University of BRISTOL

COMS 30115

Rendering: Raytracing

Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

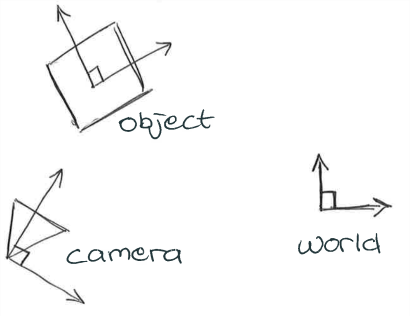
February 11, 2019

http://www.carlhenrik.com

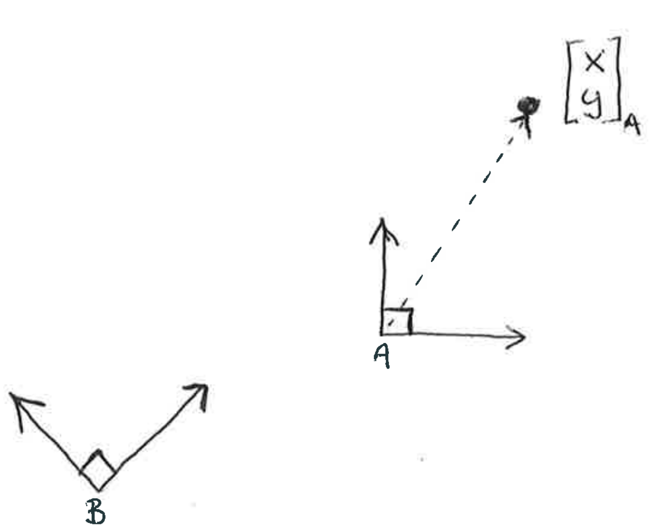
Today

- View Transformations (recap)
- Visibility Problem (last time)
- Lighting

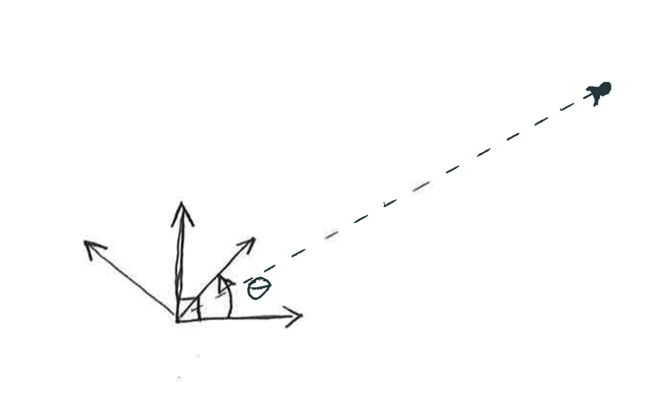
View Transformations



View Transformations

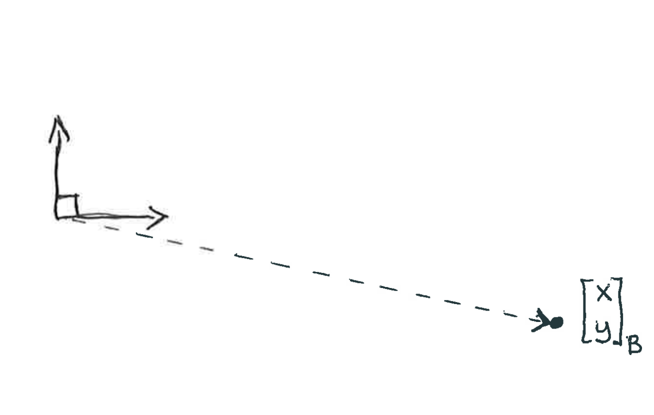


View Transformations



Translate

View Transformations



Rotate negative

The Book

Raytracing

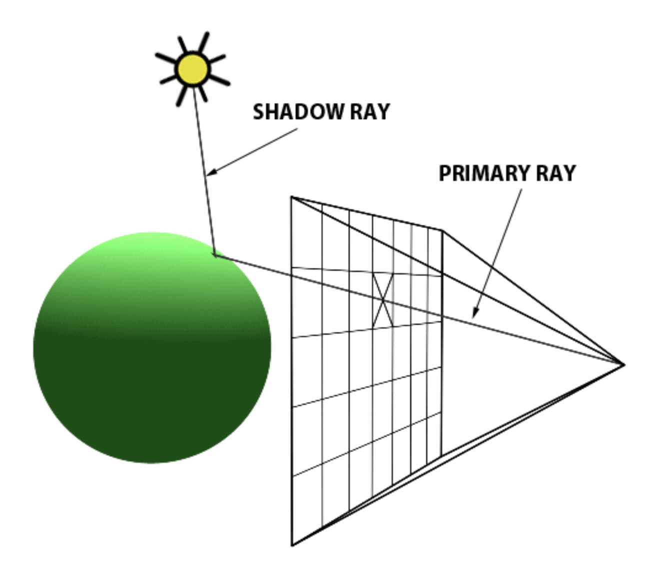
- Introduction to Ray Tracing: a Simple Method for Creating 3D Images
- An Overview of the Ray-Tracing Rendering Technique

