markers on body, limbs, …
- film motion with an infrared strobe light and multiple calibrated cameras
- reconstruct 3D trajectories of markers, filling in gaps and eliminating noise
- infer motion of abstract skeleton
- clean up data
- drive CG skeleton with recorded motion curves
What it looks like…

(from Zoran Popovic’s website)
Footskate and Clean Up

- most common problem: footskate
  - feet that in reality were stuck to floor hover and slip around
- fix using IK: determine target footplants, automatically adjust joint angles to keep feet planted
  - often OK to even adjust limb lengths...
Motion Control

- how do you adapt mocap data to new purposes?
  - motion graphs (remixing)
  - motion parameterization (adjust mocap data)
  - motion texturing (add mocap details to traditional animation)
Motion Graphs

- chop up recorded data into tiny clips
  - aim to cut at common poses
- build graph on clips: connect two clips if the end pose of one is similar to the start pose of another
- then walk the graph
  - figure out smooth transitions from clip to clip
  - navigate a small finite graph instead of infinite space of all possible motions
Physics-based modeling

- like procedural modeling, only based on laws of physics
- if you want realistic motion, simulate reality
- human motion:
  - specify muscle forces (joint torques), simulate actual motion
  - has to conserve momentum etc.
  - can handle the unexpected (e.g. a tackle)
  - but need to write motion controllers
- passive motion:
  - figure out physical laws behind natural phenomena
  - simulate (close cousin of scientific computing)
Advanced Rendering
Reading

- FCG Chapter 9: Ray Tracing
  - only 9.1-9.7
- FCG Chap 22: Image-Based Rendering
Errata

- p 155
  - line 1: \( p(t) = e + td \), not \( p(t) = o + td \)
  - equation 5: 2\textsuperscript{nd} term \( 2d^* (e-c) \), not \( 2d^* (o-e) \)

- p 157
  - matrices: \( c_x \rightarrow x_c, c_y \rightarrow y_c, c_z \rightarrow z_c \)

- p 162
  - \( r = d - 2(d.n)n \), not \( r = d + 2(d.n)n \)

- p 163
  - eqn 4 last term: \( n \cos \theta \) not \( n \cos \theta' \)
  - eqn 5: no \( \theta \) term at end
Global Illumination Models

- simple shading methods simulate local illumination models
  - no object-object interaction
- global illumination models
  - more realism, more computation
- approaches
  - ray tracing
  - subsurface scattering
  - radiosity
Simple Ray Tracing

- view dependent method
  - cast a ray from viewer’s eye through each pixel
  - compute intersection of ray with first object in scene
  - cast ray from intersection point on object to light sources

projection reference point
pixel positions on projection plane
Recursive Ray Tracing

- ray tracing can handle
  - reflection (chrome)
  - refraction (glass)
  - shadows
- spawn secondary rays
  - reflection, refraction
    - if another object is hit, recurse to find its color
  - shadow
    - cast ray from intersection point to light source, check if intersects another object
Reflection

- mirror effects
  - perfect specular reflection
Refraction

- Happens at interface between transparent object and surrounding medium
  - e.g. glass/air boundary

- Snell’s Law
  - \( c_1 \sin \theta_1 = c_2 \sin \theta_2 \)
  - Light ray bends based on refractive indices \( c_1, c_2 \)
Total Internal Reflection

As the angle of incidence increases from 0 to greater angles ...

...the refracted ray becomes dimmer (there is less refraction)
...the reflected ray becomes brighter (there is more reflection)
...the angle of refraction approaches 90 degrees until finally a refracted ray can no longer be seen.

http://www.physicsclassroom.com/Class/refrn/U14L3b.html
**Basic Ray Tracing Algorithm**

\[
\textbf{RayTrace}(r, \text{scene}) \\
\text{obj} := \textbf{FirstIntersection}(r, \text{scene}) \\
\text{if (no obj) return BackgroundColor;} \\
\text{else begin} \\
\quad \text{if ( Reflect(obj) ) then} \\
\quad \quad \text{reflect\_color} := \textbf{RayTrace}(\text{ReflectRay}(r, \text{obj})); \\
\quad \text{else} \\
\quad \quad \text{reflect\_color} := \text{Black;} \\
\quad \text{if ( Transparent(obj) ) then} \\
\quad \quad \text{refract\_color} := \textbf{RayTrace}(\text{RefractRay}(r, \text{obj})); \\
\quad \text{else} \\
\quad \quad \text{refract\_color} := \text{Black;} \\
\quad \text{return Shade(reflect\_color, refract\_color, obj);} \\
\text{end;}
\]
Algorithm Termination Criteria

