

```
ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0.25)
    {
        nt = nt / nc; ddn = ddn * ddn;
        r2t = 1.0f - nnt * nnt;
        D, N );
    }
    else
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * r2t);
        (Tr) R = (D * nnt - N * (ddn > 0.5 ? 1 : -1));

        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, R, Align );
            e.x + radiance.y + radiance.z );
            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;
        }
    }
}
```

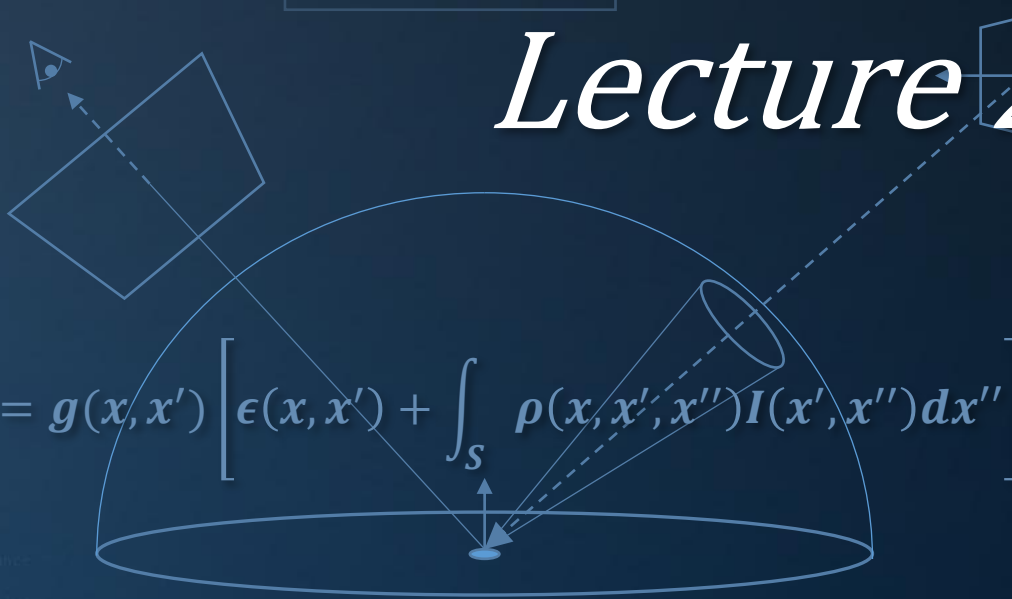


INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

Lecture 2 - “Whitted”

Welcome!



Today's Agenda:

- Introduction: Appel
- Whitted
- Cook



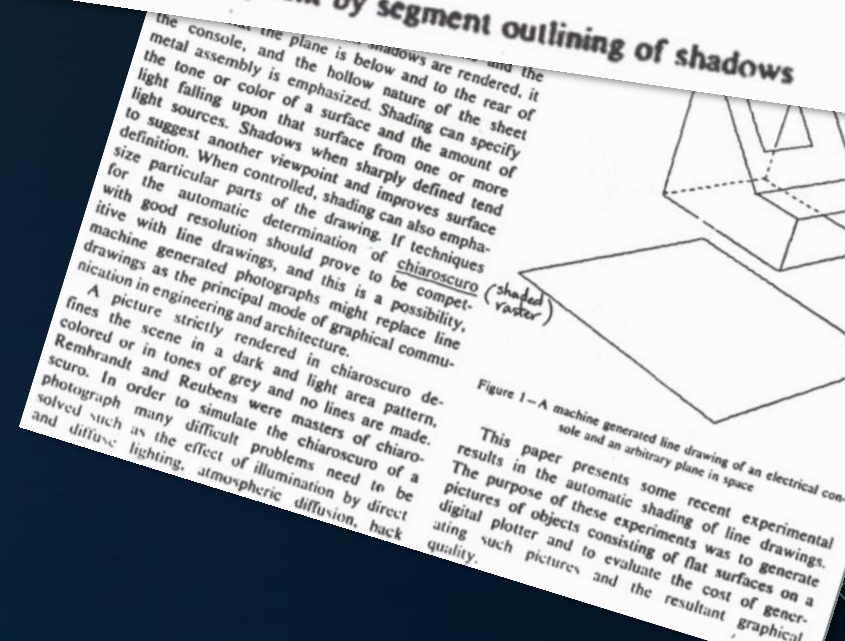
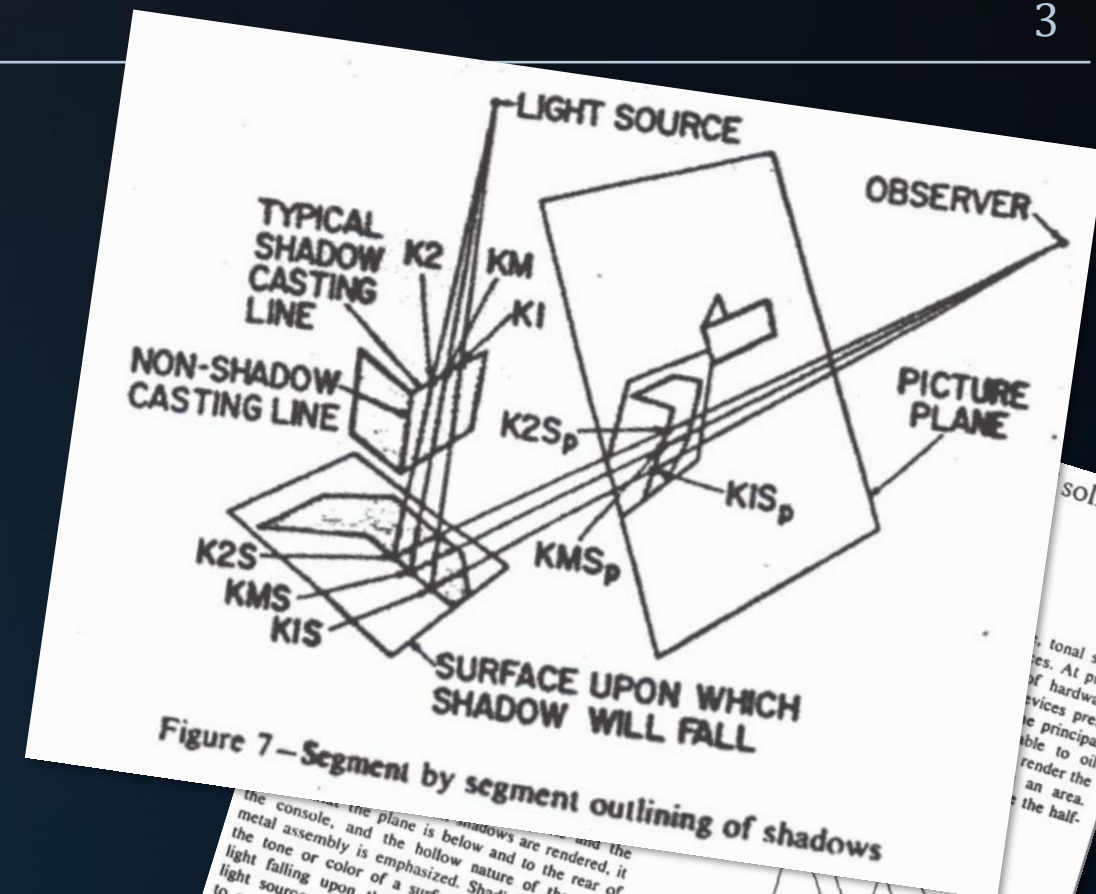
Appel