Light

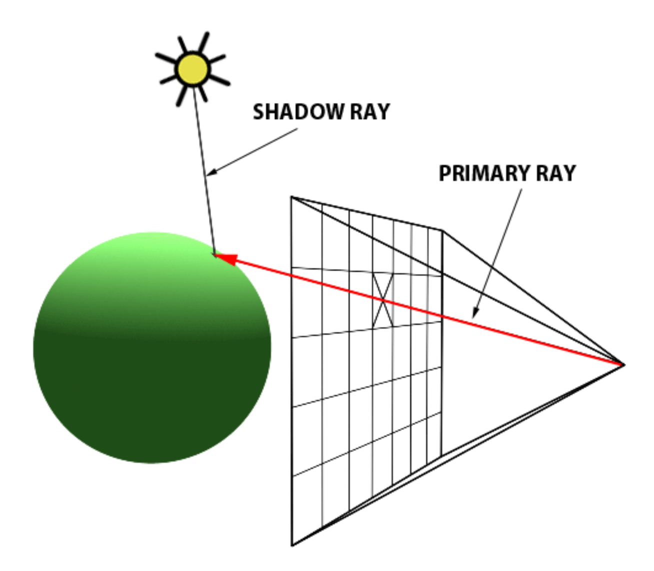
- Surface Properties
- Light
- Reflection and Refraction
- Any old physics book is also a good place to read

Raytracing

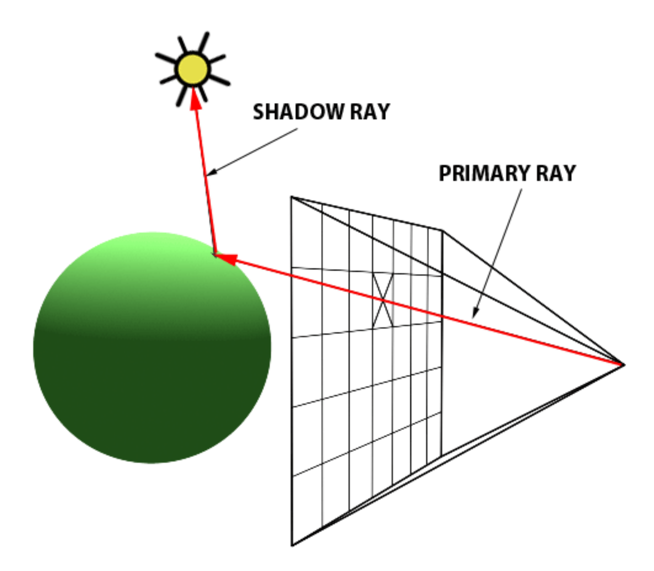
Raytracing



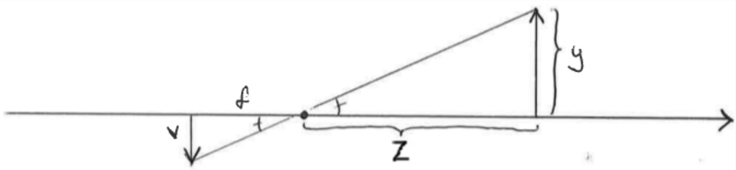
Raytracing



Raytracing

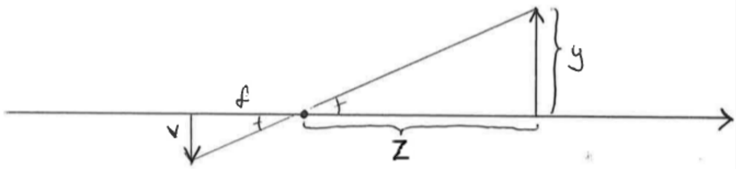


Pinhole camera



- Simplest form camera, where one ray of light lits up each pixel
- Defined by a *location*, *focal length*, *view-direction* and *up-direction*