- termination criteria
  - no intersection
  - reach maximal depth
    - number of bounces
  - contribution of secondary ray attenuated below threshold
    - each reflection/refraction attenuates ray
Ray - Object Intersections

- inner loop of ray-tracing
  - must be extremely efficient
- solve a set of equations
  - ray-sphere
  - ray-triangle
  - ray-polygon
Ray - Sphere Intersection

- **ray:** \( x(t) = p_x + v_x t, \quad y(t) = p_y + v_y t, \quad z(t) = p_z + v_z t \)

- **unit sphere:** \( x^2 + y^2 + z^2 = 1 \)

- **quadratic equation in t:**

\[
0 = (p_x + v_x t)^2 + (p_y + v_y t)^2 + (p_z + v_z t)^2 - 1 \\
= t^2 (v_x^2 + v_y^2 + v_z^2) + 2t(p_x v_x + p_y v_y + p_z v_z) \\
+ (p_x^2 + p_y^2 + p_z^2) - 1
\]
Optimized Ray-Tracing

- basic algorithm simple but very expensive
- optimize by reducing:
  - number of rays traced
  - number of ray-object intersection calculations
- methods
  - bounding volumes: boxes, spheres
  - spatial subdivision
    - uniform
    - BSP trees
- (not required reading)
Subsurface Scattering: Translucency

- light enters and leaves at *different* locations on the surface
  - bounces around inside
- technical Academy Award, 2003
  - Jensen, Marschner, Hanrahan
Subsurface Scattering: Marble
Subsurface Scattering: Milk vs. Paint
Subsurface Scattering: Faces
Subsurface Scattering: Faces
Radiosity

- radiosity definition
  - rate at which energy emitted or reflected by a surface
- radiosity methods
  - capture diffuse-diffuse bouncing of light
    - indirect effects difficult to handle with raytracing
Radiosity

- recall radiative heat transfer
- conserve light energy in a volume
- model light transport until convergence
- solution captures diffuse-diffuse bouncing of light

- view independent technique
  - calculate solution for entire scene offline
  - browse from any viewpoint in realtime
Radiosity

- divide surfaces into small patches
- loop: check for light exchange between all pairs
  - form factor: orientation of one patch wrt other patch (n x n matrix)
Raytracing vs. Radiosity Comparison

- ray-tracing: great specular, approx. diffuse
  - view dependent
- radiosity: great diffuse, specular ignored
  - view independent, mostly-enclosed volumes
- advanced hybrids: combine them

raytraced

radiosity
Image-Based Rendering

- store and access only pixels
  - no geometry, no light simulation, ...
  - input: set of images
  - output: image from new viewpoint
    - surprisingly large set of possible new viewpoints
IBR Characteristics

- display time not tied to scene complexity
  - expensive rendering or real photographs
- massive compression possible (120:1)
- can point camera in or out
  - QuickTimeVR: camera rotates, no translation
Characterizing Light

- 7D plenoptic function: $P(x, y, z, \theta, \phi, \lambda, t)$
  - $(x,y,z)$: every position in space
  - $(\theta, \phi)$: every angle
  - $\lambda$: every wavelength of light
  - $t$: every time

- can simplify to 4D function
  - fix time: static scene
  - fix wavelength: static lighting
  - partially fix position: empty space between camera and object
4D Light Field / Lumigraph

- $P(u,v,s,t)$
  - images: just one kind of 2D slice
Non-Photorealistic Rendering

- look of hand-drawn sketches or paintings

www.red3d.com/cwr/npr/
NPRQuake

www.cs.wisc.edu/graphics/Gallery/NPRQuake
Advanced Rendering

- so many more algorithms, so little class time!
  - Renderman REYES
  - photon mapping
  - and lots more...
Final Review
Final Logistics

- 12:00pm-2:30pm Thu Jun 16 here (MCLD 202)
- notes: both sides 8.5”x11” handwritten page
- calculator OK if you want
- have photo ID face up on desk
- spread out, sit where there is an exam
Reading from OpenGL Red Book

- 1: Introduction to OpenGL
- 2: State Management and Drawing Geometric Objects
- 3: Viewing
- 4: Display Lists
- 6: Lighting
- 9: Texture Mapping
- 12: Selection and Feedback
- 13: Now That You Know
  - only section Object Selection Using the Back Buffer
- Appendix: Basics of GLUT (Aux in v 1.1)
- Appendix: Homogeneous Coordinates and Transformation Matrices
Reading from Shirley: Foundations of CG

