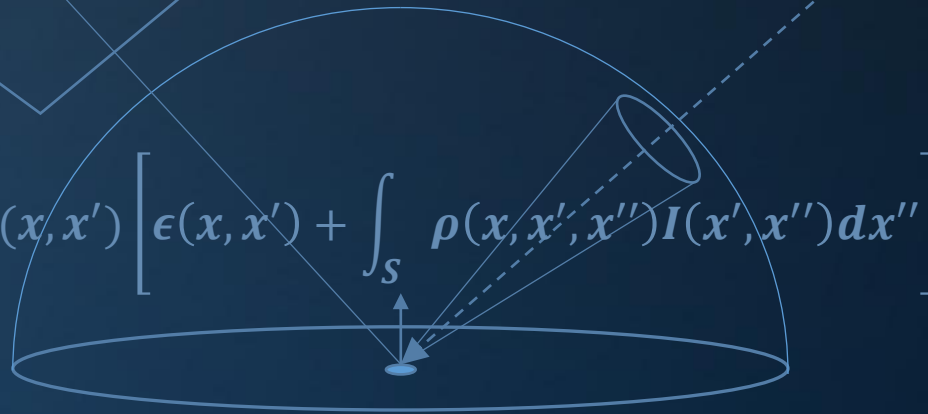


INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 - February 2023

Lecture 2 - “Whitted”

Welcome!



Today's Agenda:

- Introduction: Appel
- Whitted
- Cook



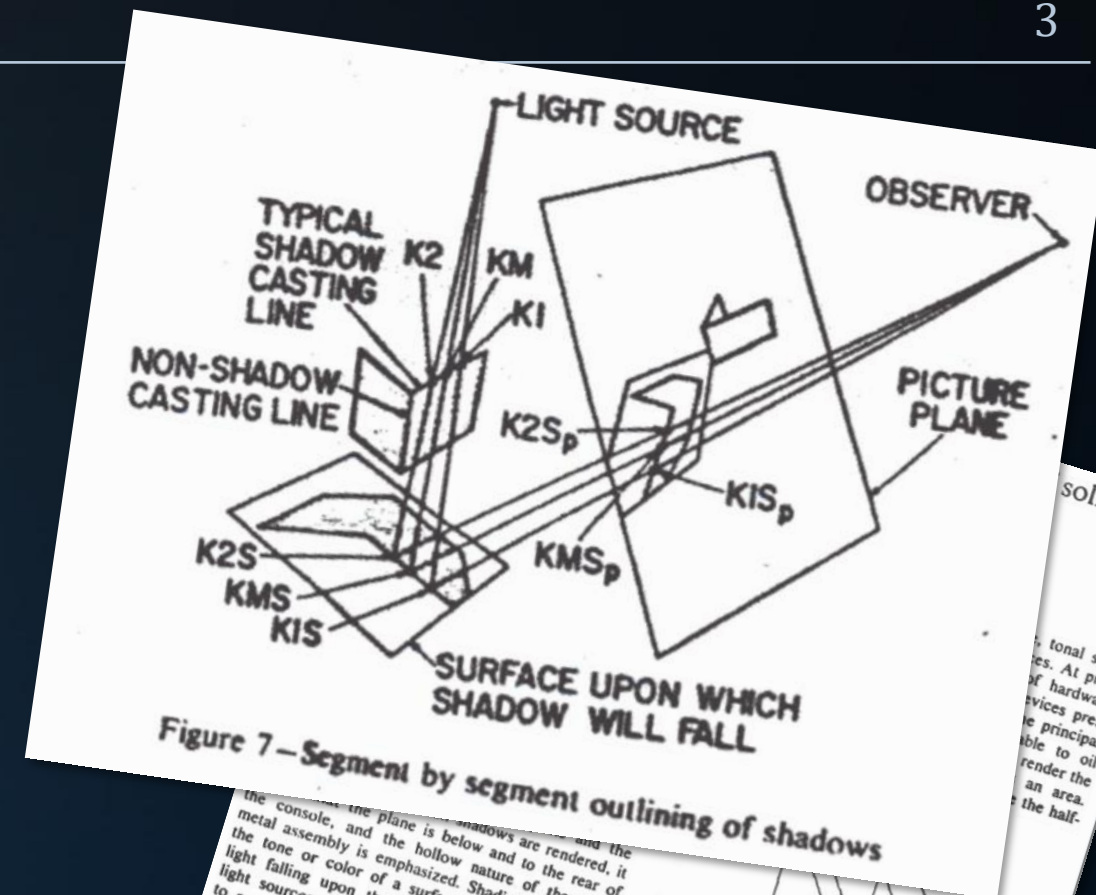
Appel

Some Techniques for Shading Machine Renderings of Solids, Arthur Appel, 1964.

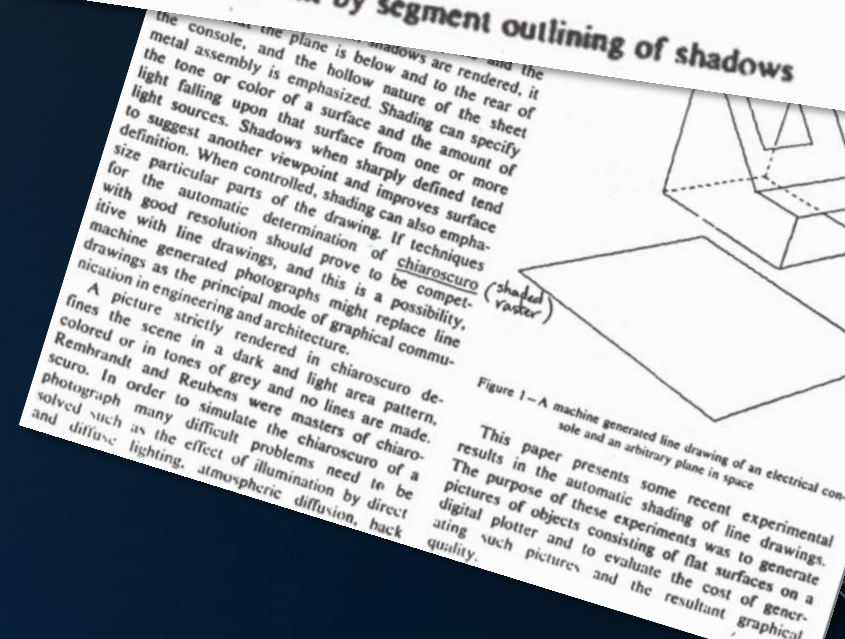
Idea: use rays to find geometry and shadows.

“Graphics” for games in 1964:

