Ray Tracing

Last Time?

- Keyframing
- Procedural Animation
- Physically-Based Animation
- Forward and Inverse Kinematics
- Motion Capture

Two solutions
Readings for Friday 3/6: (pick one)


Readings for Friday 3/6: (pick one)

“Articulated Swimming Creatures” Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu, SIGGRAPH 2011

Figure 8: A five-link eel swims in a 2D fluid environment. In contrast to the simulation in 3D, an eel swimming in 2D fluid sheds only one single vortex street. Red traces show the counter-clockwise vortices while blue traces show the clockwise vortices.

http://www.cc.gatech.edu/~jtan34/project/articulatedSwimmingCreatures.html
Paper comments/questions

- Well written, intuitive, clear, “transparent” (what language was used for implementation)
- Could be used to discover optimal techniques for sports
- Swimming patterns of unobservable things (deep sea fish, or oceanic paleontology)
- Extend to schooling fish?

Readings for Friday 3/6: (pick one)

"Flexible Muscle-Based Locomotion for Bipedal Creatures", Geijtenbeek, van de Panne, and van der Stappen, SIGGRAPH 2013
Paper comments/questions

• Simulated characters start with a lot of knowledge about walking already (stages of gait, target orientations). It would be near impossible for the creature to learn to walk from scratch.
• Optimized muscle positions
• Different target speeds have different optimal gaits
• Stages of walking gait – only 4 states (surprisingly few)
• How does this extend to organisms with more legs?
• How does this extend to obstacle avoidance?

Today

• Ray Casting
  – Ray-Plane Intersection
  – Ray-Sphere Intersection
  – Point in Polygon
• Ray Tracing
• Recursive Ray Tracing
• Distribution Ray Tracing
Durer’s Ray Casting Machine

- Albrecht Durer, 16th century

Ray Casting

For every pixel
  Construct a ray from the eye
  For every object in the scene
    Find intersection with the ray
    Keep if closest
  Shade depending on light and normal vector

Finding the intersection and normal is the central part of ray casting
A Note on *Local* Shading

- Surface/Scene Characteristics:
  - surface normal
  - direction to light
  - viewpoint
- Material Properties
  - color/texture
  - diffuse (matte)
  - specular (shiny)
  - …
- More later!

Ray Representation?

- Two vectors:
  - Origin
  - Direction (normalized is better)
- Parametric line (*explicit* representation)
  - \( P(t) = \text{origin} + t \cdot \text{direction} \)
3D Plane Representation?

- Plane defined by
  - \( P_o = (x,y,z) \)
  - \( n = (A,B,C) \)

- *Implicit* plane equation
  - \( H(P) = Ax + By + Cz + D = 0 \)
  - \( n \cdot P + D = 0 \)

- Point-Plane distance?
  - If \( n \) is normalized,
    distance to plane, \( d = H(P) \)
  - \( d \) is the *signed distance*

Explicit vs. Implicit?

- Ray equation is explicit
  - \( P(t) = R_o + t \cdot R_d \)
  - Parametric
  - Generates points
  - Harder to verify that a point is on the ray

- Plane equation is implicit
  - \( H(P) = n \cdot P + D = 0 \)
  - Solution of an equation
  - Does not generate points
  - Verifies that a point is on the plane
Ray-Plane Intersection

- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t

\[ P(t) = R_o + t \cdot R_d \]
\[ H(P) = n \cdot P + D = 0 \]
\[ n \cdot (R_o + t \cdot R_d) + D = 0 \]
\[ t = -\frac{(D + n \cdot R_o)}{n \cdot R_d} \]

Additional Housekeeping

- Verify that intersection is closer than previous
  \[ P(t) < t_{current} \]
- Verify that it is not out of range (behind eye)
  \[ P(t) > t_{min} \]
**Normal**

- Needed for shading
  - diffuse: dot product between light and normal
- Normal of a plane is constant!

![Diagram of normal](image)

**Ray-Triangle Intersection**

- Use barycentric coordinates:
  - \( P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c \)
  - with \( \alpha + \beta + \gamma = 1 \)
- If \( 0 < \alpha < 1 \) & \( 0 < \beta < 1 \) & \( 0 < \gamma < 1 \)
  then the point is inside the triangle!

![Diagram of ray-triangle intersection](image)
How Do We Compute $\alpha$, $\beta$, $\gamma$?

- Ratio of opposite sub-triangle area to total area
  - $\alpha = A_a/A$  $\beta = A_b/A$  $\gamma = A_c/A$
- Use signed areas for points outside the triangle

But how do I know if the point is outside the triangle?  
That's what I was trying to determine!

Using Cramer’s Rule…

- Used to solve for one variable at a time in system of equations

$$
\begin{align*}
\beta &= \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \\
\gamma &= \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|} \\
t &= \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}
\end{align*}
$$

| denotes the determinant

Can be copied mechanically into code
Sphere Representation?

- Implicit sphere equation
  - Assume centered at origin (easy to translate)
  - \( H(P) = P \cdot P - r^2 = 0 \)

Ray-Sphere Intersection

- Insert explicit equation of ray into implicit equation of sphere & solve for \( t \)

\[
P(t) = R_o + t R_d \quad H(P) = P \cdot P - r^2 = 0
\]

\[
(R_o + t R_d) \cdot (R_o + t R_d) - r^2 = 0
\]

\[
R_d \cdot R_d t^2 + 2 R_d \cdot R_o t + R_o \cdot R_o - r^2 = 0
\]
Ray-Sphere Intersection

- Quadratic: \(at^2 + bt + c = 0\)
  - \(a = 1\) (remember, \(||R_d|| = 1||\)
  - \(b = 2R_d \cdot R_o\)
  - \(c = R_o \cdot R_o - r^2\)

- with discriminant \(d = \sqrt{b^2 - 4ac}\)

- and solutions \(t_{\pm} = \frac{-b \pm d}{2a}\)

- What does it mean if there are no solutions, 1 solution, or 2 solutions?