- 2: Misc Math
- 3: Raster Algs
  - except for 3.8
- 4: Linear Algebra
  - only 4.1-4.2.5
- 5: Transforms
  - except 5.1.6
- 6: Viewing
- 7: Hidden Surfaces
- 8: Surface Shading
- 9: Ray Tracing
  - only 9.1-9.7
- 10: Texture Mapping
- 11: Graphics Pipeline
  - only 11.1-11.4
- 12: Data Structures
  - only 12.3
- 13: Curves and Surfaces
- 17: Human Vision
- 18: Color
  - only 18.1-18.8
- 22: Image-Based Rendering
- 23: Visualization
Studying Advice

- do problems!
  - work through old homeworks, exams
Midterm Topics Covered

- rendering pipeline
- projective rendering pipeline
  - coordinate systems
- transformations
- viewing
- projections
Review: Rendering Pipeline

- pros and cons of pipeline approach
Review: Projective Rendering Pipeline

OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

glVertex3f(x,y,z)
going to
modeling
transformation

glTranslatef(x,y,z)
going to
viewing
transformation

glRotatef(th,x,y,z)
going to
projection
transformation

gluLookAt(…)
going to
perspective
division

glFrustum(…)
going to
viewport
transformation

glutInitWindowSize(w,h)

glViewport(x,y,a,b)

OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system
Review: Transformations, Homog. Coords

\[
\begin{align*}
\text{translate}(a,b,c) & : \\
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} & = \\
\begin{bmatrix}
1 & a & x \\
1 & b & y \\
1 & c & z \\
1 & 1 & 1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{scale}(a,b,c) & : \\
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} & = \\
\begin{bmatrix}
a & x \\
b & y \\
c & z \\
1 & 1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{Rotate}(x, \theta) & : \\
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} & = \\
\begin{bmatrix}
1 & \cos \theta & -\sin \theta & x \\
\cos \theta & -\sin \theta & \cos \theta & y \\
\sin \theta & \cos \theta & 1 & z \\
1 & 1 & 1 & 1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{Rotate}(y, \theta) & : \\
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} & = \\
\begin{bmatrix}
\cos \theta & \sin \theta & 1 & x \\
-\sin \theta & \cos \theta & 1 & y \\
\cos \theta & -\sin \theta & 1 & z \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{Rotate}(z, \theta) & : \\
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} & = \\
\begin{bmatrix}
\cos \theta & 0 & 0 & x \\
0 & \cos \theta & -\sin \theta & y \\
0 & \sin \theta & \cos \theta & z \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{align*}
\]
Review: Transforming View Volumes

- Perspective view volume
- Orthographic view volume
- NDCS
- VCS
Review: Basic Perspective Projection

\[ \frac{y'}{d} = \frac{y}{z} \rightarrow y' = \frac{y \cdot d}{z} \]

- nonuniform foreshortening
- not affine
Post-Midterm Topics Covered

- rasterization
- interpolation/bary coords
- color
- lighting
- shading
- compositing
- clipping
- curves
- picking
- collision
- textures
- procedural approaches
- sampling
- virtual trackball
- visibility
- scientific visualization
- information visualization
- advanced rendering
- animation
Review: Rasterization

- lines: midpoint algorithm
  - optimized: Bresenham

- polygons
  - flood fill
  - scanline algorithms
  - parity test for general case
Review: Barycentric Coordinates

- weighted combination of vertices

\[
P = \alpha \cdot P_1 + \beta \cdot P_2 + \gamma \cdot P_3
\]
\[
\alpha + \beta + \gamma = 1
\]
\[
0 \leq \alpha, \beta, \gamma \leq 1
\]
Review: Color

- color perception
  - color is combination of stimuli from 3 cones
  - metamer: identically perceived color caused by very different spectra

- simple model: based on RGB triples
- component-wise multiplication of colors
  - \((a_0, a_1, a_2) \times (b_0, b_1, b_2) = (a_0 \times b_0, a_1 \times b_1, a_2 \times b_2)\)
Review: Lighting

- reflection equations

\[ I_{\text{diffuse}} = k_d \cdot I_{\text{light}} \cdot (n \cdot l) \]

\[ I_{\text{specular}} = k_s \cdot I_{\text{light}} \cdot (v \cdot r)^{n_{\text{shiny}}} \]

\[ R = 2 \cdot (N \cdot (N \cdot L)) - L \]

- full Phong lighting model

- combine ambient, diffuse, specular components

\[ I_{\text{total}} = k_s \cdot I_{\text{ambient}} + \sum_{i=1}^{\# \text{lights}} I_i \cdot (k_d \cdot (n \cdot l_i) + k_s \cdot (v \cdot r_i)^{n_{\text{shiny}}}) \]
Review: Shading Models

- **flat shading**
  - compute Phong lighting once for entire polygon

- **Gouraud shading**
  - compute Phong lighting at the vertices and interpolate lighting values across polygon

- **Phong shading**
  - compute averaged vertex normals
  - interpolate normals across polygon and perform Phong lighting across polygon
Review: Compositing