https://en.wikipedia.org/wiki/The_Sumerian_Game



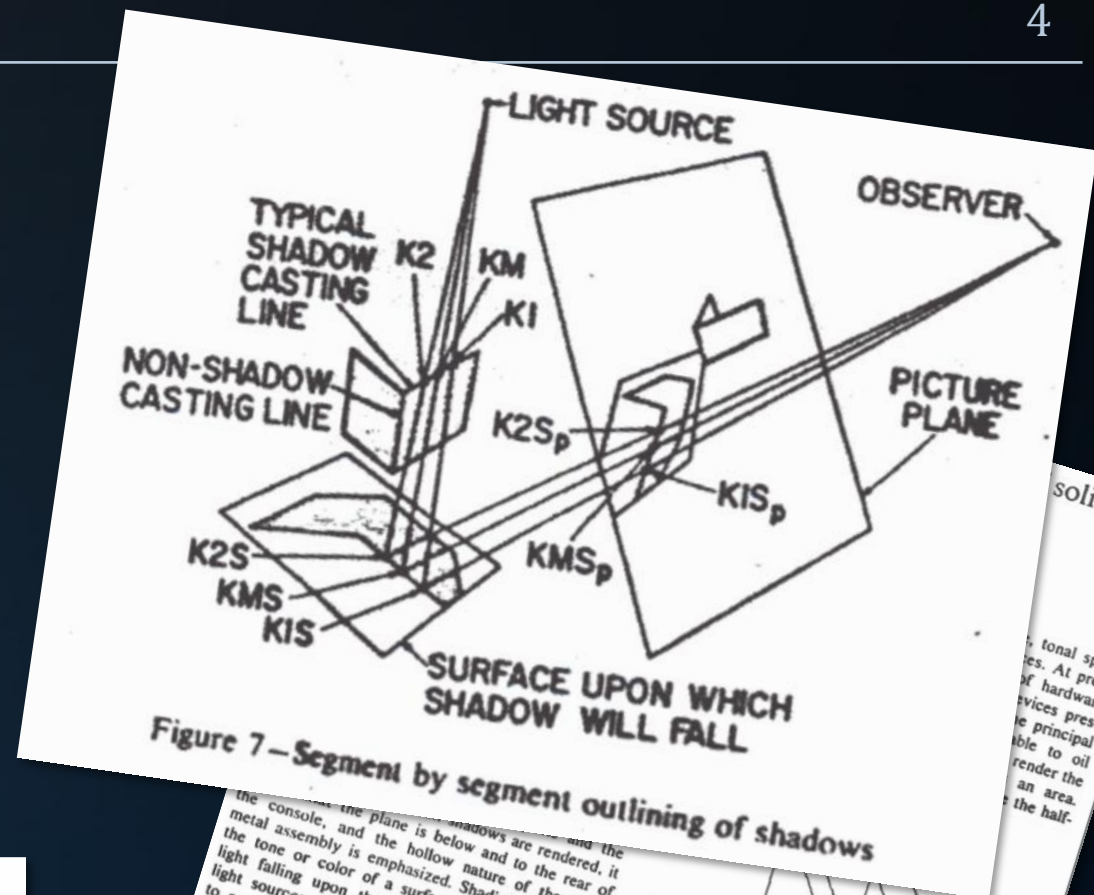
IBM 7090



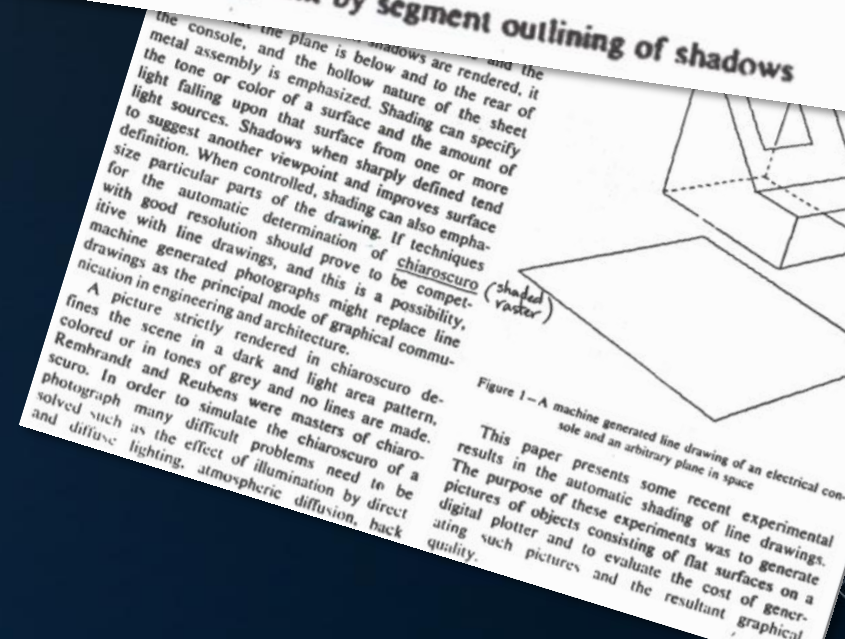
Appel

Some Techniques for Shading Machine Renderings of Solids, Arthur Appel, 1964.

Idea: use rays to find geometry and shadows.



This method is very time consuming, usually requiring for useful results several thousand times as much calculation time as a wire frame drawing. About one half of this time is devoted to determining the



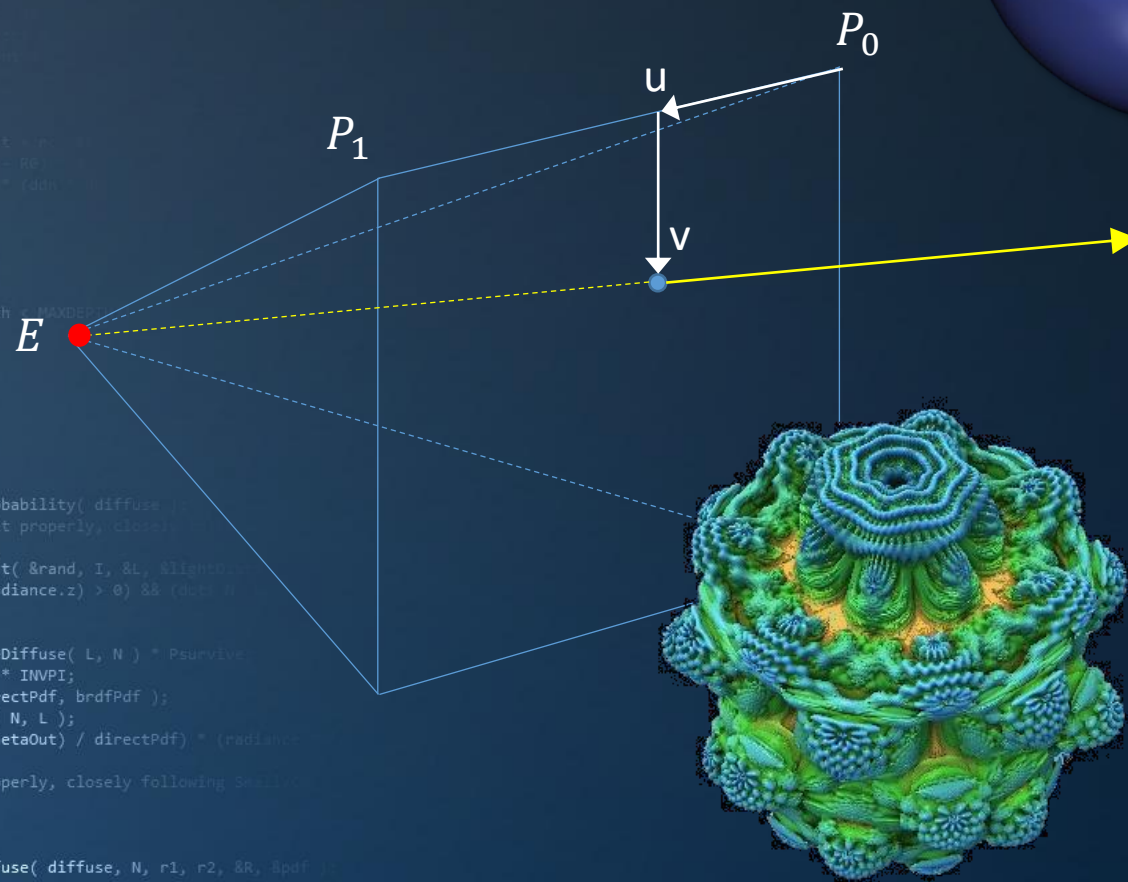
Appel

Recap

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * nc;
        cos2t = 1.0f - nnt * ddn;
        t = sqrt(cos2t);
        D, N );
        P0 = O + tD;
        P1 = O + (1-t)D;
        at a = nt - nc, b = nt * nc;
        Tr = 1 - (R0 + (1 - R0) * cos2t);
        R = (D * nnt - N * (ddn * cos2t));
        E * diffuse;
        = true;
    }
    else
    {
        refl + refr)) && (depth < MAXDEPTH)
        {
            D, N );
            refl * E * diffuse;
            = true;
        }
    }
    MAXDEPTH)
    {
        survive = SurvivalProbability( diffuse );
        estimation - doing it properly, closely following
        if;
        radiance = SampleLight( &rand, I, &L, &align, &pdf );
        e.x + radiance.y + radiance.z > 0) && (depth < MAXDEPTH)
        {
            w = true;
            at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
            at3 factor = diffuse * INVPI;
            at weight = Mis2( directPdf, brdfPdf );
            at cosThetaOut = dot( N, L );
            E * ((weight * cosThetaOut) / directPdf) * (radiance
            random walk - done properly, closely following
            survive)
        }
    }
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;
}

```



$$\text{Plane: } P \cdot \vec{N} + d = 0$$

$$\text{Ray: } P(t) = O + t\vec{D}$$

Substituting for $P(t)$, we get

$$(O + t\vec{D}) \cdot \vec{N} + d = 0$$

$$t = -(O \cdot \vec{N} + d) / (\vec{D} \cdot \vec{N})$$

$$P = O + t\vec{D}$$

$$\text{Sphere: } (P - C) \cdot (P - C) - r^2 = 0$$

Substituting for $P(t)$, we get

$$(O + t\vec{D} - C) \cdot (O + t\vec{D} - C) - r^2 = 0$$

$$\vec{D} \cdot \vec{D} t^2 + 2\vec{D} \cdot (O - C) t + (O - C)^2 - r^2 = 0$$

$$at^2 + bt + c = 0 \rightarrow t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$a = \vec{D} \cdot \vec{D}$$

$$b = 2\vec{D} \cdot (O - C)$$

$$c = (O - C) \cdot (O - C) - r^2$$



Appel

$$\text{Ray: } P(t) = O + t\vec{D}$$

```

ics
& (depth < MAXDEPTH)
{
    if (nt < 0) nt = 0;
    if (nt > 1) nt = 1;
    nt = nt / nc; ddn = ddn * nt;
    rnt = 1.0f - nnt * nnt;
    if (rnt < 0) rnt = 0;
    D = N * (D * nnt - N * (ddn * nnt));
    if (rnt > 0)
    {
        at a = nt - nc, b = nt * nc;
        at Tr = 1 - (R0 + (1 - R0) * rnt);
        Tr) R = (D * nnt - N * (ddn * nnt));
        E * diffuse;
        = true;
    }
    else
    {
        refl + refr)) && (depth < MAXDEPTH)
        {
            D, N );
            refl * E * diffuse;
            = true;
        }
    }
    MAXDEPTH)
    survive = SurvivalProbability( diffuse );
    estimation - doing it properly, closely following Small's
    if;
    radiance = SampleLight( &rand, I, &L, &align, &pdf );
    e.x + radiance.y + radiance.z) > 0) && (depth < MAXDEPTH)
    {
        w = true;
        at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
        at3 factor = diffuse * INVPI;
        at weight = Mis2( directPdf, brdfPdf );
        at cosThetaOut = dot( N, L );
        E * ((weight * cosThetaOut) / directPdf) * (radiance
    }
    random walk - done properly, closely following Small's
    (survive)
    ;
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    ion = true;
}

```



Appel

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1.0f : 0.0f)
    {
        nt = nt / nc, ddn = ddn / nc;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * cos2t);
        Tr) R = (D * nnt - N * (ddn * cos2t));
    }
    E * diffuse;
    = true;
    -
    refl + refr)) && (depth < MAXDEPTH)
    {
        D, N );
        refl * E * diffuse;
        = true;
    }
    MAXDEPTH)
    survive = SurvivalProbability( diffuse );
    estimation - doing it properly, closely following Small's
    if;
    radiance = SampleLight( &rand, I, &L, &align, &pdf );
    e.x + radiance.y + radiance.z) > 0) && (depth < MAXDEPTH)
    {
        w = true;
        at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
        at3 factor = diffuse * INVPI;
        at weight = Mis2( directPdf, brdfPdf );
        at cosThetaOut = dot( N, L );
        E * ((weight * cosThetaOut) / directPdf) * (radiance
    }
    random walk - done properly, closely following Small's
    vive)
    ;
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;
}