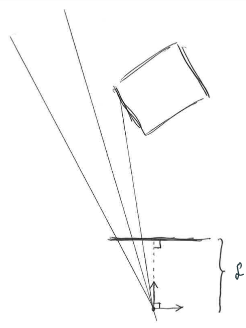
Pinhole camera



$$\frac{v}{f} = \frac{y}{z}$$
$$\Rightarrow v = \frac{f}{z} y$$

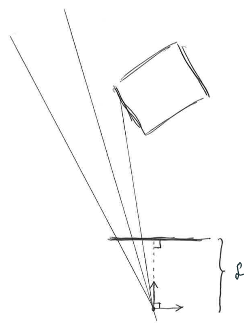
Raycasting

1. Camera at $[0, 0, 0]^{\mathrm{T}}$



Raycasting

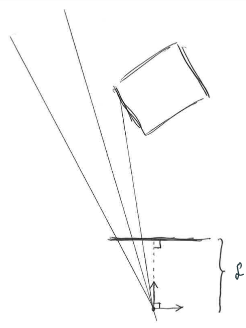
1. Camera at $[0, 0, 0]^{\mathrm{T}}$
2. View direction along z-axis $[0, 0, 1]^{\mathrm{T}}$



Raycasting

1. Camera at $[0,0,0]^{\mathrm{T}}$
2. View direction along z-axis $[0,0,1]^{\mathrm{T}}$
3. Image plane

$$\left\{[u,v,f]^{\mathrm{T}} \middle| -\frac{W}{2} \leq u < \frac{W}{2}, -\frac{H}{2} < v < \frac{H}{2}\right\}$$



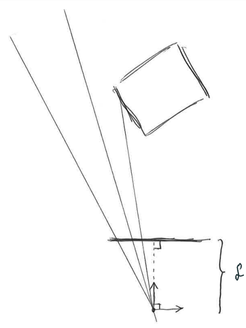
Raycasting

1. Camera at $[0, 0, 0]^{\mathrm{T}}$
2. View direction along z-axis $[0, 0, 1]^{\mathrm{T}}$
3. Image plane

$$\left\{[u, v, f]^{\mathrm{T}} | -\frac{W}{2} \leq u < \frac{W}{2}, -\frac{H}{2} < v < \frac{H}{2}\right\}$$

4. Compute normalised vector from origin to image-plane

$$\mathbf{d}_{ij} = \frac{1}{\sqrt{[u_i, v_j, f][u_i, v_j, f]^{\mathrm{T}}}}[u_i, v_j, f]^{\mathrm{T}}$$



Raycasting

1. Camera at $[0,0,0]^{\mathrm{T}}$
2. View direction along z-axis $[0,0,1]^{\mathrm{T}}$
3. Image plane

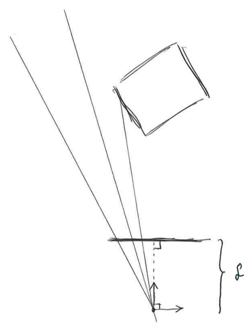
$$\left\{[u,v,f]^{\mathrm{T}} \middle| -\frac{W}{2} \leq u < \frac{W}{2}, -\frac{H}{2} < v < \frac{H}{2}\right\}$$

4. Compute normalised vector from origin to image-plane

$$\mathbf{d}_{ij} = \frac{1}{\sqrt{[u_i, v_j, f][u_i, v_j, f]^{\mathrm{T}}}}[u_i, v_j, f]^{\mathrm{T}}$$

5. Parametrise Ray

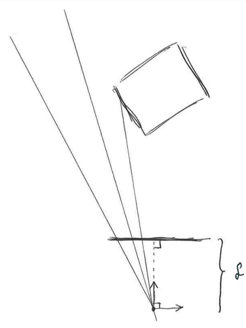
$$\mathbf{r}_{ij} = \mathbf{s} + t \cdot \mathbf{d}_{ij}$$



Raycasting

1. Camera at $[0, 0, 0]^{\mathrm{T}}$
2. View direction along z-axis $[0, 0, 1]^{\mathrm{T}}$
3. Image plane

$$\left\{[u, v, f]^{\mathrm{T}} | -\frac{W}{2} \leq u < \frac{W}{2}, -\frac{H}{2} < v < \frac{H}{2}\right\}$$



4. Compute normalised vector from origin to image-plane

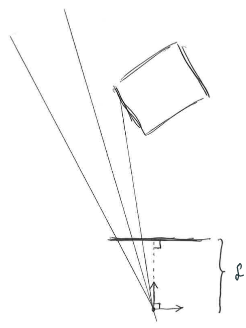
$$\mathbf{d}_{ij} = \frac{1}{\sqrt{[u_i, v_j, f][u_i, v_j, f]^{\mathrm{T}}}}[u_i, v_j, f]^{\mathrm{T}}$$

