Visible Surface Ray Tracing

- Determine visibility of a surface by tracing rays of light from the viewer’s eye to the objects in the scene
- Image precision algorithm
- VSRT is sometimes called “Ray Casting”

Problem: computing intersections between the ray and the objects

Each point \((x,y,x)\) of the ray from \((x_0,y_0,z_0)\) to \((x_1,y_1,z_1)\) is defined by:

\[
x = x_0 + t\Delta x, \quad y = y_0 + t\Delta y, \quad z = z_0 + t\Delta z
\]

where:

\[
\Delta x = x_1 - x_0, \quad \Delta y = y_1 - y_0, \quad \Delta z = z_1 - z_0
\]
Visible Surface Ray Tracing

• Example:

Ray: \[ x = x_0 + tΔx, \quad y = y_0 + tΔy, \quad z = z_0 + tΔz \]

Sphere: \[ (x - a)^2 + (y - b)^2 + (z - c)^2 = r^2 \]

\[
\begin{align*}
\Delta x^2 - 2ax + a^2 + y^2 - 2by + b^2 + z^2 - 2cz + c^2 &= r^2, \\
(x_0 + tΔx)^2 - 2a(x_0 + tΔx) + a^2 + (y_0 + tΔy)^2 - 2b(y_0 + tΔy) + b^2 \\
\quad + (z_0 + tΔz)^2 - 2c(z_0 + tΔz) + c^2 &= r^2, \\
\Delta x^2 + 2\Delta xΔy + Δy^2 - 2ax_0 - 2aΔx + a^2 \\
\quad + 2\Delta yΔx - 2by_0 - 2bΔy + b^2 \\
\quad + 2\Delta zΔx - 2c\Delta x - 2c\Delta z + c^2 &= r^2.
\end{align*}
\]

Collecting terms:

\[
(Δx^2 + Δy^2 + Δz^2)t^2 + 2t(ΔxΔy, ΔxΔz + ΔyΔz) + (Δx^2 + Δy^2 + Δz^2) - r^2 = 0,
\]

Quadratic equation in \( t \):

\[
At^2 + Bt + C = 0
\]

Has zero, one or two real roots

\[
t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}
\]

Visible Surface Ray Tracing

• Example:

Zero real roots: no intersections
One root: ray grazes the sphere
Two roots: smallest positive \( t \) is the closest

At the intersection point, the surface has normal:

\[
\frac{x-a}{r}, \quad \frac{y-b}{r}, \quad \frac{z-c}{r}
\]

Similar approach with other quadratic surfaces

• Intersection between a ray and a polygon is harder to find:
  - Find the intersection between the ray and the polygon’s plane;
  - Check whether the intersection lies within the polygon

Visible Surface Ray Tracing

• Similar approach for other quadratic surfaces:
  - Sphere: \( x^2 + y^2 + z^2 - r^2 \)
  - Cylinder: \( x^2 + y^2 - r^2 \)
  - Cone: \( x^2 + y^2 - z^2 \)
  - Paraboloid: \( x^2 + y^2 - z \)
  - Hyperboloid: \( x^2 + y^2 - z^2 \pm r^2 \)

• Roots of equations of degree higher than 2 can be found with an iterative method like Newton

Intersection between a ray and a polygon is harder to find:

- Find the intersection between the ray and the polygon's plane;
- Check whether the intersection lies within the polygon
Visible Surface Ray Tracing

• Example:
  Ray: \[ x = x_0 + t\Delta x, \quad y = y_0 + t\Delta y, \quad z = z_0 + t\Delta z \]
  Plane: \[ Ax + By + Cz + D = 0 \]

If the denominator is 0, the ray and the plane are parallel

Testing whether the intersection lies within the polygon is done with an orthographic projection and 2D tests

Visible Surface Ray Tracing

Efficiency considerations:
• Ray tracing is slow because it intersects every ray with every object
• To make ray tracing faster we can use coherence:
  Image coherence - neighboring pixel, same object,
  Spatial coherence - neighboring points, same object

Visible Surface Ray Tracing

• Problem:
  - Cartman is composed by 100000 polygons
  - Ray tracing computes 100000 ray-polygon intersections
  - Even when the ray misses Cartman

• Solution:
  - Place a sphere around Cartman
  - If ray misses sphere then the ray misses Cartman

Visible Surface Ray Tracing

• Efficiency considerations: optimizing intersection calculations by bounding objects with parametrized slabs

Visible Surface Ray Tracing

• Efficiency considerations: avoiding intersection calculations with bounding volumes hierarchies
Visible Surface Ray Tracing

- **Efficiency considerations**: avoiding intersection calculation with spatial partitioning
- 3-D array of cells
- Each cell contains list of all objects it intersects
- Ray intersected with all objects in a given cell’s list
- Cells visited in Bresenham order

Recursive Ray Tracing

- Extends the basic Ray Tracing algorithm to handle shadows, reflection and refraction
- **Shadows**: fire an additional ray from the intersection to the light source. If this shadow ray intersects any object along the way, then the point is in shadow

Ray Tracing

- **Reflection and Refraction**: secondary reflection and refraction rays are fired at intersections. In turn, these rays may spawn shadow, reflection and refraction rays
  - **N**: Surface normal
  - **L**: Shadow ray
  - **R**: Reflected ray
  - **T**: Transmitted ray

The rays form a ray tree
<table>
<thead>
<tr>
<th>Recursive Ray Tracing</th>
<th>Ray Tracing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> Motion Blur</td>
<td><strong>Software</strong></td>
</tr>
<tr>
<td><img src="image1" alt="Recursive Ray Tracing Example" /></td>
<td><strong>Ray tracer:</strong></td>
</tr>
<tr>
<td><img src="image2" alt="Recursive Ray Tracing Example" /></td>
<td>Persistence Of Vision POV-Ray</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.povray.org/">http://www.povray.org/</a></td>
</tr>
<tr>
<td></td>
<td><strong>Wireframe modeller for POV-Ray:</strong></td>
</tr>
<tr>
<td></td>
<td>MoRay</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.stmuc.com/moray/">http://www.stmuc.com/moray/</a></td>
</tr>
</tbody>
</table>