- specify opacity with alpha channel: $(r,g,b,\alpha)$
  - $\alpha=1$: opaque, $\alpha=.5$: translucent, $\alpha=0$: transparent
- A over B
  - $C = \alpha A + (1-\alpha)B$
- premultiplying by alpha
  - $C' = \gamma C$, $B' = \beta B$, $A' = \alpha A$
  - $C' = B' + A' - \alpha B'$
  - $\gamma = \beta + \alpha - \alpha \beta$
Review: Clipping

- Cohen Sutherland lines: combining trivial accepts/rejects
  - trivially accept lines: both endpoints inside all edges
    - outcode test: $OC(p1)== 0 \&\& OC(p2)==0$
  - trivially reject lines: both endpoints outside same edge
    - outcode test: $OC(p1) \& OC(p2))!= 0$ reject
  - otherwise, reduce to trivial: splitting into two segments

- Sutherland-Hodgeman polygons
  - for each viewport edge: clip polygon against edge
  - process input edge list to make output edge list
  - inside or outside status between each vertex pair
Review: Curves

- **Hermite**
  - endpoints and their derivatives

- **Bezier**
  - four control points
  - curve remains within their convex hull
  - subdivision construction

- **continuity**
  - $C^0$: share join point
  - $C^1$: share continuous derivatives
  - $C^2$: share continuous second derivatives

- **B-splines**
  - locality of control point influence
Review: Picking

- manual ray intersection
- bounding extents
- backbuffer coding
- select/hit
Review: Collision Detection

- naive approach very expensive: $O(n^2)$

- collision proxies

- spatial data structures to localize

- temporal sampling, fast moving objects

- responding to collisions
Review: Textures

glTexCoord2d(4, 4);
glVertex3d (x, y, z);

Texture
(4,0)

Object
(0,4)

Mapped Texture
(4,4)

Texture
(0,0)

Object
(0,4)

Mapped Texture

Texture
(1,0)

Object
(0,1)

Mapped Texture
(1,1)
Review: Procedural Approaches

- Perlin noise
  - coherency: smooth not abrupt changes
  - turbulence: multiple feature sizes

- particle systems

- fractal landscapes

- L-systems
Review: Sampling

- Shannon Sampling Theorem
  - continuous signal can be completely recovered from its samples iff sampling rate greater than twice maximum frequency present in signal
  - sample past Nyquist Rate to avoid aliasing
    - twice the highest frequency component in the image’s spectrum

Fig. 14.17 Sampling below the Nyquist rate. (Courtesy of George Wolberg, Columbia University.)
Review: Virtual Trackball Rotation

- correspondence:
  - moving point on plane from \((x, 0, z)\) to \((a, 0, c)\)
  - moving point on ball from \(p_1 = (x, y, z)\) to \(p_2 = (a, b, c)\)

- correspondence:
  - translating mouse from \(p_1\) (mouse down) to \(p_2\) (mouse up)
  - rotating about axis \(n = p_1 \times p_2\) by \(\arccos(\frac{p_1 \cdot p_2}{|p_1| |p_2|})\)
Review: Visibility

- painter’s algorithm
  - back to front, incorrect
- BSP trees
  - build, then traverse
- Warnock’s algorithm
  - subdivide viewport
- Z-buffer
  - depth buffer in addition to framebuffer
- backface culling
  - optimization for closed objects
Review: Scientific Visualization

- volume graphics
- isosurfaces
  - extracting with Marching Cubes
- direct volume rendering
  - transfer functions to classify
Review: Information Visualization

- interactive visual representation of abstract data
  - help human perform some task more effectively
- techniques
  - overview, zoom and filter, details on demand
  - focus+context
  - linked views
  - small multiples
- visual channels
  - preattentive visual popout
  - categorical, ordered, quantitative data types
Review: Animation

- traditional direct specification of motion curves
  - key framing: straight-ahead, layering
  - retiming
  - inverse kinematics
- procedural modeling
  - particle systems
- data-driven modeling
  - motion capture
- physics-based modeling
  - cloth, fluid simulation
Review: Advanced Rendering

- ray tracing
  - reflection, refraction, hard shadows

- subsurface scattering
  - marble, milk

- radiosity
  - diffuse lighting, soft shadows

- image-based rendering
  - store/access only pixels
Other Graphics Courses

- **424: Geometric Modelling**
  - not offered next year
- **426: Computer Animation**
  - will be offered next year
- **514: Image-Based Rendering - Heidrich**
- **526: Algorithmic Animation - van de Panne**
- **533A: Digital Geometry - Sheffer**
- **533B: Animation Physics - Bridson**
- **533C: Information Visualization - Munzner**