5. Parametrise Ray

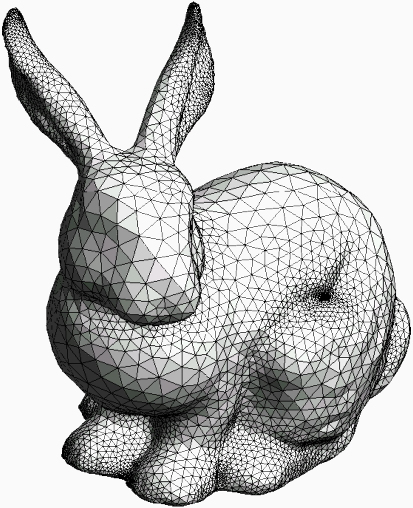
$$\mathbf{r}_{ij} = \mathbf{s} + t \cdot \mathbf{d}_{ij}$$

Raycasting

- Trace a ray for each pixel
- Compute intersection with all objects in world
- Plot colour of intersecting surface
- No "lighting" often called Raycaster



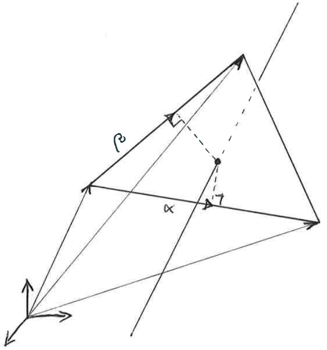
Intersections



Ray-Triangel Intersection

- Compute two vectors in plane

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$$
$$\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$$



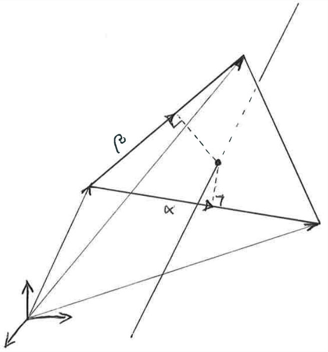
Ray-Triangel Intersection

- Compute two vectors in plane

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$$
$$\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$$

- All points on plane that triangle lies in

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$



Ray-Triangel Intersection

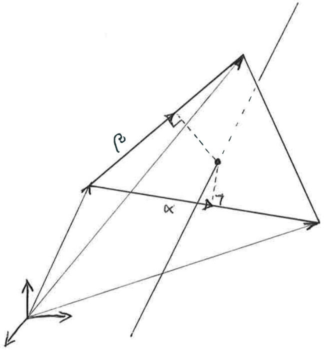
- Compute two vectors in plane

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$$
$$\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$$

- All points on plane that triangle lies in

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Two degrees-of-freedom



Ray-Triangel Intersection

- Compute two vectors in plane

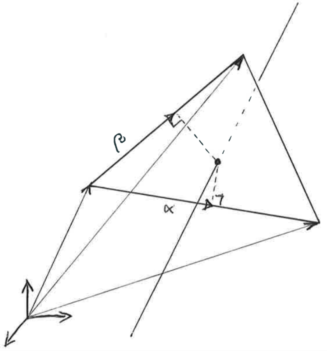
$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$$
$$\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$$

- All points on plane that triangle lies in

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Two degrees-of-freedom
- Inside triangle

$$\{u, v | u \geq 0, v \geq 0, v + u \leq 1\}$$



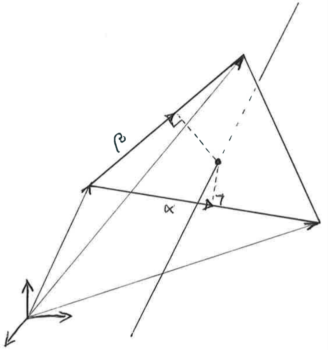
Ray-Triangel Intersection

- Solve for intersection

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

$$\mathbf{r}_{ij} = \mathbf{s} + t \cdot \mathbf{d}_{ij}$$

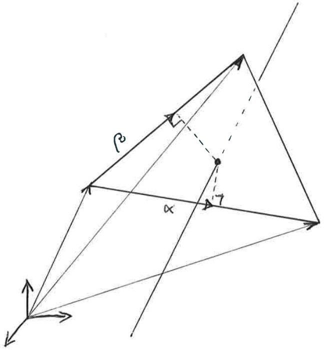
$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$



Ray-Triangel Intersection

- Solve for intersection

$$
\begin{aligned}
\mathbf{r} &= \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1 \\
\mathbf{r}_{ij} &= \mathbf{s} + t \cdot \mathbf{d}_{ij} \\
\mathbf{s} + t \cdot \mathbf{d}_{ij} &= \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1
\end{aligned}
$$



- Unknown: $[u, v, t]^{\mathrm{T}} \in \mathbb{R}^3$

Ray-Triangel Intersection

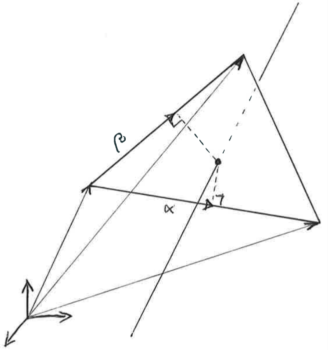
- Solve for intersection

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

$$\mathbf{r}_{ij} = \mathbf{s} + t \cdot \mathbf{d}_{ij}$$

$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Unknown: $[u, v, t]^{\mathrm{T}} \in \mathbb{R}^3$
- Three equations three unknowns!