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

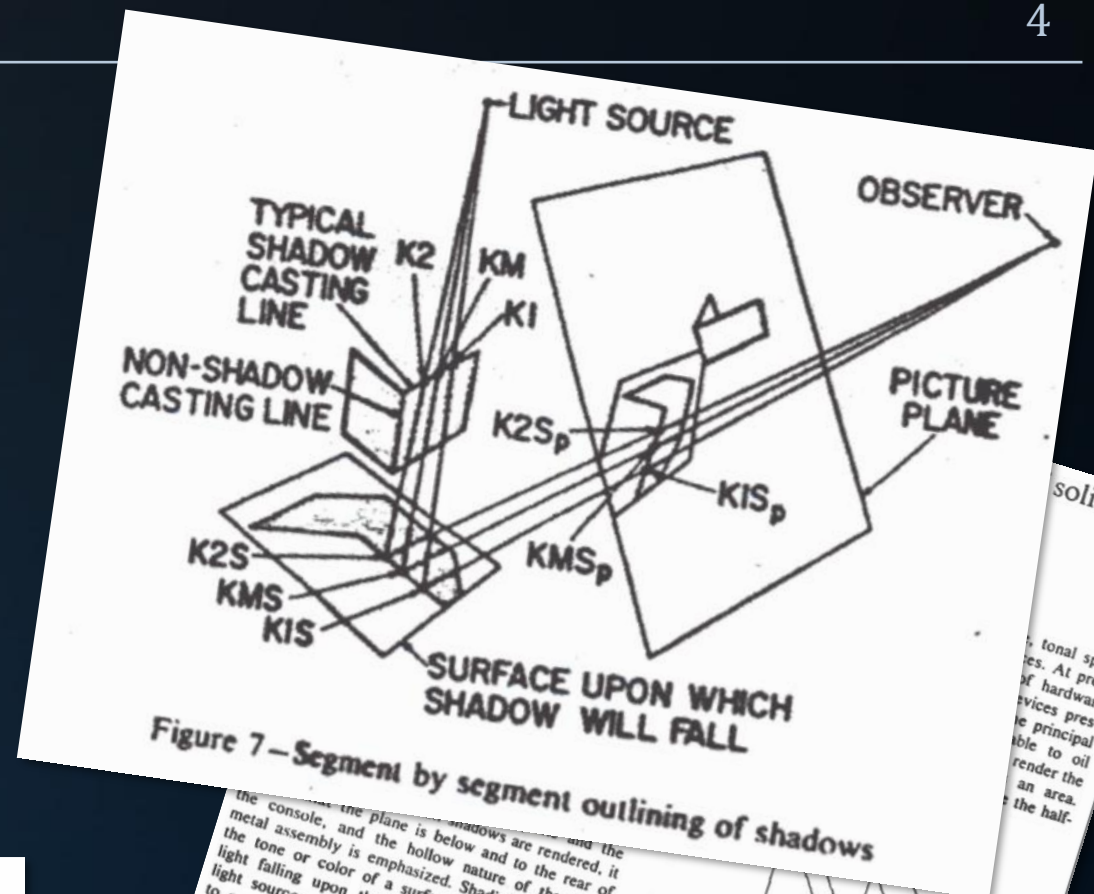


IBM 7090

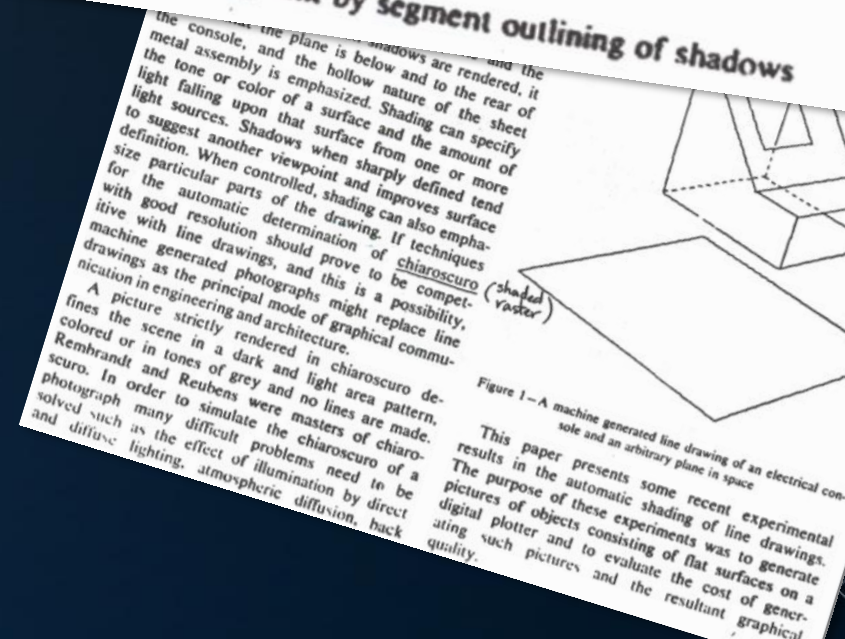
Appel

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

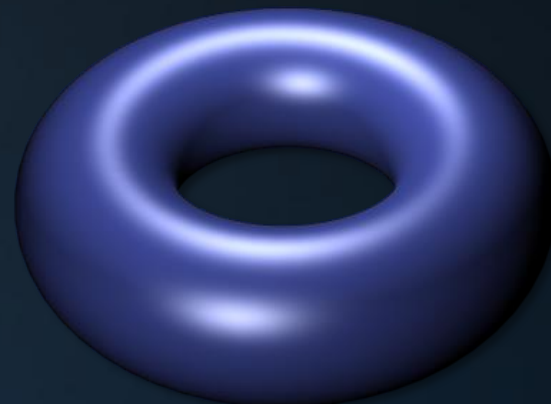
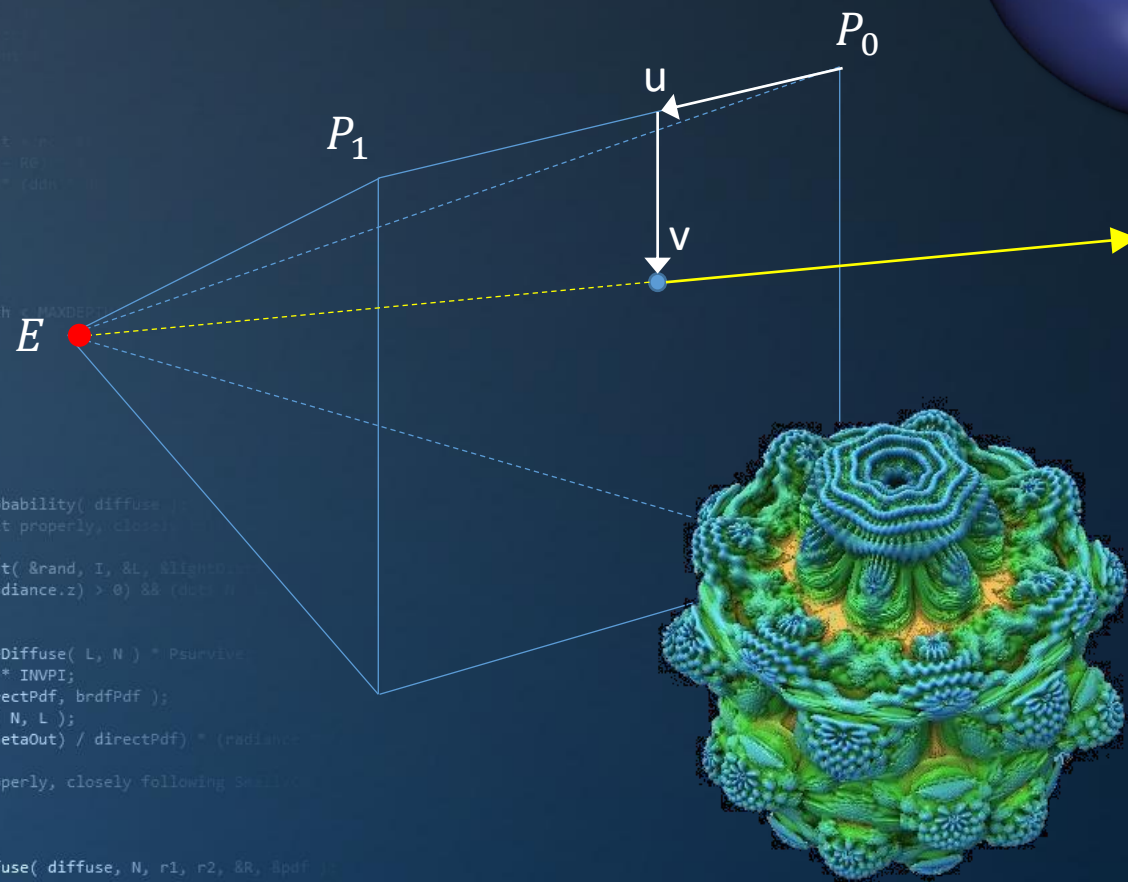


Appel

Recap

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        t = sqrt(cos2t);
        D, N);
    }
    else
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * t);
        at Tr = (D * nnt - N * (ddn * cos2t));
    }
    E * diffuse;
    = true;
    if (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, &N, &D );
    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);
    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 (t < inside ? 1.f : 1.25f)
    {
        nt = nt / nc, ddn = ddn * nc;
        r2s2t = 1.0f - nnt * ddn;
        D, N );
    }
    else
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * r2s2t);
        Tr) R = (D * nnt - N * (ddn < 0 ? 1 : -1));
    }
    E * diffuse;
    = true;
}
-
efl + 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, &lightPos );
    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);
tion = true;

```



Appel

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 1.0f - 0.9f * nt)
    {
        nt = nt / nc, ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    {
