Recursive Ray Tracing

Design a scene with objects in 3D, a viewpoint, and a viewing plane subdivided into pixels.

For each pixel on view plane, cast a ray from viewpoint thru pixel, see what ray intersects in the scene. If it hits nothing, colour pixel as background.

If ray strikes an object, determine normal at intersection location, compute ambient component. Next, cast rays from intersection point to each light in scene. If ray strikes no object, light can be "seen" at intersection point (not in shadow) so add diffuse plus specular component for given light.

Finally, from point of intersection cast secondary rays in direction of perfect reflection.
(reflected rays) and in direction of transmission if surface of object is transparent (transmissive rays). Secondary rays may, in turn, spawn their own reflected, transmissive rays.

Reflected rays will cause shiny objects to reflect colour of surrounding objects.

Transmissive rays will allow viewer to see through surface of an object. If surface is transparent.

\[ I = I_{0}k_{a} + I_{0}k_{s} \sum_{i} \left( k_{d}(\hat{N} \cdot \hat{L}) + k_{r}(\hat{R} \cdot \hat{V})^{m} \right) + k_{t}I_{t} \]

\[ s = \begin{cases} 0 & \text{if light is cannot be seen} \\ 1 & \text{otherwise} \end{cases} \]
Efficiency

Main consideration is attempting to reduce the number of ray-object intersection calculations. Can be accomplished in a variety of ways.

Item Buffer:

Preprocess scene with less costly visible surface detection algorithm, record at each pixel position ID of closest object, use this object for initial intersection calculation.

Adaptive Depth Control

Many objects in scene are not highly specular, so terminate reflected ray if maximum possible contribution of ray is below some pre-chosen threshold.

\[
\begin{align*}
\vec{E}_2 &\leq 0.05 \\
\vec{E}_1 &\leq 0.1 \\
\vec{E}_2 \max &\leq 0.1 \times 0.05 \\
\vec{E}_1 \max &\leq 0.1 \\
\end{align*}
\]

If max child

\[
\begin{align*}
\vec{E}_1 = \vec{E}_2 \max \\
\vec{E}_2 &= \vec{E}_2 \max \\
\end{align*}
\]

\[
\begin{align*}
\vec{E}_1 \max &= 0.1 \times 0.05 \\
\Rightarrow &\leq 0.005
\end{align*}
\]
Light Buffer

Designed to reduce number of intersection tests performed on shadow rays. Place a "light cube" centered about each light, tessellate faces of cube into regular grid.

For each light buffer "beam" emanating from light through one grid on one face of cube, maintain an ordered list of any surface that intersects light buffer. (can be done by placing viewpoint at light, then scan converting scene onto view plane and tessellating).

So ordered list is A B E F
During shadow processing determine which light buffer square shadow ray passes through, only need to check for intersection with objects in front of intersection point in that light buffer's ordered list.