University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2008

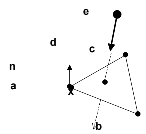
Alla Sheffer

Advanced Rendering
Week 8, Wed Mar 5

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2008

Ray-Triangle Intersection

- method in book is elegant but a bit complex
- easier approach: triangle is just a polygon
  - intersect ray with plane



normal: $\mathbf{n} = (\mathbf{b}-\mathbf{a})\times(\mathbf{c}-\mathbf{a})$

ray: $\mathbf{x} = \mathbf{e} + t\mathbf{d}$

plane: $(\mathbf{p}-\mathbf{x})\cdot\mathbf{n} = 0 \Rightarrow \mathbf{x} = \frac{\mathbf{p}\cdot\mathbf{n}}{\mathbf{n}}$

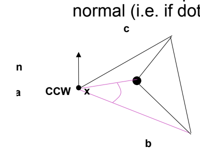
$$\frac{\mathbf{p}\cdot\mathbf{n}}{\mathbf{n}} = \mathbf{e}+t\mathbf{d} \Rightarrow t = -\frac{(\mathbf{e}-\mathbf{p})\cdot\mathbf{n}}{\mathbf{d}\cdot\mathbf{n}}$$

$\mathbf{p}$ is $\mathbf{a}$ or $\mathbf{b}$ or $\mathbf{c}$

  - check if ray inside triangle

Ray-Triangle Intersection

- check if ray inside triangle
  - check if point counterclockwise from each edge (to its left)
  - check if cross product points in same direction as normal (i.e. if dot is positive)



$$(\mathbf{b}-\mathbf{a})\times(\mathbf{x}-\mathbf{a})\cdot\mathbf{n} \geq 0$$
$$(\mathbf{c}-\mathbf{b})\times(\mathbf{x}-\mathbf{b})\cdot\mathbf{n} \geq 0$$
$$(\mathbf{a}-\mathbf{c})\times(\mathbf{x}-\mathbf{c})\cdot\mathbf{n} \geq 0$$

  - more details at http://www.cs.cornell.edu/courses/cs465/2003fa/homeworks/raytri.pdf

Ray Tracing

- issues:
  - generation of rays
  - intersection of rays with geometric primitives
  - geometric transformations
  - lighting and shading
  - efficient data structures so we don't have to test intersection with *every* object

Geometric Transformations

- similar goal as in rendering pipeline:
  - modeling scenes more convenient using different coordinate systems for individual objects
- problem
  - not all object representations are easy to transform
    - problem is fixed in rendering pipeline by restriction to polygons, which are affine invariant
  - ray tracing has different solution
    - ray itself is always affine invariant
    - thus: transform ray into object coordinates!

Geometric Transformations

- ray transformation
  - for intersection test, it is only important that ray is in same coordinate system as object representation
  - transform all rays into object coordinates
    - transform camera point and ray direction by inverse of model/view matrix
  - shading has to be done in world coordinates (where light sources are given)
    - transform object space intersection point to world coordinates
    - thus have to keep both world and object-space ray

Ray Tracing

- issues:
  - generation of rays
  - intersection of rays with geometric primitives
  - geometric transformations
  - lighting and shading
  - efficient data structures so we don't have to test intersection with *every* object

Local Lighting

- local surface information (normal…)
  - for implicit surfaces $F(x,y,z)=0$: normal $\mathbf{n}(x,y,z)$ can be easily computed at every intersection point using the gradient

$$\mathbf{n}(x,y,z) = \begin{pmatrix} \partial F(x,y,z)/\partial x \\ \partial F(x,y,z)/\partial y \\ \partial F(x,y,z)/\partial z \end{pmatrix}$$

  - example: $F(x,y,z) = x^2 + y^2 + z^2 - r^2$

$$\mathbf{n}(x,y,z) = \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix}$$

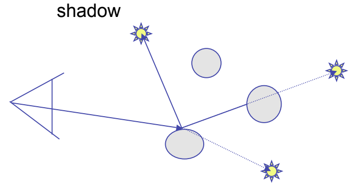
needs to be normalized!

Local Lighting

- local surface information
  - alternatively: can interpolate per-vertex information for triangles/meshes as in rendering pipeline
    - now easy to use Phong shading!
      - as discussed for rendering pipeline
  - difference with rendering pipeline:
    - interpolation cannot be done incrementally
    - have to compute barycentric coordinates for every intersection point (e.g plane equation for triangles)

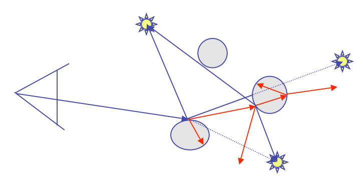
Global Shadows

- approach
  - to test whether point is in shadow, send out **shadow rays** to all light sources
    - if ray hits another object, the point lies in shadow



Global Reflections/Refractions

- approach
  - send rays out in reflected and refracted direction to gather incoming light
  - that light is multiplied by local surface color and added to result of local shading

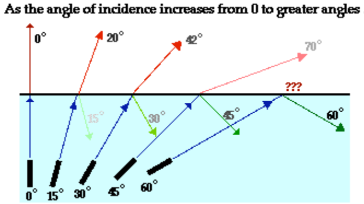


Advanced Phenomena

- Can (not allways efficiently) simulate
  - Soft Shadows
  - Fog
  - Frequency Dependent Light (diamonds & prisms)
  - Barely handle S*DS*
    - S – Specular
    - D - diffuse

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.