        a = nt - nc, b = nt + nc;
        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( 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 taking into account fall-off (Phong)
- Reflection, using environment maps (Blinn & Newell *)
- Stencil shadows (Williams **)

```

ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = dd * nt;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    at a = nt - nc, b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) *
    Tr) R = (D * nnt - N * (ddn
    E * diffuse;
    = true;
    defl + refr)) && (depth < MAXDEPTH)

```



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

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

```

random walk - done properly, closely following
ive)
;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2**
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Whitted

An Improved Illumination Model for Shaded Display

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

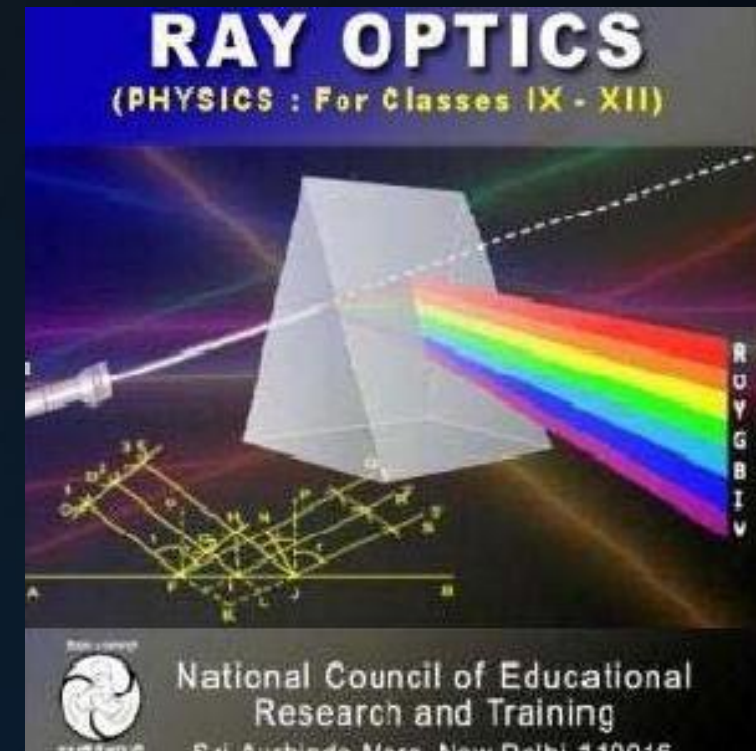
- Rasterization
- Shading: either diffuse ($N \cdot L$) or specular ($(N \cdot H)^n$), both not taking into account fall-off (Phong)
- Reflection, using environment maps (Blinn & Newell)
- Stencil shadows (Williams)

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:

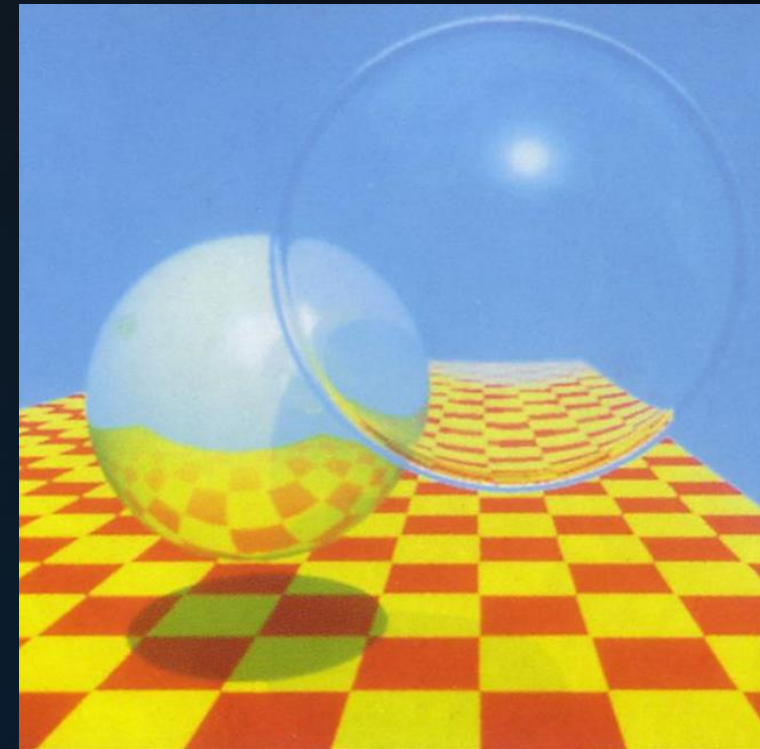
Light paths are generated (backwards) from the camera to the light sources, using rays to simulate optics.

```
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 )
```

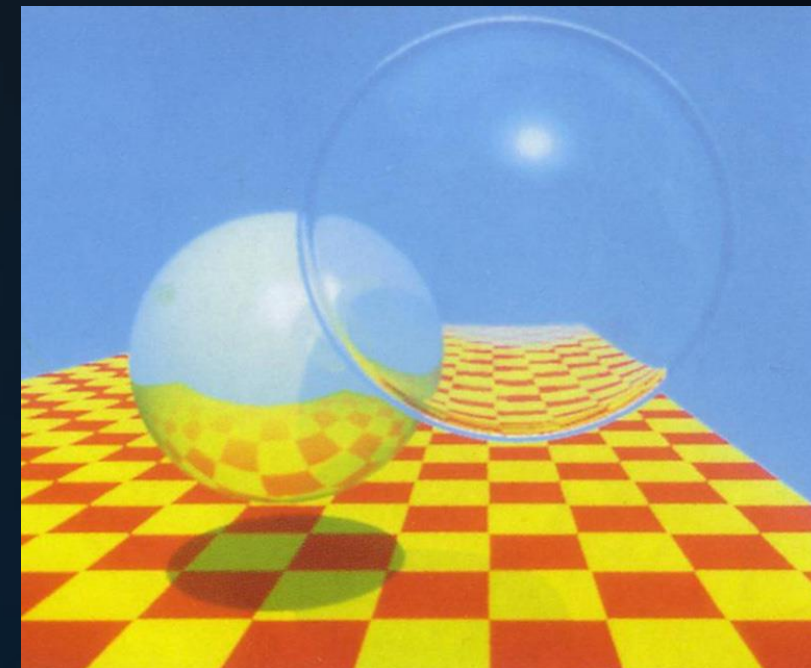
```
    return mat.color * DirectIllumination( I,  $\vec{N}$  )
```

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

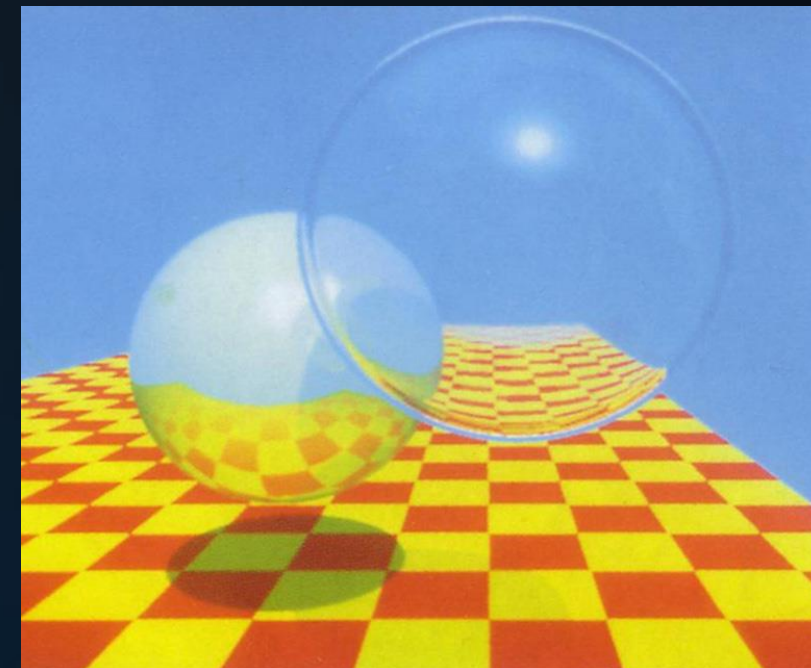
An Improved Illumination Model for Shaded Display

```
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:

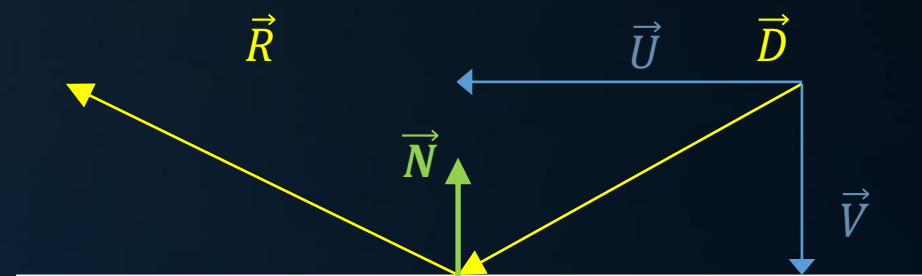
$$\vec{V} = \vec{N}(\vec{D} \cdot \vec{N})$$

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

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

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

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