```



Today's Agenda:

- Introduction: Appel
- Whitted
- Cook



Whitted

An Improved Illumination Model for Shaded Display

In 1980, “State of the Art” consisted of:

- Rasterization
- Shading: either diffuse ($N \cdot L$) or specular ($(N \cdot H)^n$), both not considering fall-off (Phong)
- Reflection, using environment maps (Blinn & Newell *)
- Stencil shadows (Williams **)

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    }
}

```

```

at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * a);
Tr) R = (D * nnt - N * (ddn *

```

```

E * diffuse;
= true;

```

```

efl + refr)) && (depth < MAXDEPTH)

```

```

D, N );
refl * E * d
= true;

```

```

MAXDEPTH)

```

```

survive = Su
estimation
df;
radiance = S
e.x + radian

```

```

w = true;
at brdfPdf =
at3 factor =
at weight =
at cosTheta0
E * ((weight * cosThetaOut) / directPdf) * (new

```

```

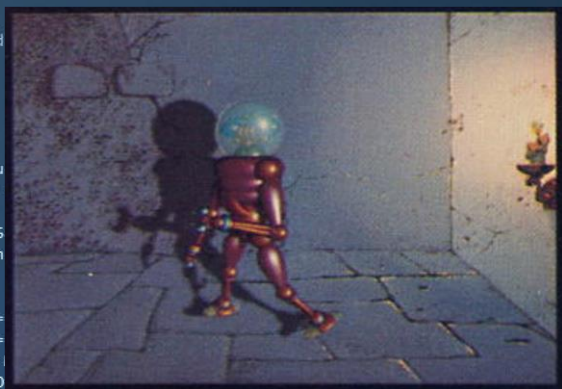
random walk - done properly, closely following
ive)

```

```

at3 brdf = SampleDiffuse( diffuse, N, r1, r2**
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



* : Blinn, J. and Newell, M. 1976. Texture and Reflection in Computer Generated Images.

** : Williams, L. 1978. Casting curved shadows on curved surfaces..



Whitted

An Improved Illumination Model for Shaded Display

In 1980, “State of the Art” consisted of:

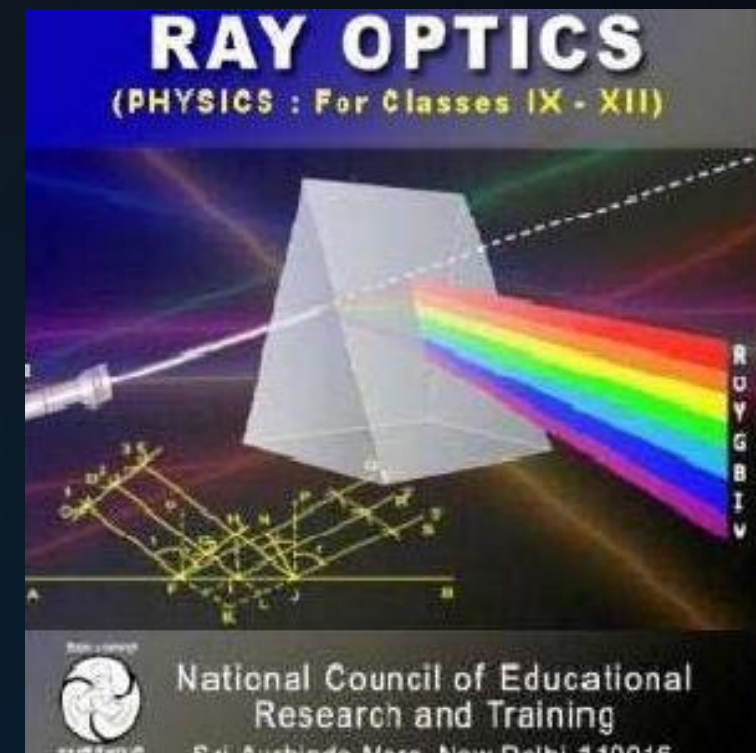
- Rasterization
- Shading: either diffuse ($N \cdot L$) or specular ($(N \cdot H)^n$), both not taking into account fall-off (Phong)
- Reflection, using environment maps (Blinn & Newell)
- Stencil shadows (Williams)

Goal:

- Solve *reflection and refraction*

Improved model:

- Based on classical ray optics



Whitted

An Improved Illumination Model for Shaded Display*

Physical basis of Whitted-style ray tracing:

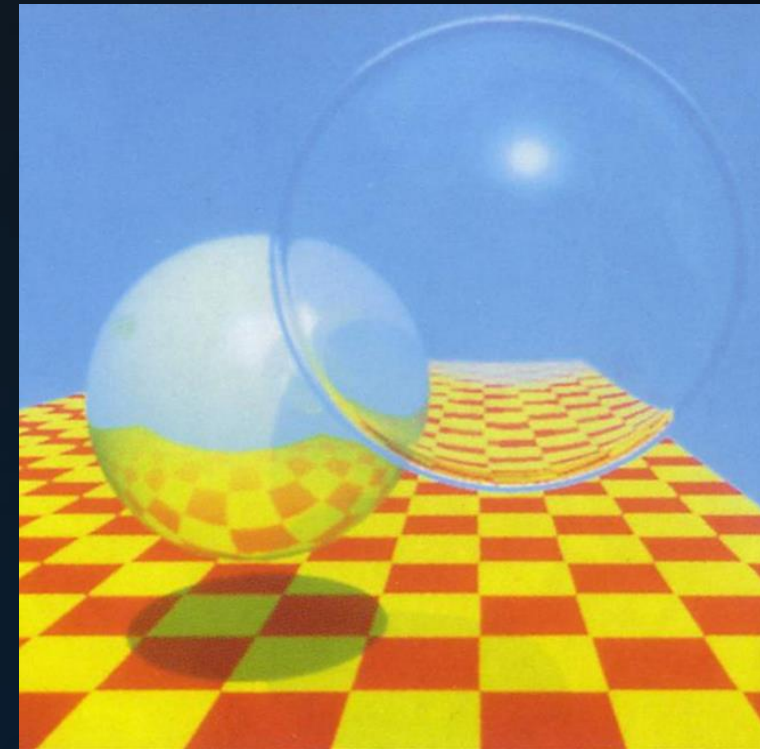
Light paths are generated (backwards) from the camera to the light sources, using rays to simulate optics.

```
Color Trace( ray r )
```

```
    I, N, mat = NearestIntersection( scene, r )
```

```
    return mat.color * DirectIllumination( I, N )
```

* : T. Whitted. An Improved Illumination Model for Shaded Display.
Commun. ACM, 23(6):343–349, 1980.



Whitted

An Improved Illumination Model for Shaded Display

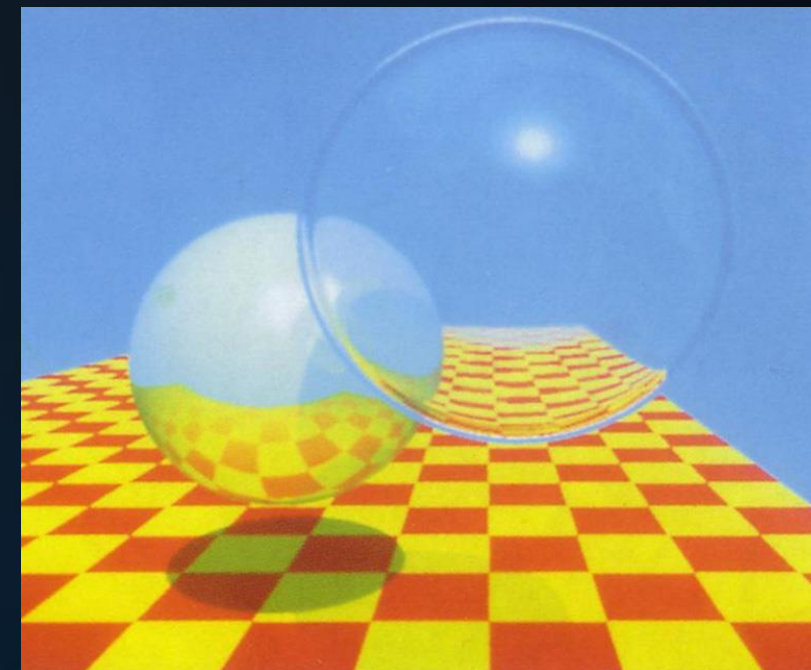
```
Color Trace( ray r )
    I,  $\vec{N}$ , mat = NearestIntersection( scene, r )
    return mat.color * DirectIllumination( I,  $\vec{N}$  )
```

“Direct illumination”:

Summed contribution of *unoccluded* point light sources, taking into account:

- Distance to I
- Angle between $\vec{L}$ and $\vec{N}$
- Intensity of light source

Note that this requires a ray per light source.