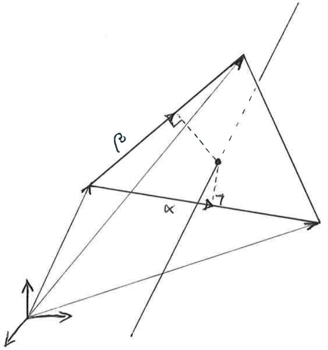


Ray-Triangel Intersection

- Solve for intersection

$$\mathbf{r} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$
$$\mathbf{r}_{ij} = \mathbf{s} + t \cdot \mathbf{d}_{ij}$$
$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Unknown: $[u, v, t]^{\mathrm{T}} \in \mathbb{R}^3$
- Three equations three unknowns!
- Intersection of plane

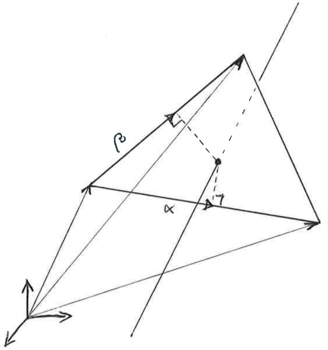


Ray-Triangel Intersection

- Solve for intersection

$$
\begin{aligned}
\mathbf{r} &= \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1 \\
\mathbf{r}_{ij} &= \mathbf{s} + t \cdot \mathbf{d}_{ij} \\
\mathbf{s} + t \cdot \mathbf{d}_{ij} &= \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1
\end{aligned}
$$

- Unknown: $[u, v, t]^{\mathrm{T}} \in \mathbb{R}^3$
- Three equations three unknowns!
- Intersection of plane
- Check boundary conditions of triangle



Ray-Triangle Intersection

- Ray in plane equation

$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$
$$s - \mathbf{v}_0 = -t \cdot \mathbf{d}_{ij} + u\mathbf{e}_0 + v\mathbf{e}_1$$

Ray-Triangle Intersection

- Ray in plane equation

$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

$$s - \mathbf{v}_0 = -t \cdot \mathbf{d}_{ij} + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Write on matrix form

$$\begin{bmatrix} -d_{ij}^x & e_0^x & e_1^x \\ -d_{ij}^y & e_0^y & e_1^y \\ -d_{ij}^z & e_0^z & e_1^z \end{bmatrix} \cdot \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} s^x - v_0^x \\ s^y - v_0^y \\ s^z - v_0^z \end{bmatrix}$$

Ray-Triangle Intersection

- Ray in plane equation

$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

$$s - \mathbf{v}_0 = -t \cdot \mathbf{d}_{ij} + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Write on matrix form

$$\begin{bmatrix} -d_{ij}^x & e_0^x & e_1^x \\ -d_{ij}^y & e_0^y & e_1^y \\ -d_{ij}^z & e_0^z & e_1^z \end{bmatrix} \cdot \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} s^x - v_0^x \\ s^y - v_0^y \\ s^z - v_0^z \end{bmatrix}$$

- This we know from basic linear algebra as

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Ray-Triangle Intersection

- Ray in plane equation

$$\mathbf{s} + t \cdot \mathbf{d}_{ij} = \mathbf{v}_0 + u\mathbf{e}_0 + v\mathbf{e}_1$$

$$s - \mathbf{v}_0 = -t \cdot \mathbf{d}_{ij} + u\mathbf{e}_0 + v\mathbf{e}_1$$

- Write on matrix form

$$\begin{bmatrix} -d_{ij}^x & e_0^x & e_1^x \\ -d_{ij}^y & e_0^y & e_1^y \\ -d_{ij}^z & e_0^z & e_1^z \end{bmatrix} \cdot \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} s^x - v_0^x \\ s^y - v_0^y \\ s^z - v_0^z \end{bmatrix}$$

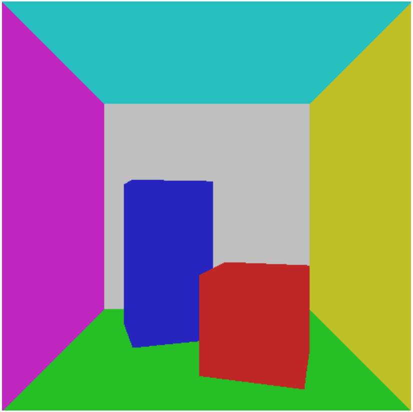
- This we know from basic linear algebra as