```
h = E * brdf * (dot( N, R ) / pdf);  
sion = true;
```

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

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

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:

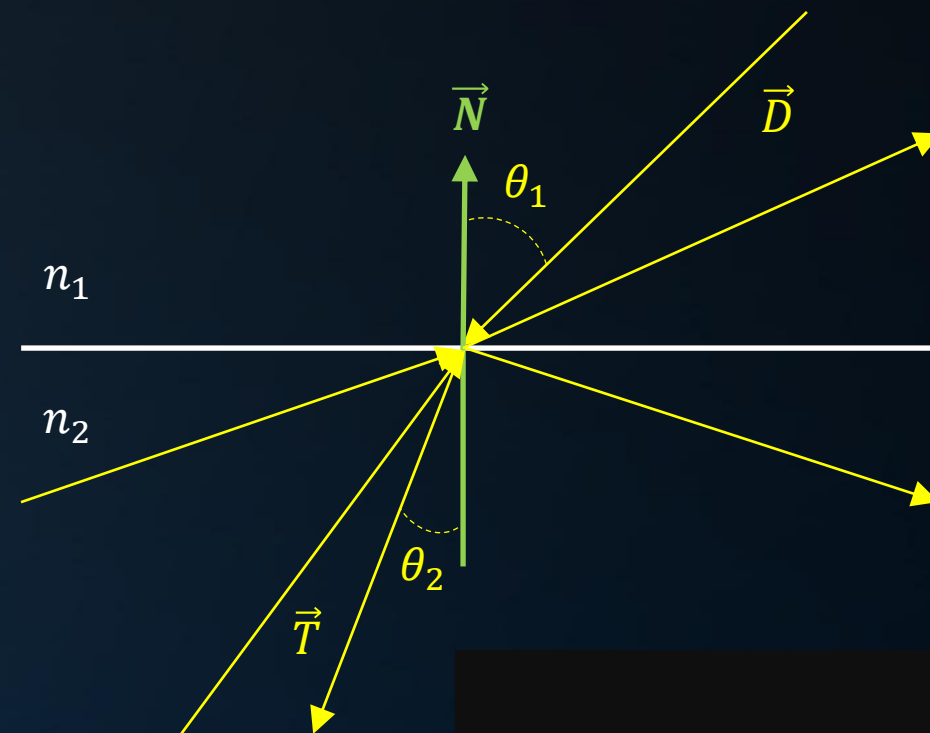
$$n_1 \sin \theta_1 = n_2 \sin \theta_2$$

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 (inside ? 1 : 0)
    {
        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 * 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
    df;
    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 ? 1 : 0)
    {
        nt = nt / nc; ddn = dd * nt;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    }
    else
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * ddn);
        Tr) R = (D * nnt - N * (ddn > 0 ? 1 : -1));
    }
    E * diffuse;
    = true;
    -
    refl + refr)) && (depth < MAXDEPTH)
    {
        D, N );
        refl * E * diffuse;
        = true;
    }
    MAXDEPTH)
    survive = SurvivalProbability( diffuse );
    estimation - doing it properly, closely following
    df;
    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 Snell's law
    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

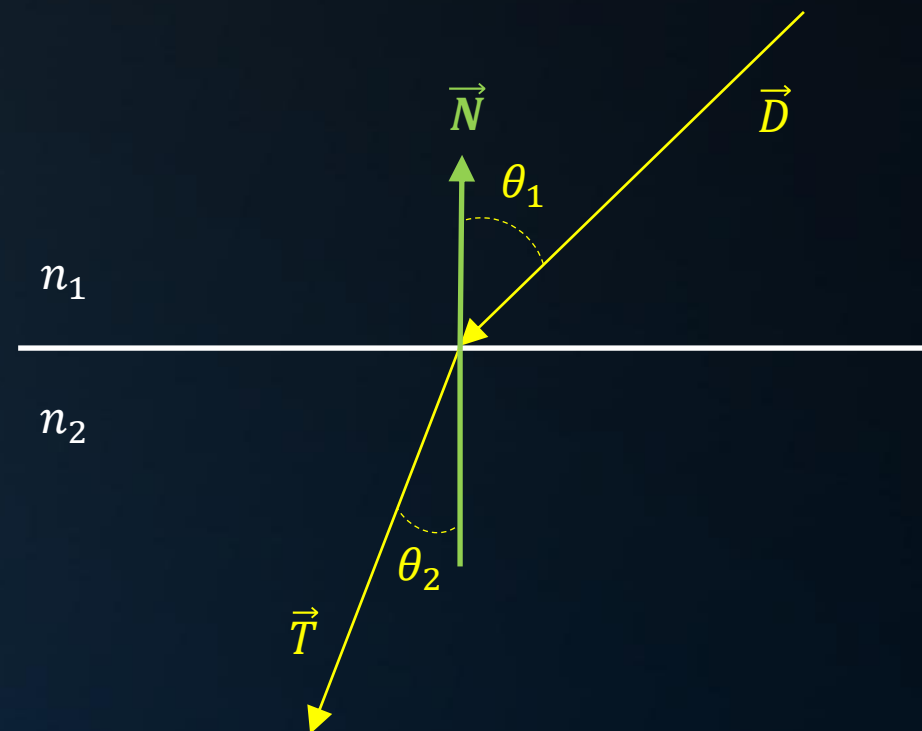
Dielectrics

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

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

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

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

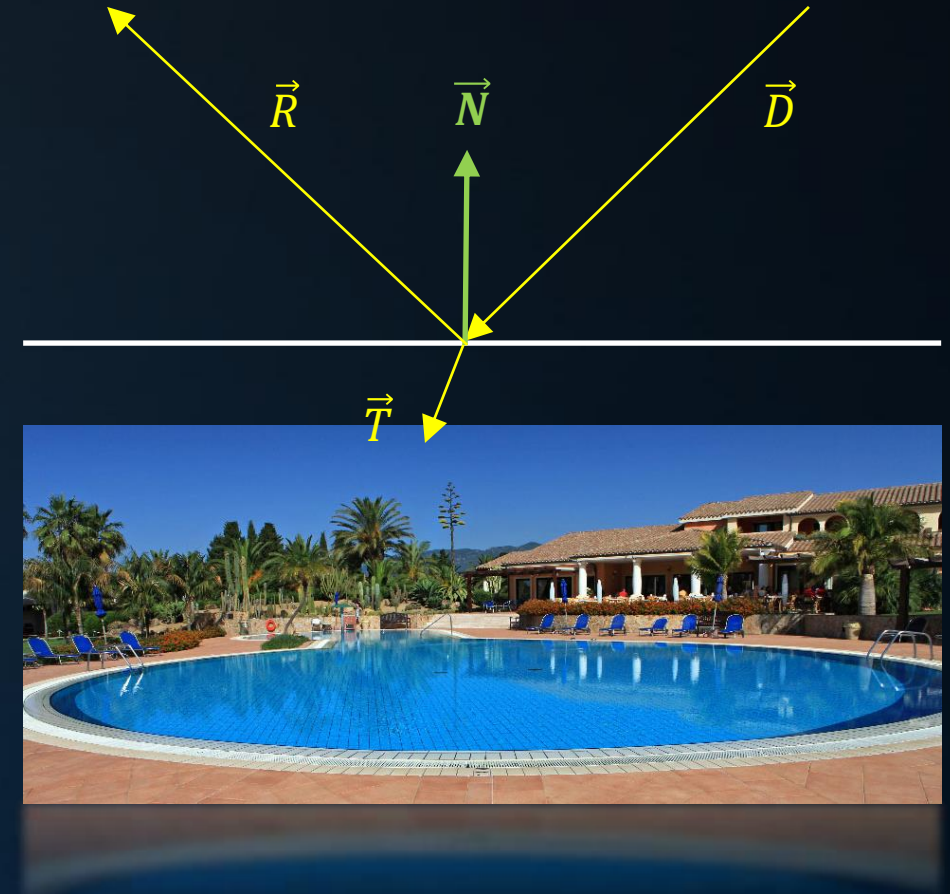


Whitted

Dielectrics

A typical dielectric transmits *and* reflects light.