Questions?

- depth
- normals
- local shading
Reading for Next Friday


Today

- Ray Casting
- Ray Tracing
  - Shadows
  - Reflection
  - Refraction
- Recursive Ray Tracing
- Distribution Ray Tracing
How Can We Add Shadows?

Find the point to be shaded
For every light,
  Construct ray from point to light
For every object
  find intersection of ray with object
If no objects between point and light
  Add contribution from light

Mirror Reflection

- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient (color)
Reflection

• Reflection angle = view angle
• \( \mathbf{R} = \mathbf{V} - 2 (\mathbf{V} \cdot \mathbf{N}) \mathbf{N} \)

Transparency

• Cast ray in refracted direction
• Multiply by transparency coefficient (color)
Qualitative Refraction

From “Color and Light in Nature” by Lynch and Livingston

Refraction

\[ I = N \cos \theta_i - M \sin \theta_i \]
\[ M = \frac{(N \cos \theta_i - I)}{ \sin \theta_i} \]
\[ T = -N \cos \theta_T + M \sin \theta_T \]
\[ = -N \cos \theta_T + (N \cos \theta_i - I) \sin \theta_T / \sin \theta_i \]
\[ = -N \cos \theta_T + (N \cos \theta_i - I) \eta_r \]
\[ = [\eta_r \cos \theta_i - \cos \theta_T] N - \eta_r I \]
\[ = [\eta_r \cos \theta_i - \sqrt{1 - \eta_r^2 \sin^2 \theta_i}] N - \eta_r I \]
\[ = [\eta_r \cos \theta_i - \sqrt{1 - \eta_r^2 (1 - \cos^2 \theta_i)}] N - \eta_r I \]
\[ = [\eta_r (N \cdot I) - \sqrt{1 - \eta_r^2 (1 - (N \cdot I)^2)}] N - \eta_r I \]

Snell-Descartes Law:
\[ \eta_r \sin \theta_i = \eta_T \sin \theta_T \]
\[ \eta_r \sin \theta_T^2 = \eta_T \sin \theta_i^2 \]
\[ \frac{\sin \theta_T}{\sin \theta_i} = \frac{\eta_r}{\eta_T} \]

- Total internal reflection when the square root is imaginary
- Don’t forget to normalize!
Refraction & the Sidedness of Objects

- Make sure you know whether you’re entering or leaving the transmissive material:

\[ n_T = n_i \]

- What about intersecting transparent objects?

Total Internal Reflection

From “Color and Light in Nature” by Lynch and Livingston
Questions?

Today

- Ray Casting
- Ray Tracing
- Recursive Ray Tracing
- Distribution Ray Tracing
Ray Tracing

*trace ray*
Intersect all objects
For every light
  *cast shadow ray*
  color += local shading term
If mirror
  color += color_{refl} * trace reflected ray
If transparent
  color += color_{trans} * trace transmitted ray

*Does it ever end?*

Stopping criteria:
- Recursion depth
  - Stop after a number of bounces
- Ray contribution
  - Stop if reflected / transmitted contribution becomes too small

The Ray Tree

N_i surface normal
R_i reflected ray
L_i shadow ray
T_i transmitted (refracted) ray

Complexity?
Ray Debugging

• Visualize the ray tree for single image pixel

Other Reading for Next Friday

Today

- Ray Casting
- Ray Tracing
- Recursive Ray Tracing
- Distribution Ray Tracing
  - Soft shadows
  - Antialiasing (getting rid of jaggies)
  - Glossy reflection
  - Motion blur
  - Depth of field (focus)

Shadows

- one shadow ray per intersection per point light source
Shadows & Light Sources

http://www.pa.uky.edu/~sciworks/light/preview/bulb2.htm

http://www.davidfay.com/index.php


Soft Shadows

- multiple shadow rays to sample area light source

area light source

one shadow ray

lots of shadow rays
Antialiasing – Supersampling

- multiple rays per pixel

<table>
<thead>
<tr>
<th>Point Light</th>
<th>Area Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>jaggies</td>
<td>w/ antialiasing</td>
</tr>
</tbody>
</table>

Reflection

- one reflection ray per intersection

perfect mirror
**Glossy Reflection**

- multiple reflection rays

**Motion Blur**

- Sample objects temporally
Depth of Field

- multiple rays per pixel

Ray Tracing Algorithm Analysis

- Ray casting
- Lots of primitives
- Recursive
- Distributed Ray Tracing Effects
  - Soft shadows
  - Anti-aliasing
  - Glossy reflection
  - Motion blur
  - Depth of field

\[
\text{cost} \approx \text{height} \times \text{width} \times \text{num primitives} \times \text{intersection cost} \times \text{size of recursive ray tree} \times \text{num shadow rays} \times \text{num supersamples} \times \text{num glossy rays} \times \text{num temporal samples} \times \text{num focal samples} \times \ldots
\]

can we reduce this?

does this serve double duty?
HW3: Raytracing & Epsilon

Solution: advance the ray start position \( \epsilon \) distance along the ray direction OR ignore all intersections \( < \epsilon \) (rather than \( < 0 \))

What’s a good value for \( \epsilon \)? Depends on hardware precision & scene dimensions