Whitted

An Improved Illumination Model for Shaded Display

```

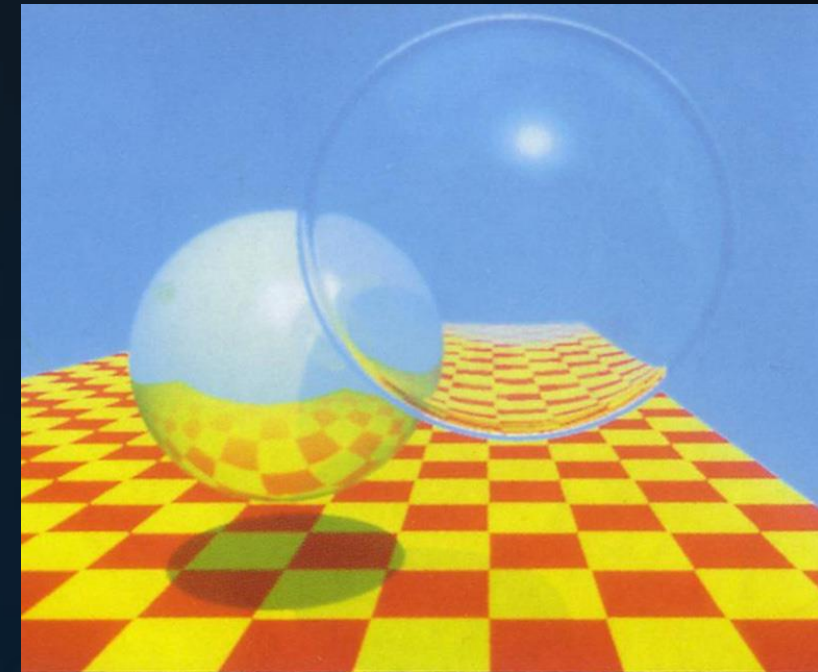
Color Trace( ray r )
    I,  $\vec{N}$ , mat = NearestIntersection( scene, r )
    if (mat == DIFFUSE)
        return mat.color * DirectIllumination( I,  $\vec{N}$  )
    if (mat == MIRROR)
        return mat.color * Trace( I, reflect( r,  $\vec{D}$ ,  $\vec{N}$  )

```

Indirect illumination:

For perfect specular object (mirrors) we extend the primary ray with an extension ray:

- We still modulate transport with the material color
- We do not apply $\vec{N} \cdot \vec{L}$
- We do not calculate direct illumination



Whitted

Reflection

Given a ray direction $\vec{D}$ and a normalized surface normal $\vec{N}$, the reflected vector $\vec{R} = \vec{D} - 2(\vec{D} \cdot \vec{N})\vec{N}$.

Derivation:

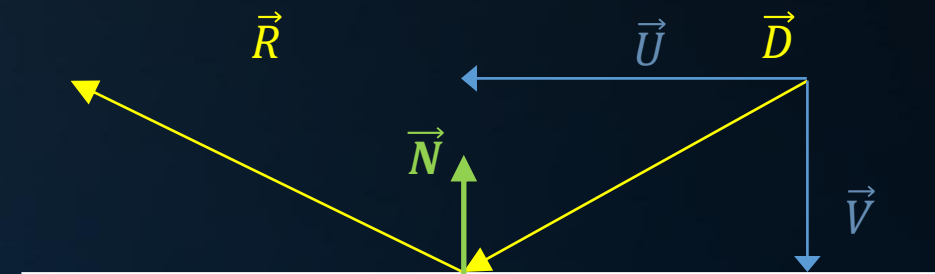
$$\vec{V} = \vec{N}(\vec{D} \cdot \vec{N})$$

$$\vec{U} = \vec{D} - \vec{V}$$

$$\vec{R} = \vec{U} + (-\vec{V})$$

$$\vec{R} = \vec{D} - \vec{N}(\vec{D} \cdot \vec{N}) - \vec{N}(\vec{D} \cdot \vec{N})$$

$$\vec{R} = \vec{D} - 2(\vec{D} \cdot \vec{N})\vec{N}$$



Whitted

```

ics
& (depth < MAXDEPTH)
{
    if (inside) {
        nt = nt / nc; ddn = ddn * ddn;
        rnt = 1.0f - rnt * ddn;
        D, N );
    }
    at a = nt - nc, b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * rnt);
    Tr) R = (D * nnt - N * (ddn
    E * diffuse;
    = true;
    -
    efl + refr)) && (depth < MAXDEPTH)
    D, N );
    -refl * E * diffuse;
    = true;
    MAXDEPTH)
    survive = SurvivalProbability( diffuse,
    estimation - doing it properly, closely
    if;
    radiance = SampleLight( &rand, I, &L, &lightP;
    e.x + radiance.y + radiance.z) > 0) && (survive
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
    random walk - done properly, closely following Smith's
    vive)
    ;
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;

```

Question 1: For direct illumination, we take into account:

- Material color
- Distance to light source
- $\vec{N} \cdot \vec{L}$

Why?

Question 2: We use the summed contribution of all light sources.

Is this correct?

Question 3: Why do we not sample the light sources for a pure specular surface? (*can you cast a shadow on a bathroom mirror?*)

Question 4: Show geometrically that, for normalized vectors $\vec{D}$ and $\vec{N}$, $\vec{R} = \vec{D} - 2(\vec{D} \cdot \vec{N})\vec{N}$ yields a normalized vector.



Whitted

An Improved Illumination Model for Shaded Display

Handling partially reflective materials:

```
Color Trace( ray r )
{
    I,  $\vec{N}$ , mat = NearestIntersection( scene, r )
    s = mat.specularity
    d = 1 - mat.specularity
    return mat.color * (
        s * Trace( ray( I, reflect( r. $\vec{D}$ ,  $\vec{N}$  ) ) ) +
        d * DirectIllumination( I,  $\vec{N}$  ) )
}
```

Note: this is not efficient. (why not?)



Whitted

An Improved Illumination Model for Shaded Display

Dielectrics

```

Color Trace( ray r )
{
    I,  $\vec{N}$ , mat = NearestIntersection( scene, r )
    if (mat == DIFFUSE)
        return mat.color * DirectIllumination( I,  $\vec{N}$  )
    if (mat == MIRROR)
        return mat.color * Trace( I, reflect( r. $\vec{D}$ ,  $\vec{N}$  ) )
    if (mat == GLASS)
        return mat.color * ?
}

```



Whitted

Dielectrics

The direction of the transmitted vector $\vec{T}$ depends on the refraction indices n_1, n_2 of the media separated by the surface. According to Snell's Law:

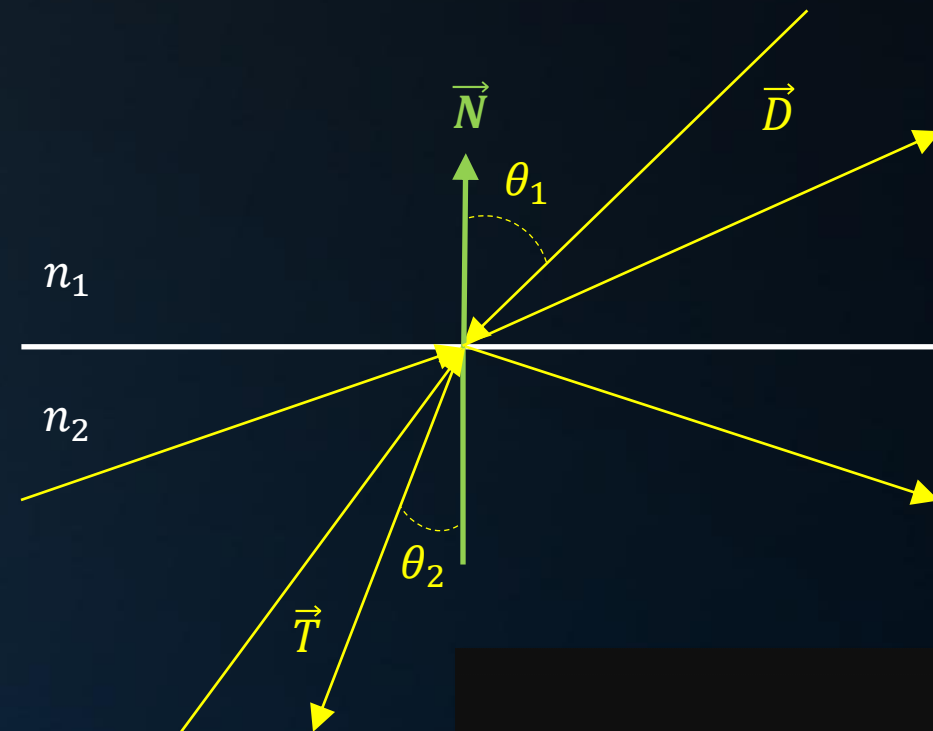
$$n_1 \sin \theta_1 = n_2 \sin \theta_2$$

or

$$\frac{n_1}{n_2} \sin \theta_1 = \sin \theta_2$$

Note: left term may exceed 1, in which case θ_2 cannot be computed. Therefore:

$$\frac{n_1}{n_2} \sin \theta_1 = \sin \theta_2 \Leftrightarrow \sin \theta_1 \leq \frac{n_2}{n_1} \rightarrow \theta_{critical} = \arcsin\left(\frac{n_2}{n_1} \sin \theta_2\right)$$