```
ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : -1) * nt < 0)
        nt = nt / nc; ddn = dot(N, D);
        if (ddn < 0) ddn = -ddn;
        r0s2t = 1.0f - nnt * nnt;
        if (rand() < r0s2t)
            D = N * ddn;
        else
            D = N * ddn;
        D = D * 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, co
        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, brdfPo
        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: $\cos \theta_t = \sqrt{1 - \left(\frac{n_1}{n_2} \sin \theta_i \right)^2}$



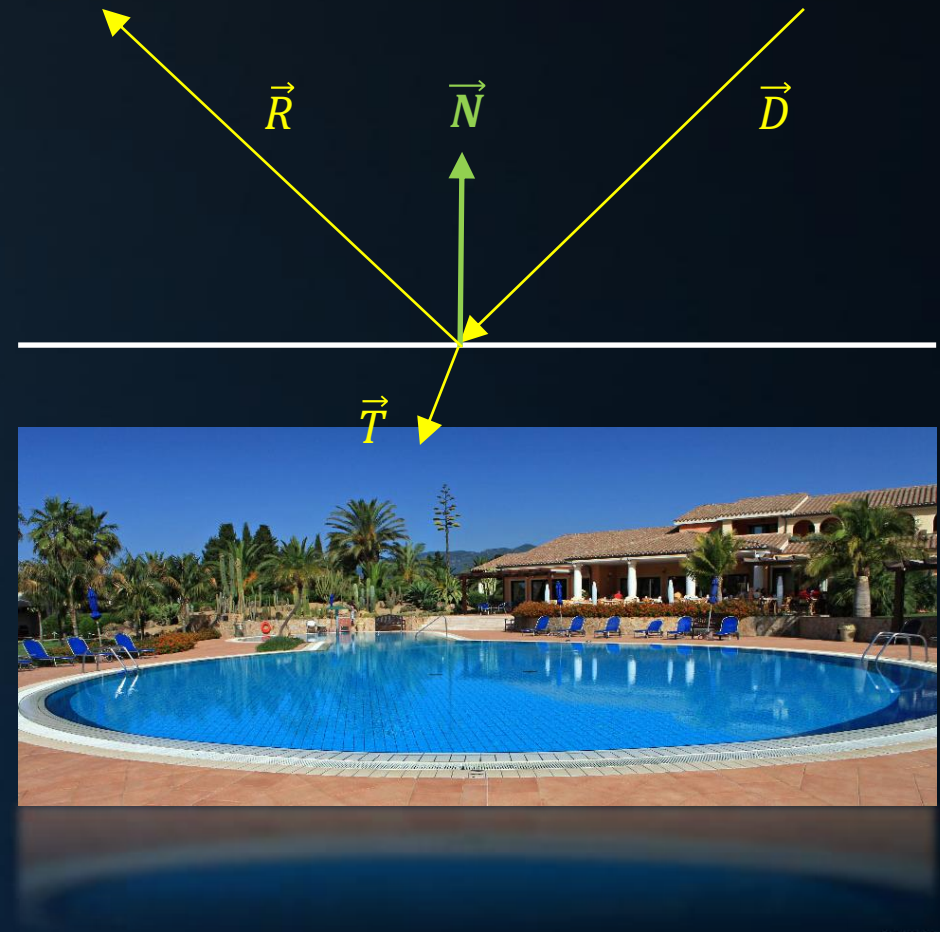
Whitted

Dielectrics

$$F_r = \dots$$

Based on the law of conservation of energy:

$$F_t = 1 - F_r$$



Whitted

Ray Tracing

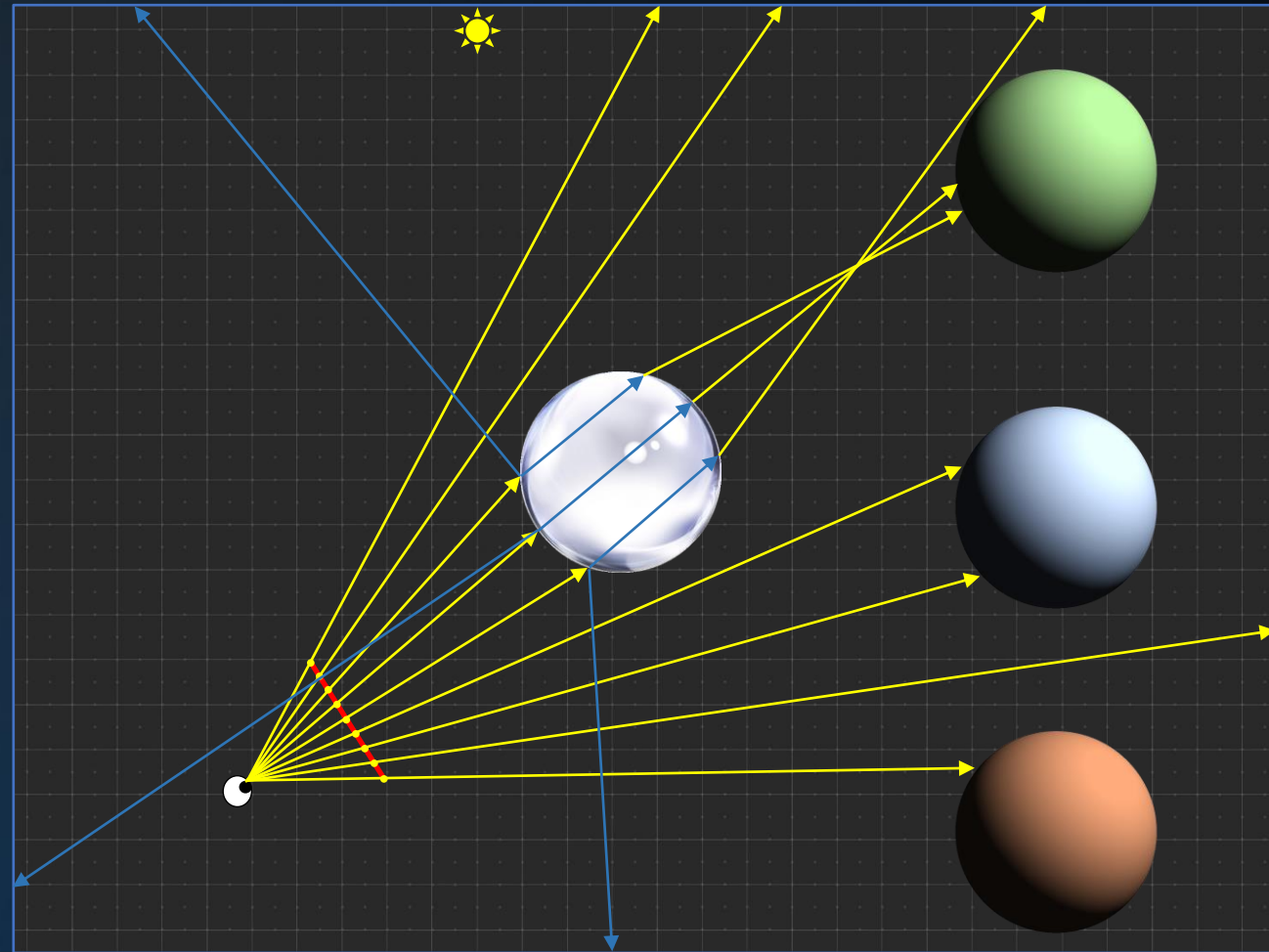
World space

- 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?