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

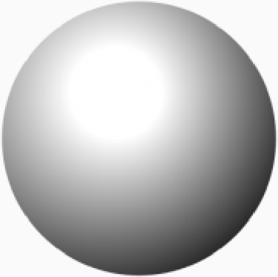
- *silliest solution*

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Intersections



Sphere



Ray-Sphere Intersection

- A point $\mathbf{p}$ lies the surface of a sphere with radius r and center $\mathbf{c}$ iff

$$\sqrt{(\mathbf{p}-\mathbf{c})^{\mathrm{T}}(\mathbf{p}-\mathbf{c})} - r = 0$$

Ray-Sphere Intersection

- A point $\mathbf{p}$ lies the surface of a sphere with radius r and center $\mathbf{c}$ iff

$$\sqrt{(\mathbf{p}-\mathbf{c})^{\mathrm{T}}(\mathbf{p}-\mathbf{c})}-r=0$$

- Insert ray-equation into sphere

$$\begin{aligned}\sqrt{(\mathbf{s}+t\mathbf{d}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}+t\mathbf{d}-\mathbf{c})}-r&=0\\(\mathbf{s}+t\mathbf{d}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}+t\mathbf{d}-\mathbf{c})&=r^2\\t^2+2t(\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c}))+(\mathbf{s}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}-\mathbf{c})-r^2&=0\end{aligned}$$

Ray-Sphere Intersection

- A point $\mathbf{p}$ lies the surface of a sphere with radius r and center $\mathbf{c}$ iff

$$\sqrt{(\mathbf{p}-\mathbf{c})^{\mathrm{T}}(\mathbf{p}-\mathbf{c})} - r = 0$$

- Insert ray-equation into sphere

$$\begin{aligned} \sqrt{(\mathbf{s}+t\mathbf{d}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}+t\mathbf{d}-\mathbf{c})} - r &= 0 \\ (\mathbf{s}+t\mathbf{d}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}+t\mathbf{d}-\mathbf{c}) &= r^2 \\ t^2 + 2t(\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c})) + (\mathbf{s}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}-\mathbf{c}) - r^2 &= 0 \end{aligned}$$

- Quadratic expression i t

$$t = -\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c}) \pm \sqrt{(\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c}))^2 - (\mathbf{s}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}-\mathbf{c}) - r^2}$$

Ray Intersections

- A Raytracer spends most its time doing intersection computations (profile your code)
- Rules of Thumb
    - Can we trivially reject something?
    - Can we re-use computations?
- If we need to do several tests do them in order of computational cost
- Realtime

Ray Intersections

- A Raytracer spends most its time doing intersection computations (profile your code)
- Rules of Thumb
  - Can we trivially reject something?
  - Can we re-use computations?
- If we need to do several tests do them in order of computational cost
- Realtime
- *Its really fun optimisations as the code is often very small and the solutions are really cute*

Ray-Sphere Intersection

- Sphere intersection

$$t = -\mathbf{d}^{\mathrm{T}}(\mathbf{s} - \mathbf{c}) \pm \sqrt{\left(\mathbf{d}^{\mathrm{T}}(\mathbf{s} - \mathbf{c})\right)^2 - (\mathbf{s} - \mathbf{c})^{\mathrm{T}}(\mathbf{s} - \mathbf{c}) - r^2}$$

- Look at square-root
  - $= 0$ Only one solution, no need to compute square-root
  - < 0 No real solution, no intersection
  - > 0 We have to compute square root (two intersections)

Ray-Sphere Intersection

- Sphere intersection

$$t = -\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c}) \pm \sqrt{\left(\mathbf{d}^{\mathrm{T}}(\mathbf{s}-\mathbf{c})\right)^2 - (\mathbf{s}-\mathbf{c})^{\mathrm{T}}(\mathbf{s}-\mathbf{c}) - r^2}$$

- Look at square-root
  - $= 0$ Only one solution, no need to compute square-root
  - < 0 No real solution, no intersection
  - > 0 We have to compute square root (two intersections)
- *Which of the above cases do we have most of the time?*

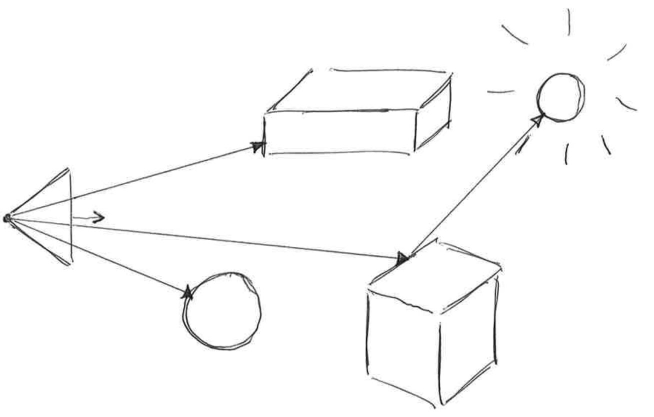
Raycaster

Code

```
for (int y=-h/2;y<h/2;y++)
  {
    for(int x=-w/2;x<w/2;x++)
      {
        /*compute primary ray*/
        for(int i=0;i<N_primitives;i++)
          {
            /*1. compute intersection
              2. check closest*/
          }
      }
  }
```

Surface - Light

Illumination



Appearance

What does appearance depend on?