Whitted

```

ics
& (depth < MAXDEPTH)
{
    if (c == inside ? 1.0f : 0.0f)
    {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    at a = nt - nc, b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * cos2t);
    Tr) R = (D * nt - N * (ddn * cos2t));
    E * diffuse;
    = true;
    refl + refr)) && (depth < MAXDEPTH)
    D, N );
    refl * E * diffuse;
    = true;
    MAXDEPTH)
    survive = SurvivalProbability( diffuse );
    estimation - doing it properly, closely following
    if;
    radiance = SampleLight( &rand, I, &L, &light;
    e.x + radiance.y + radiance.z > 0) && (depth <
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
    random walk - done properly, closely following Small's
    vive)
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;

```



Whitted

```

ics
& (depth < MAXDEPTH)
{
    if (inside & !inside)
    {
        nt = nt / nc; ddn = ddn * nt;
        r = 1.0f - nnt * nnt;
        D, N );
    }
    else
    {
        at a = nt - nc, b = nt * nc;
        at Tr = 1 - (R0 + (1 - R0) * r);
        Tr) R = (D * nnt - N * (ddn
    }
    E * diffuse;
    = true;
    -
    refl + refr)) && (depth < MAXDEPTH)
    {
        D, N );
        refl * E * diffuse;
        = true;
    }
    MAXDEPTH)
    survive = SurvivalProbability( diffuse );
    estimation - doing it properly, closely
    df;
    radiance = SampleLight( &rand, I, &L, &light
    e.x + radiance.y + radiance.z) > 0) && (out
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurv
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
    random walk - done properly, closely following Snell's
    vive)
    ;
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;

```



https://en.wikipedia.org/wiki/Snell%27s_window



Whitted

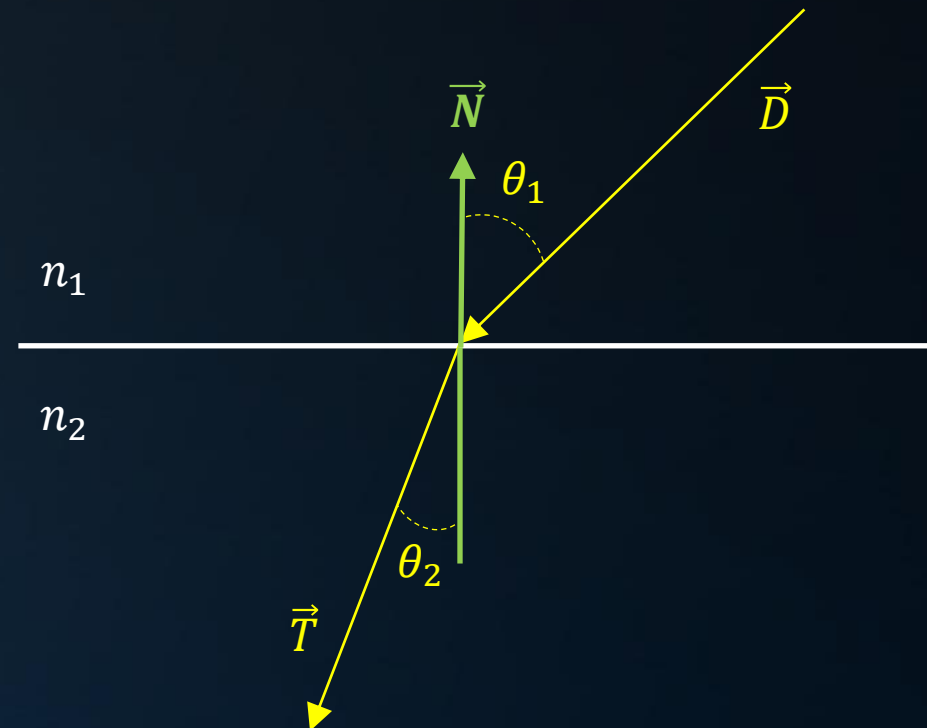
Dielectrics

$$\frac{n_1}{n_2} \sin \theta_1 = \sin \theta_2 \Leftrightarrow \sin \theta_1 \leq \frac{n_2}{n_1}$$

$$k = 1 - \left(\frac{n_1}{n_2} \right)^2 (1 - \cos \theta_1^2)$$

$$\vec{T} = \begin{cases} TIR, & \text{for } k < 0 \\ \frac{n_1}{n_2} \vec{D} + \vec{N} \left(\frac{n_1}{n_2} \cos \theta_1 - \sqrt{k} \right), & \text{for } k \geq 0 \end{cases}$$

Note: $\cos \theta_1 = \vec{N} \cdot -\vec{D}$, and $\frac{n_1}{n_2}$ should be calculated only once.



* For a full derivation, see http://www.flipcode.com/archives/reflection_transmission.pdf

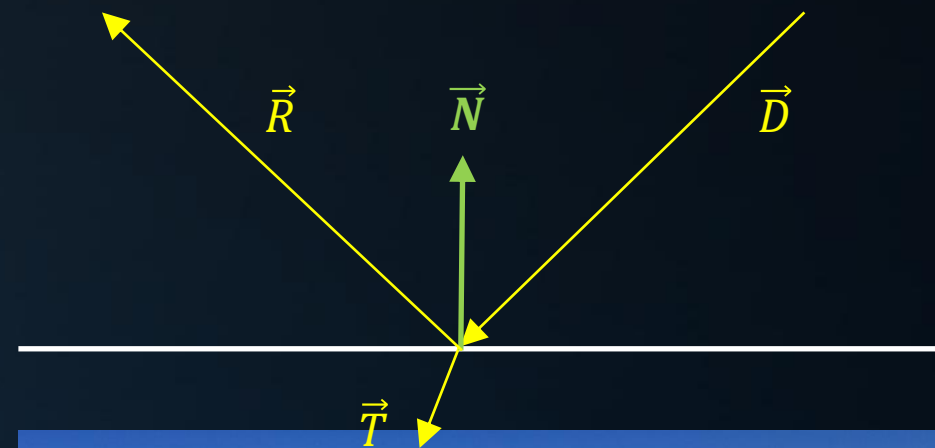


Whitted

Dielectrics

A typical dielectric transmits *and* reflects light.

```
ics
& (depth < MAXDEPTH)
{
    if (c == inside ? 1 : -1)
    {
        nt = nt / nc; ddn = dot(N, D);
        nnt = 1.0f - nnt * nnt;
        D = D - 2 * N * ddn;
        D = D + 2 * N * ddn;
    }
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    R = (D * nnt - N * (ddn < 0 ? 1 : -1));
    E * diffuse;
    = true;
    (refl + refr)) && (depth < MAXDEPTH)
    {
        D, N );
        refl * E * diffuse;
        = true;
    }
    MAXDEPTH)
    survive = SurvivalProbability( diff
    estimation - doing it properly, c
    df;
    radiance = SampleLight( &rand, I, &
    e.x + radiance.y + radiance.z) > 0)
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPd
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / dire
    random walk - done properly, closely
    vive)
    ;
    at3 brdf = SampleDiffuse( diffuse,
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;
}
```



Whitted

Dielectrics

A typical dielectric transmits *and* reflects light.

Based on the Fresnel equations, the reflectivity of the surface for non-polarized light is formulated as:

$$F_r = \frac{1}{2} \left(\underbrace{\left(\frac{n_1 \cos \theta_i - n_2 \cos \theta_t}{n_1 \cos \theta_i + n_2 \cos \theta_t} \right)^2}_{\text{Reflectance for s-polarized light}} + \underbrace{\left(\frac{n_1 \cos \theta_t - n_2 \cos \theta_i}{n_1 \cos \theta_t + n_2 \cos \theta_i} \right)^2}_{\text{Reflectance for p-polarized light}} \right)$$

Reflectance for unpolarized light

Where: $\cos \theta_t = \sqrt{1 - \left(\frac{n_1}{n_2} \sin \theta_i \right)^2}$