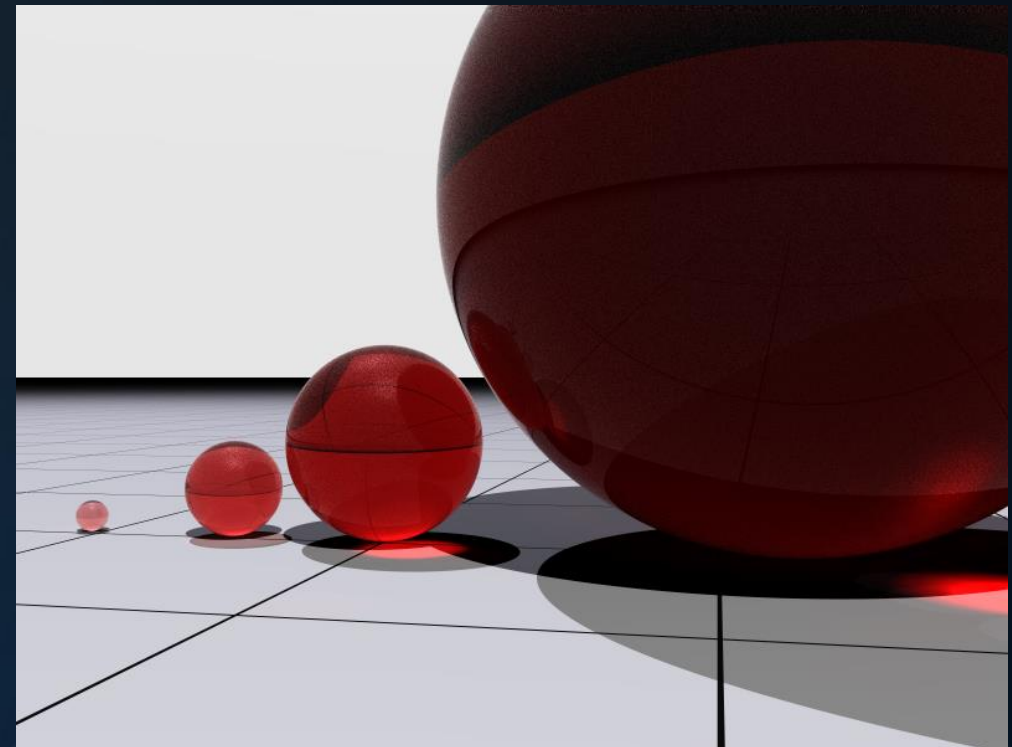
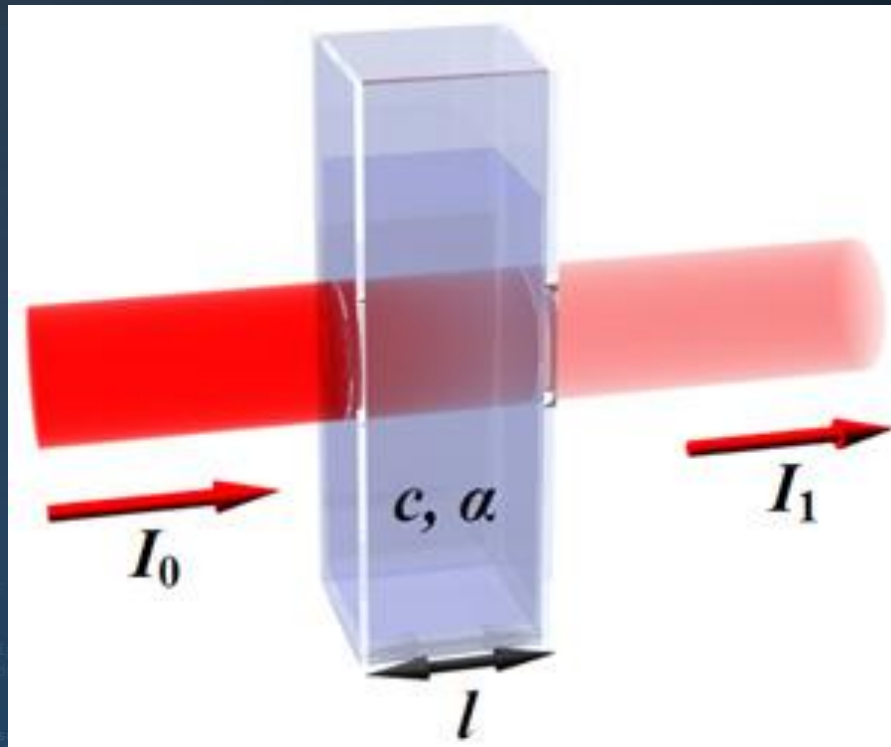
```

1013
1014 & (depth < MAXDEPTH)) {
1015     if (nt < inside) { 1.0f - nt; } else {
1016         nt = nt / nc; ddn = ddn * nc;
1017         nnt = nt / nc; ddn = ddn * nc;
1018         cos2t = 1.0f - nnt; nnt = nnt * nc;
1019         D, N );
1020     }
1021     }
1022     at a = nt - nc, b = nt + nc;
1023     at Tr = 1 - (R0 + (1 - R0) * a);
1024     Tr) R = (D * nnt - N * (ddn * a + b));
1025     }
1026     E * diffuse;
1027     = true;
1028     -
1029     refl + refr)) && (depth < MAXDEPTH)) {
1030     D, N );
1031     refl * E * diffuse;
1032     = true;
1033     -
1034     MAXDEPTH)
1035     survive = SurvivalProbability( diffuse, N );
1036     estimation - doing it properly, closely following
1037     df;
1038     radiance = SampleLight( &rand, I, &L, &lightDir,
1039     e.x + radiance.y + radiance.z) > 0) && (out.x
1040     w = true;
1041     at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
1042     at3 factor = diffuse * INVPI;
1043     at weight = Mis2( directPdf, brdfPdf );
1044     at cosThetaOut = dot( N, L );
1045     E * ((weight * cosThetaOut) / directPdf) * (radiance
1046     random walk - done properly, closely following Seelye
1047     vive)
1048     ;
1049     at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
1050     survive;
1051     pdf;
1052     n = E * brdf * (dot( N, R ) / pdf);
1053     sion = true;

```

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:

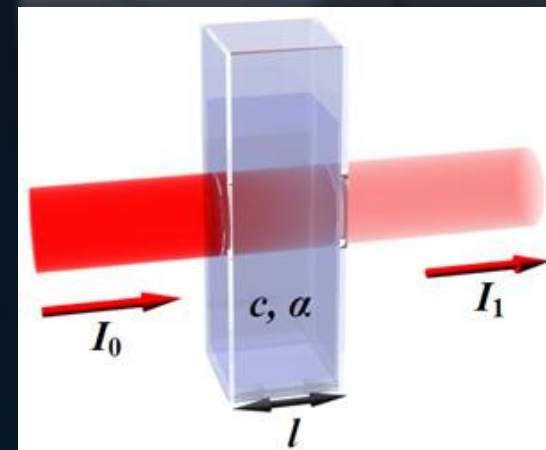
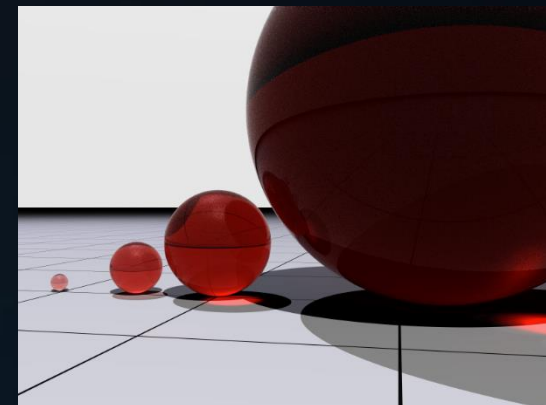
$$I(d) = I(0)e^{-\ln(a)d}$$

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

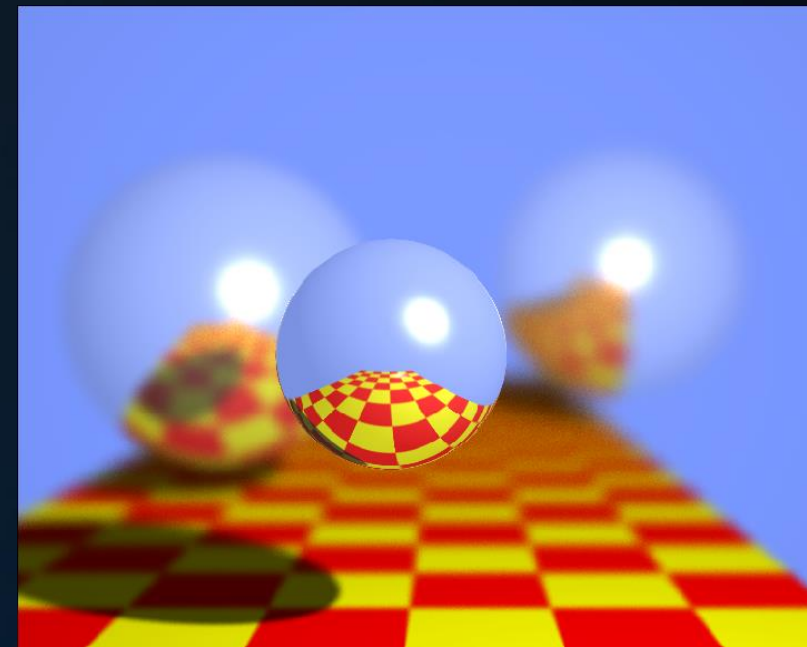
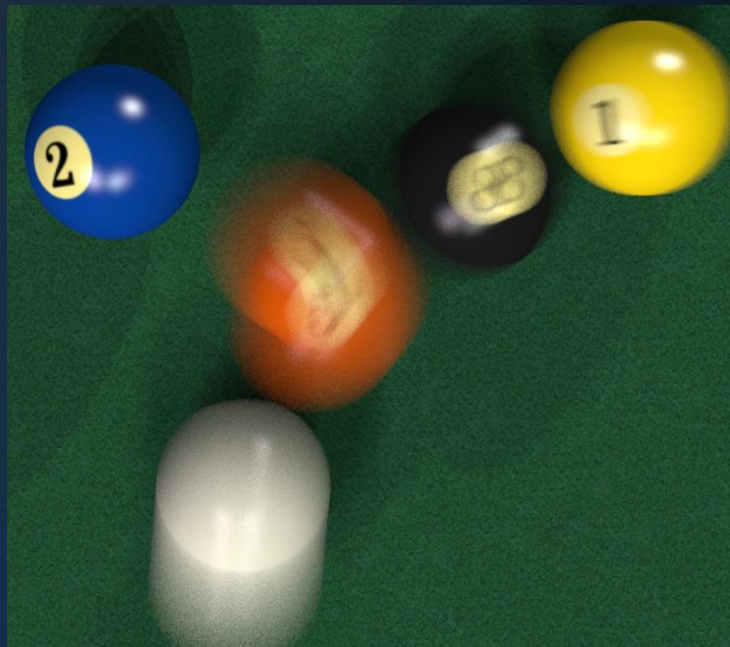
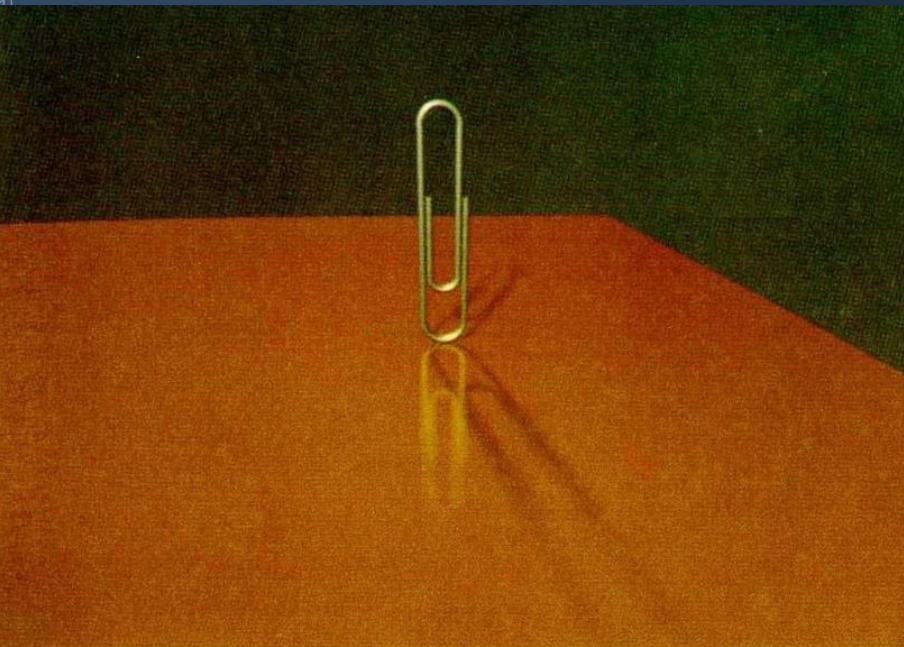


Cook

'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 * ddn;
    rcos2t = 1.0f - nnt * rnd;
    t = 0, 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 2021 – February 2022



END of “Whitted”

next lecture: “Acceleration Structures”