- surface properties
    - material
    - geometry
    - orientation
- light properties
- viewing direction

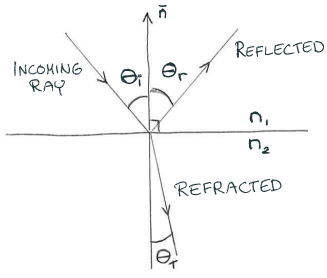
Appearance

What does appearance depend on?

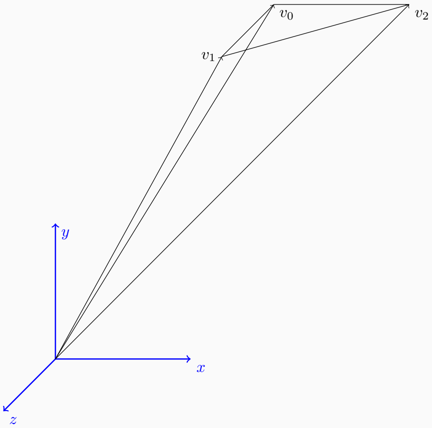
- surface properties
    - material
    - geometry
    - orientation
- light properties
- viewing direction
- *To render we need a mathematical model of these things.*

When light meet surface

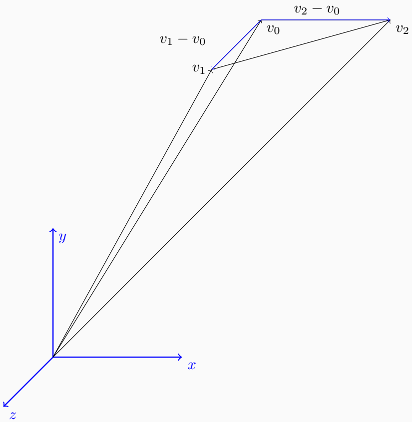
- Reflection & Refraction
- Angles are based on Fermats principle of "least time"
- a light ray will take the path of least time