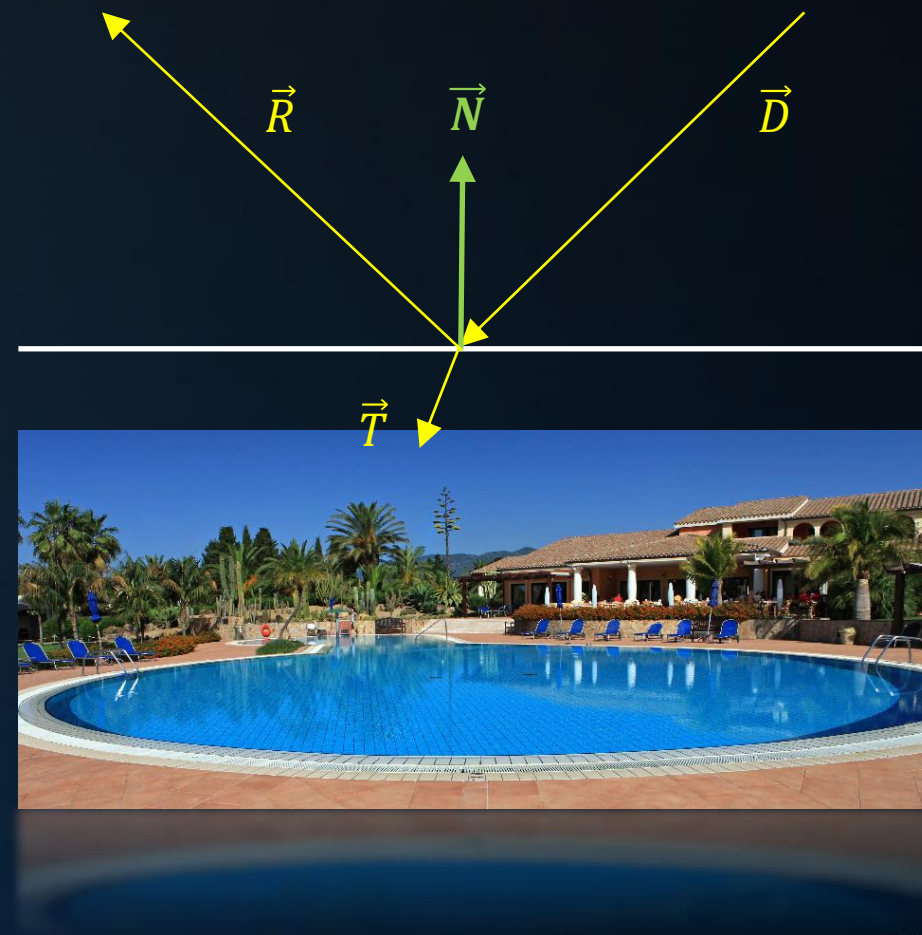
Whitted

Dielectrics

$$F_r = \dots$$

Based on the law of conservation of energy:

$$F_t = 1 - F_r$$



Whitted

Ray Tracing

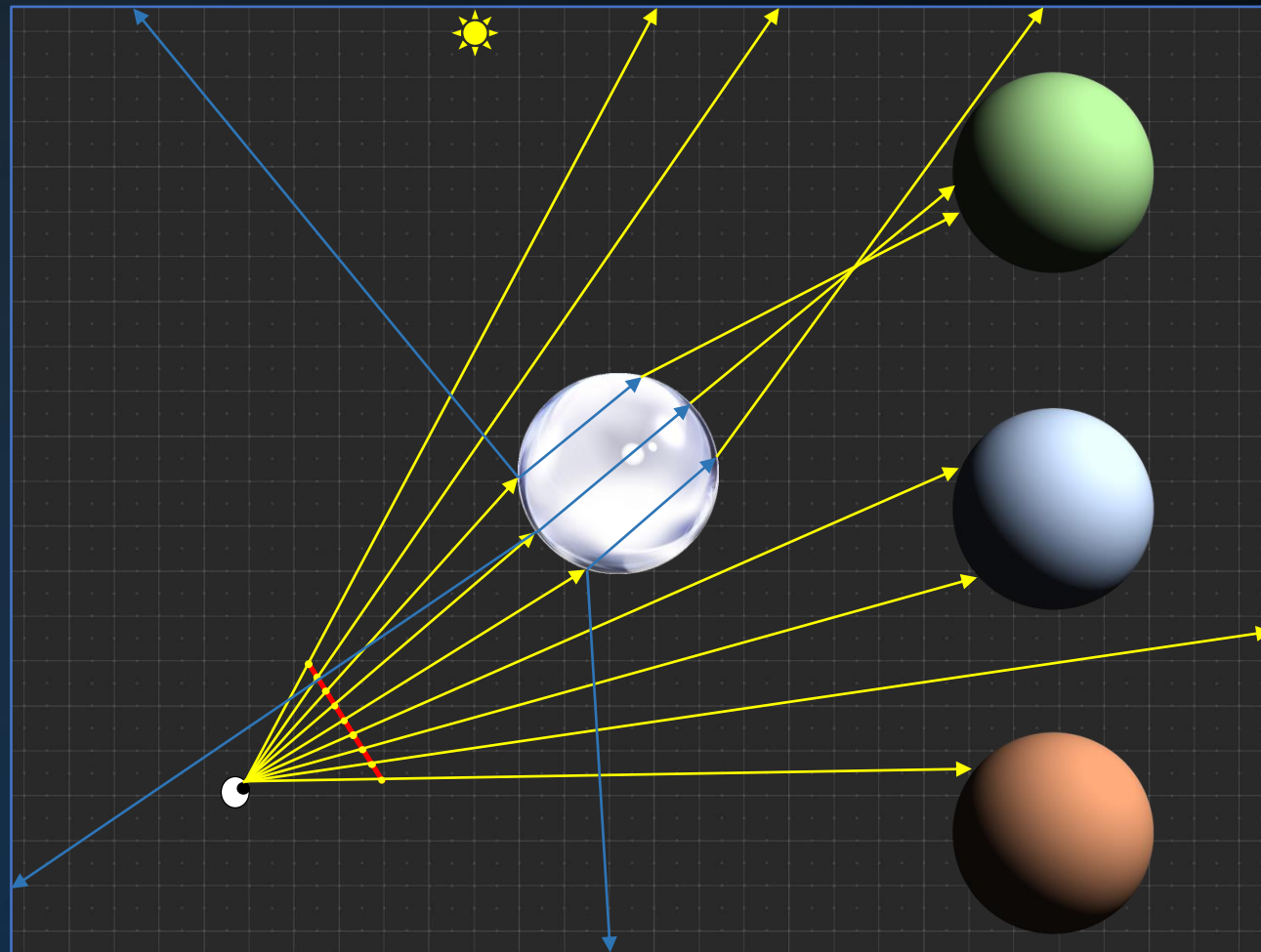
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

- Extension rays

Light transport



Whitted

Ray Tree

Using Whitted-style ray tracing, hitting a surface point may spawn:

- a shadow ray for each light source;
- a reflection ray;
- a ray transmitted into the material.

The reflected and transmitted rays may hit another object with the same material.

➔ A single primary ray may lead to a very large number of ray queries.



Question 5: imagine a scene with several point lights and dielectric materials. Considering the law of conservation of energy, what can you say about the energy transported by each individual ray?