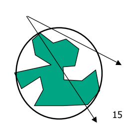
http://www.physicsclassroom.com/Class/refrn/U14L3b.html

Ray Tracing

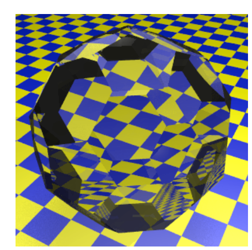
- issues:
  - generation of rays
  - intersection of rays with geometric primitives
  - geometric transformations
  - lighting and shading
  - efficient data structures so we don't have to test intersection with *every* object

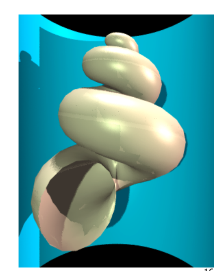
Optimized Ray-Tracing

- 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
- (more on this later with collision)



Example Images



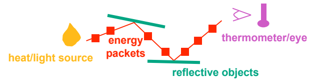


Radiosity

- 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

- illumination as radiative heat transfer



- conserve light energy in a volume
- model light transport as packet flow 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

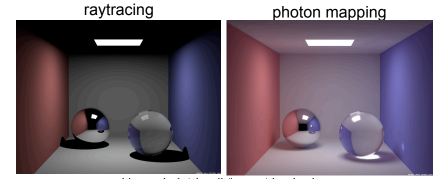
- divide surfaces into small patches
- loop: check for light exchange between all pairs
  - form factor: orientation of one patch wrt other patch (n x n matrix)

escience.anu.edu.au/lecture/cg/GlobalIllumination/Image/discrete.jpg escience.anu.edu.au/lecture/cg/GlobalIllumination/Image/continuous.jpg

Better Global Illumination

- ray-tracing: great specular, approx. diffuse
  - view dependent
- radiosity: great diffuse, specular ignored
  - view independent, mostly-enclosed volumes
- photon mapping: superset of raytracing and radiosity
  - view dependent, handles both diffuse and specular well

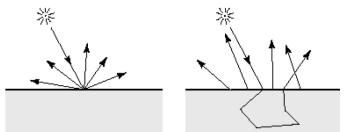
raytracing photon mapping



graphics.ucsd.edu/~henrik/images/cbox.html

Subsurface Scattering: Translucency

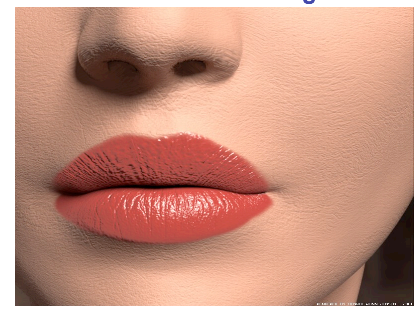
- light enters and leaves at *different* locations on the surface
  - bounces around inside
- technical Academy Award, 2003
  - Jensen, Marschner, Hanrahan



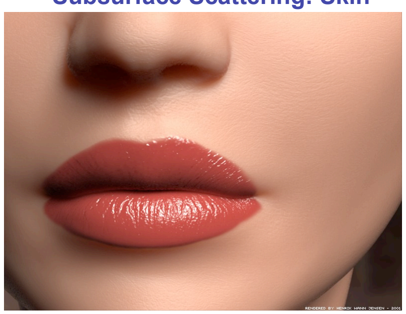
Subsurface Scattering: Marble

Subsurface Scattering: Milk vs. Paint

Subsurface Scattering: Skin

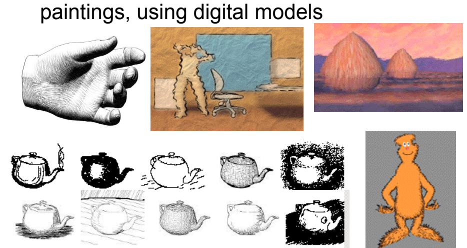


Subsurface Scattering: Skin



Non-Photorealistic Rendering

- simulate look of hand-drawn sketches or paintings, using digital models

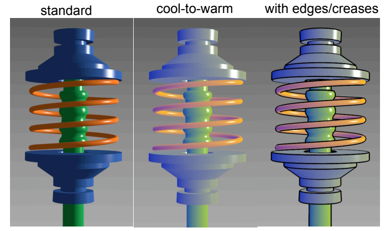


www.red3d.com/cwr/npr/

Non-Photorealistic Shading

- cool-to-warm shading

standard cool-to-warm with edges/creases

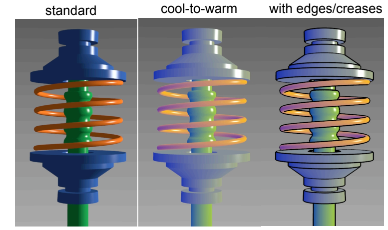


http://www.cs.utah.edu/~gooch/SIG98/paper/drawing.html

Non-Photorealistic Shading

- draw silhouettes: if $(\mathbf{e}\cdot\mathbf{n}_0)(\mathbf{e}\cdot\mathbf{n}_1)\le 0$, **e**=edge-eye vector
- draw creases: if $(\mathbf{n}_0\cdot\mathbf{n}_1)\le threshold$

standard cool-to-warm with edges/creases



http://www.cs.utah.edu/~gooch/SIG98/paper/drawing.html

Image-Based Modelling and 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
    - interpolation allows translation, not just rotation
      - lightfield, lumigraph: translate outside convex hull of object
      - QuickTimeVR: camera rotates, no translation
    - can point camera in or out