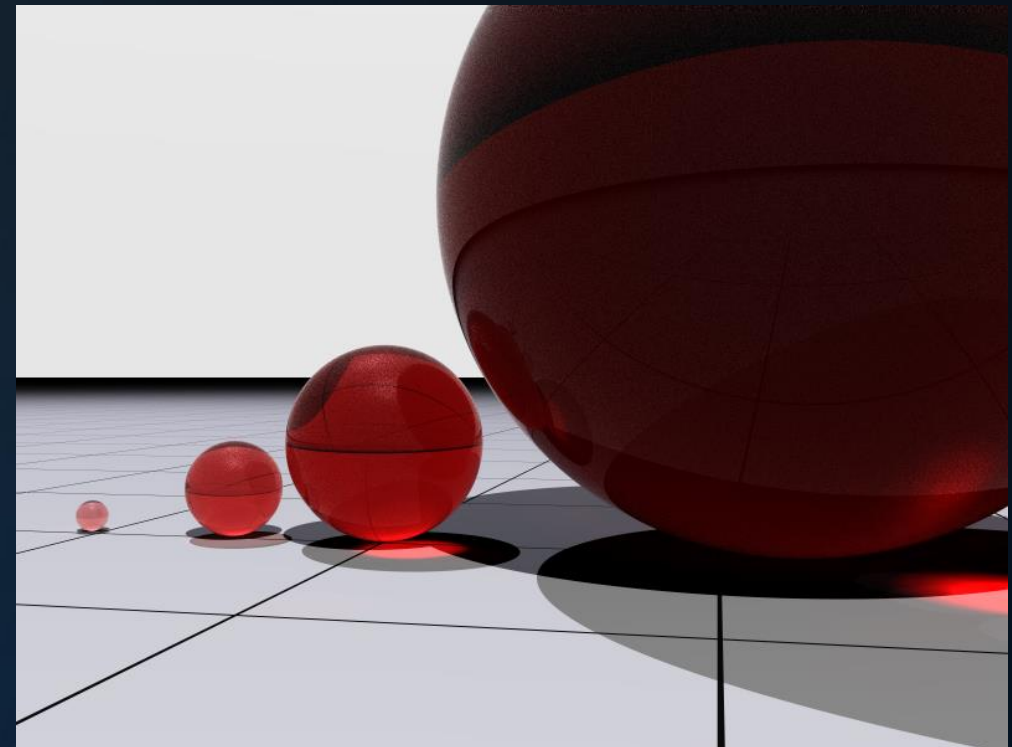
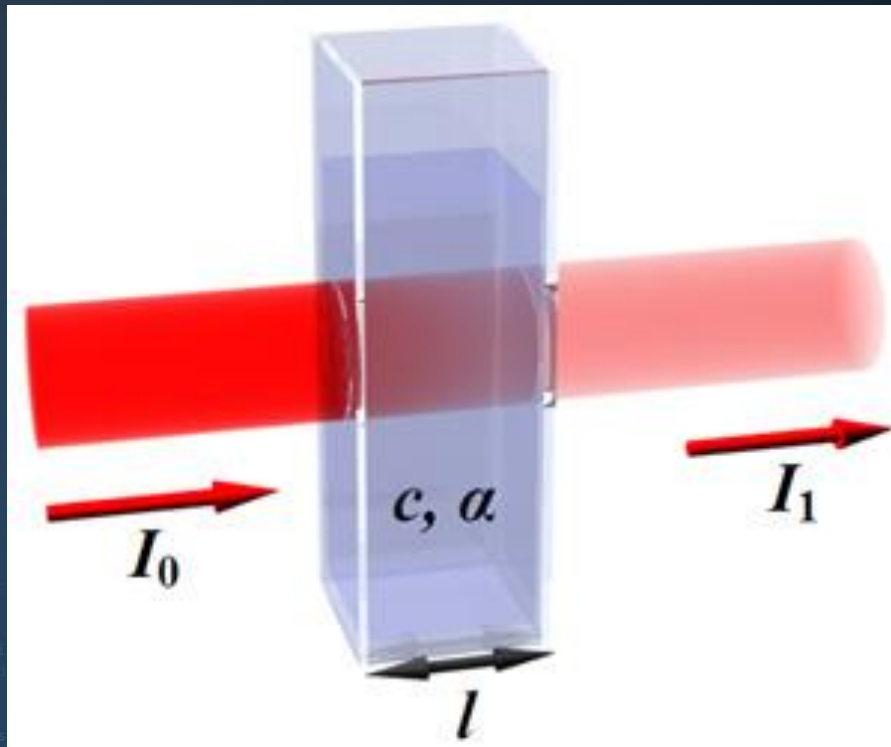
```

1113
1114 (depth < MAXDEPTH)
1115 {
1116     if (nt < inside) { 1; nnt = nnt + 1;
1117         nnt = nt / nc; ddn = dot(N, D);
1118         cos2t = 1.0f - nnt * nnt;
1119         D = D - D * ddn;
1120         D = D * (1.0f / sqrt(cos2t));
1121         D = D * N;
1122         D = D * (1.0f / sqrt(1.0f - cos2t));
1123         at a = nt - nc, b = nt - nc;
1124         Tr = 1 - (R0 + (1 - R0) * a);
1125         Fr = (D * nnt - N * (ddn * a + b));
1126         E = diffuse;
1127         = true;
1128         -
1129         refl + refr)) && (depth < MAXDEPTH)
1130         {
1131             D, N );
1132             refl * E * diffuse;
1133             = true;
1134             -
1135             MAXDEPTH)
1136             {
1137                 survive = SurvivalProbability( diffuse, N );
1138                 // - doing it properly, closely following
1139                 // diffuse;
1140                 radiance = SampleLight( &rand, I, &L, &lightDir,
1141                     e.x + radiance.y + radiance.z) > 0) && (outs.N
1142                     w = true;
1143                     at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
1144                     at3 factor = diffuse * INVPI;
1145                     at weight = Mis2( directPdf, brdfPdf );
1146                     at cosThetaOut = dot( N, L );
1147                     E * ((weight * cosThetaOut) / directPdf) * (radiance
1148                     random walk - done properly, closely following Seelye
1149                     (survive)
1150                     ;
1151                     at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
1152                     survive;
1153                     pdf;
1154                     n = E * brdf * (dot( N, R ) / pdf);
1155                     sion = true;
1156                     }
1157             }
1158         }
1159     }
1160 }
1161
1162 //
1163 //
1164 //
1165 //
1166 //
1167 //
1168 //
1169 //
1170 //
1171 //
1172 //
1173 //
1174 //
1175 //
1176 //
1177 //
1178 //
1179 //
1180 //
1181 //
1182 //
1183 //
1184 //
1185 //
1186 //
1187 //
1188 //
1189 //
1190 //
1191 //
1192 //
1193 //
1194 //
1195 //
1196 //
1197 //
1198 //
1199 //
1200 //
1201 //
1202 //
1203 //
1204 //
1205 //
1206 //
1207 //
1208 //
1209 //
1210 //
1211 //
1212 //
1213 //
1214 //
1215 //
1216 //
1217 //
1218 //
1219 //
1220 //
1221 //
1222 //
1223 //
1224 //
1225 //
1226 //
1227 //
1228 //
1229 //
1230 //
1231 //
1232 //
1233 //
1234 //
1235 //
1236 //
1237 //
1238 //
1239 //
1240 //
1241 //
1242 //
1243 //
1244 //
1245 //
1246 //
1247 //
1248 //
1249 //
1250 //
1251 //
1252 //
1253 //
1254 //
1255 //
1256 //
1257 //
1258 //
1259 //
1260 //
1261 //
1262 //
1263 //
1264 //
1265 //
1266 //
1267 //
1268 //
1269 //
1270 //
1271 //
1272 //
1273 //
1274 //
1275 //
1276 //
1277 //
1278 //
1279 //
1280 //
1281 //
1282 //
1283 //
1284 //
1285 //
1286 //
1287 //
1288 //
1289 //
1290 //
1291 //
1292 //
1293 //
1294 //
1295 //
1296 //
1297 //
1298 //
1299 //
1300 //
1301 //
1302 //
1303 //
1304 //
1305 //
1306 //
1307 //
1308 //
1309 //
1310 //
1311 //
1312 //
1313 //
1314 //
1315 //
1316 //
1317 //
1318 //
1319 //
1320 //
1321 //
1322 //
1323 //
1324 //
1325 //
1326 //
1327 //
1328 //
1329 //
1330 //
1331 //
1332 //
1333 //
1334 //
1335 //
1336 //
1337 //
1338 //
1339 //
1340 //
1341 //
1342 //
1343 //
1344 //
1345 //
1346 //
1347 //
1348 //
1349 //
1350 //
1351 //
1352 //
1353 //
1354 //
1355 //
1356 //
1357 //
1358 //
1359 //
1360 //
1361 //
1362 //
1363 //
1364 //
1365 //
1366 //
1367 //
1368 //
1369 //
1370 //
1371 //
1372 //
1373 //
1374 //
1375 //
1376 //
1377 //
1378 //
1379 //
1380 //
1381 //
1382 //
1383 //
1384 //
1385 //
1386 //
1387 //
1388 //
1389 //
1390 //
1391 //
1392 //
1393 //
1394 //
1395 //
1396 //
1397 //
1398 //
1399 //
1400 //
1401 //
1402 //
1403 //
1404 //
1405 //
1406 //
1407 //
1408 //
1409 //
1410 //
1411 //
1412 //
1413 //
1414 //
1415 //
1416 //
1417 //
1418 //
1419 //
1420 //
1421 //
1422 //
1423 //
1424 //
1425 //
1426 //
1427 //
1428 //
1429 //
1430 //
1431 //
1432 //
1433 //
1434 //
1435 //
1436 //
1437 //
1438 //
1439 //
1440 //
1441 //
1442 //
1443 //
1444 //
1445 //
1446 //
1447 //
1448 //
1449 //
1450 //
1451 //
1452 //
1453 //
1454 //
1455 //
1456 //
1457 //
1458 //
1459 //
1460 //
1461 //
1462 //
1463 //
1464 //
1465 //
1466 //
1467 //
1468 //
1469 //
1470 //
1471 //
1472 //
1473 //
1474 //
1475 //
1476 //
1477 //
1478 //
1479 //
1480 //
1481 //
1482 //
1483 //
1484 //
1485 //
1486 //
1487 //
1488 //
1489 //
1490 //
1491 //
1492 //
1493 //
1494 //
1495 //
1496 //
1497 //
1498 //
1499 //
1500 //
1501 //
1502 //
1503 //
1504 //
1505 //
1506 //
1507 //
1508 //
1509 //
1510 //
1511 //
1512 //
1513 //
1514 //
1515 //
1516 //
1517 //
1518 //
1519 //
1520 //
1521 //
1522 //
1523 //
1524 //
1525 //
1526 //
1527 //
1528 //
1529 //
1530 //
1531 //
1532 //
1533 //
1534 //
1535 //
1536 //
1537 //
1538 //
1539 //
1540 //
1541 //
1542 //
1543 //
1544 //
1545 //
1546 //
1547 //
1548 //
1549 //
1550 //
1551 //
1552 //
1553 //
1554 //
1555 //
1556 //
1557 //
1558 //
1559 //
1560 //
1561 //
1562 //
1563 //
1564 //
1565 //
1566 //
1567 //
1568 //
1569 //
1570 //
1571 //
1572 //
1573 //
1574 //
1575 //
1576 //
1577 //
1578 //
1579 //
1580 //
1581 //
1582 //
1583 //
1584 //
1585 //
1586 //
1587 //
1588 //
1589 //
1590 //
1591 //
1592 //
1593 //
1594 //
1595 //
1596 //
1597 //
1598 //
1599 //
1600 //
1601 //
1602 //
1603 //
1604 //
1605 //
1606 //
1607 //
1608 //
1609 //
1610 //
1611 //
1612 //
1613 //
1614 //
1615 //
1616 //
1617 //
1618 //
1619 //
1620 //
1621 //
1622 //
1623 //
1624 //
1625 //
1626 //
1627 //
1628 //
1629 //
1630 //
1631 //
1632 //
1633 //
1634 //
1635 //
1636 //
1637 //
1638 //
1639 //
1640 //
1641 //
1642 //
1643 //
1644 //
1645 //
1646 //
1647 //
1648 //
1649 //
1650 //
1651 //
1652 //
1653 //
1654 //
1655 //
1656 //
1657 //
1658 //
1659 //
1660 //
1661 //
1662 //
1663 //
1664 //
1665 //
1666 //
1667 //
1668 //
1669 //
1670 //
1671 //
1672 //
1673 //
1674 //
1675 //
1676 //
1677 //
1678 //
1679 //
1680 //
1681 //
1682 //
1683 //
1684 //
1685 //
1686 //
1687 //
1688 //
1689 //
1690 //
1691 //
1692 //
1693 //
1694 //
1695 //
1696 //
1697 //
1698 //
1699 //
1700 //
1701 //
1702 //
1703 //
1704 //
1705 //
1706 //
1707 //
1708 //
1709 //
1710 //
1711 //
1712 //
1713 //
1714 //
1715 //
1716 //
1717 //
1718 //
1719 //
1720 //
1721 //
1722 //
1723 //
1724 //
1725
```



Whitted

Beer's Law



Whitted

Beer's Law

Light travelling through a medium loses intensity due to absorption.

The intensity $I(d)$ that remains after travelling d units through a substance with absorption a is:

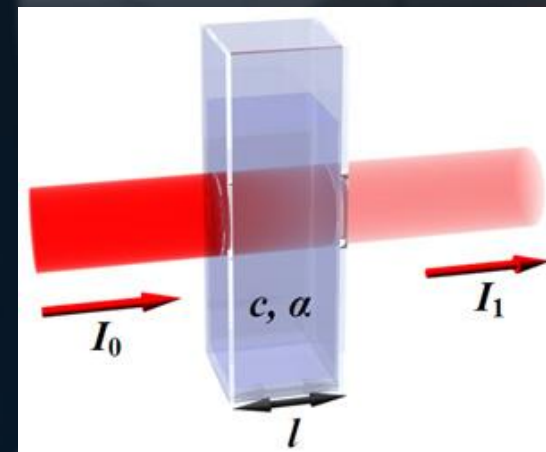
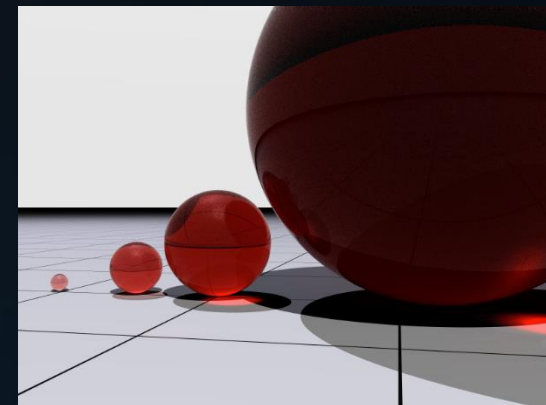
$$I(d) = I(0)e^{-\ln(a)d}$$

In pseudocode:

```

I.r *= exp( -a.r * d );
I.g *= exp( -a.g * d );
I.b *= exp( -a.b * d );

```



Whitted

Whitted - Summary

A Whitted-style ray tracer implements the following optical phenomena:

- Direct illumination of multiple light sources, taking into account

Visibility

Distance attenuation

A shading model: $N \cdot L$ for diffuse

- Pure specular reflections, with recursion
- Dielectrics, with Fresnel, with recursion
- Beer's Law

The ray tracer supports any primitive for which a ray/primitive intersection can be determined.



Today's Agenda:

- Introduction: Appel
- Whitted
- Cook

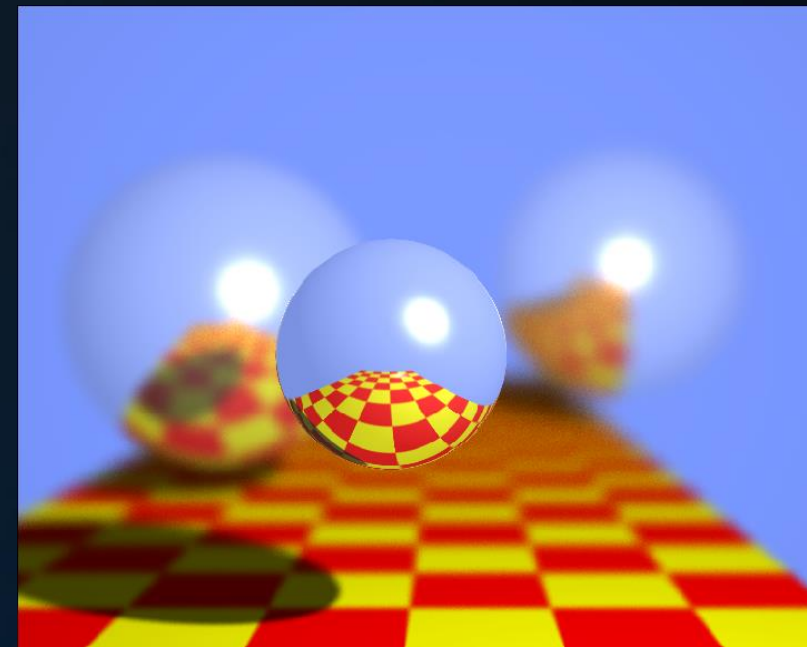
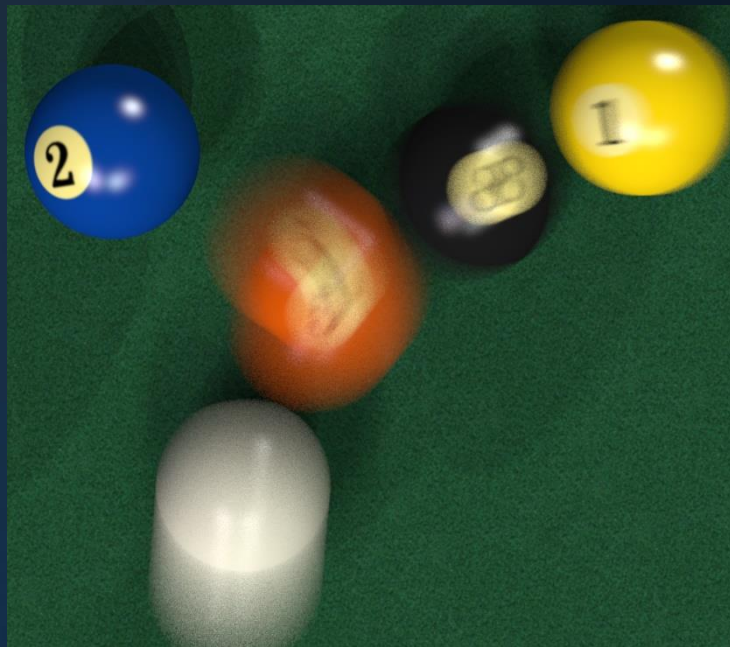
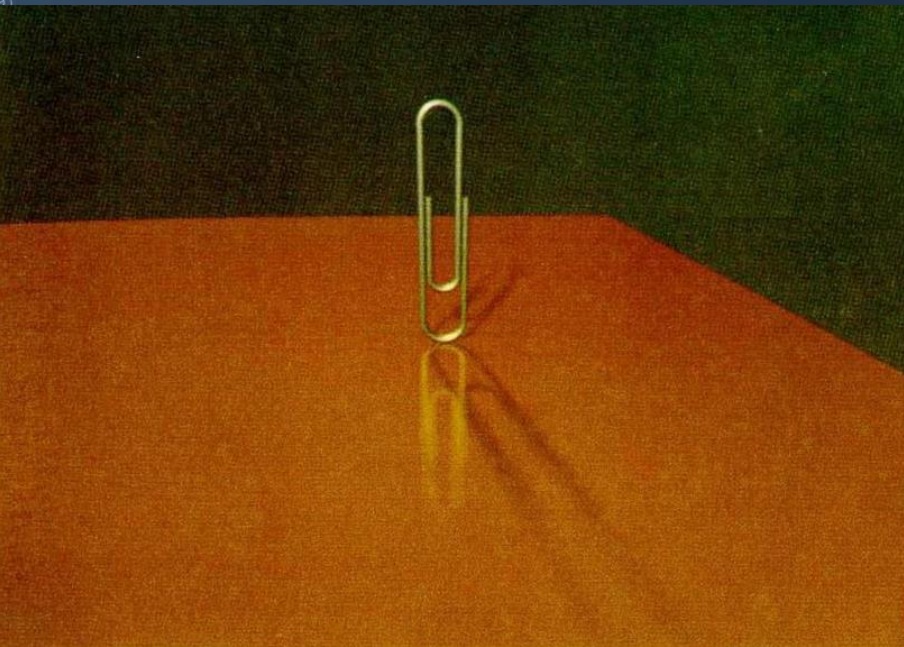


Cook

'Distributed Ray Tracing'

Whitted-style ray tracing does not handle glossy reflections, depth of field, motion blur.

```
ics
& (depth < MAXDEPTH)
{
    t = inside ? 1.0 : 0.0;
    nt = nt / nc, ddn = ddn * t;
    cos2t = 1.0f - nnt * ddn;
    D, N );
}
```



```
random walk - done properly, closely following the path of the
survive)
```

```
;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```



Cook

'Distributed Ray Tracing'

Whitted-style ray tracing does not handle glossy reflections, depth of field, motion blur:

Ray tracing is a point sampling process.

Cook et al.*:

Replace point sampling by integrals:

- Perform anti-aliasing by integrating over the pixel
- Add motion blur by integrating over time
- Calculate depth of field by integrating over the aperture.

* : Cook et al., 1984. Distributed Ray Tracing.



Today's Agenda:

- Introduction: Appel
- Whitted
- Cook



INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 – February 2023



END of “Whitted”

next lecture: “Light Transport”