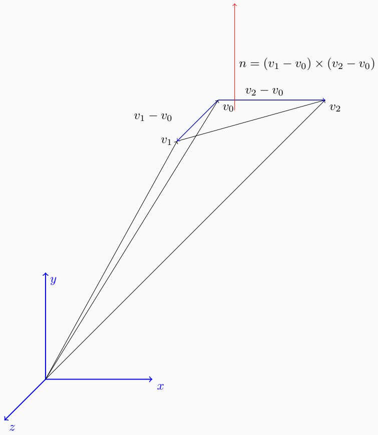
Normals



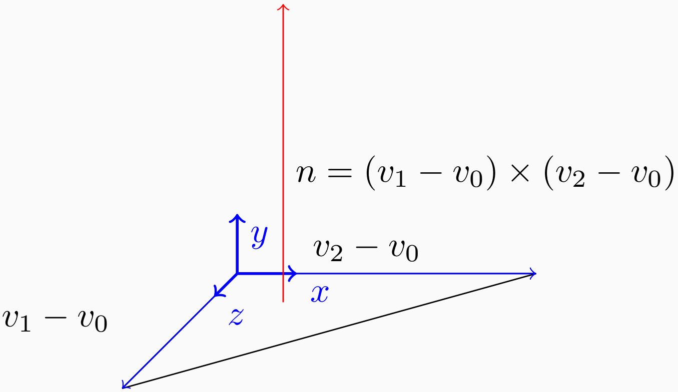
Normals



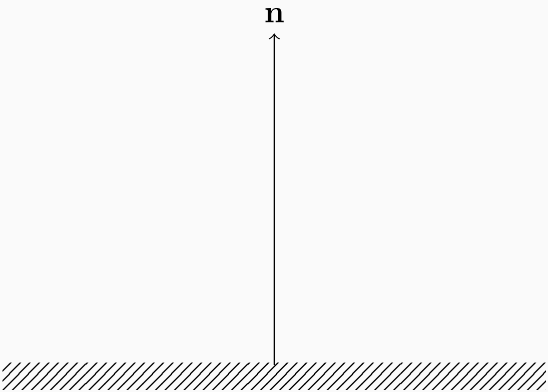
Normals



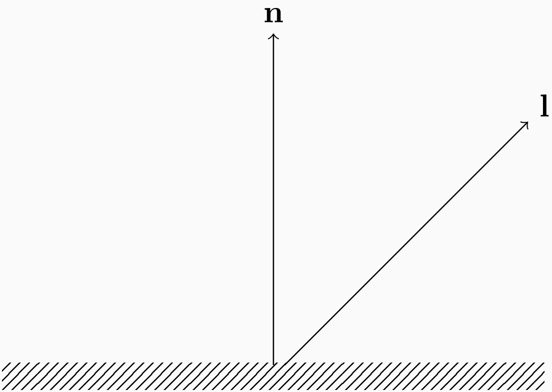
Normals



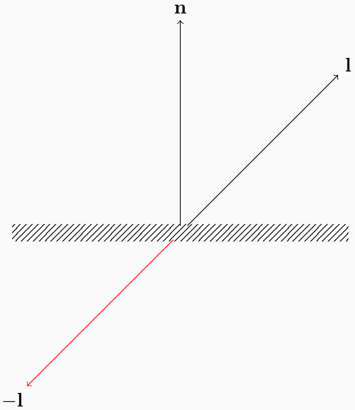
Reflection



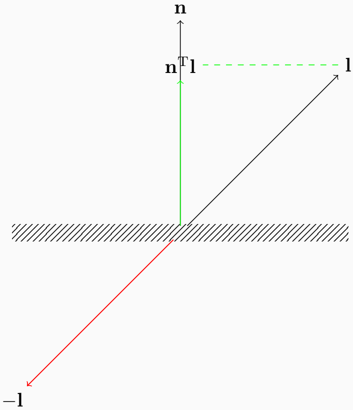
Reflection



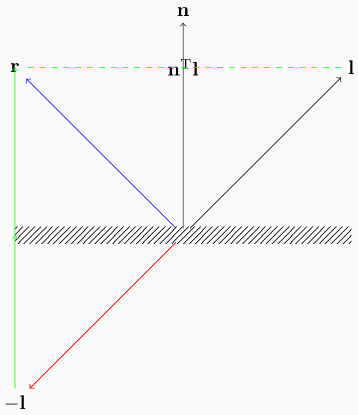
Reflection



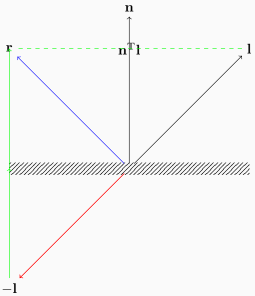
Reflection



Reflection



Reflection



$$\mathbf{r} = 2\left(\mathbf{n}^{\mathrm{T}}\mathbf{l}\right)\mathbf{n} - \mathbf{l}$$

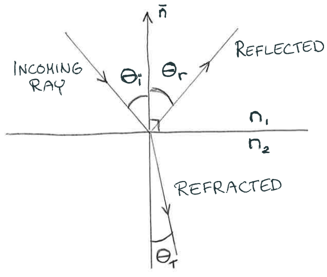
When light meet surface

Refraction

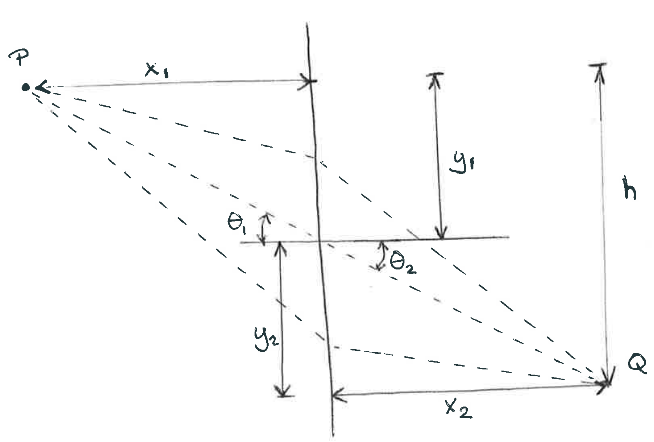
- Snell's Law

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

- Varies with wavelength
- Isotropic (pure) media



Snell's Law



Snell's Law

$$t = \frac{(x_1^2 + y_1^2)^{\frac{1}{2}}}{v_1} + \frac{(x_2^2 + y_2^2)^{\frac{1}{2}}}{v_2}$$

$$y_2 = h - y_1$$

$$\frac{\delta t}{\delta y_1} = \frac{1}{v_1}\frac{y_1}{(x_1^2 + y_1^2)^{\frac{1}{2}}} + \frac{1}{v_2}\frac{-(h - y_1)}{(x_2^2 + (h - y_1)^2)^{\frac{1}{2}}}$$

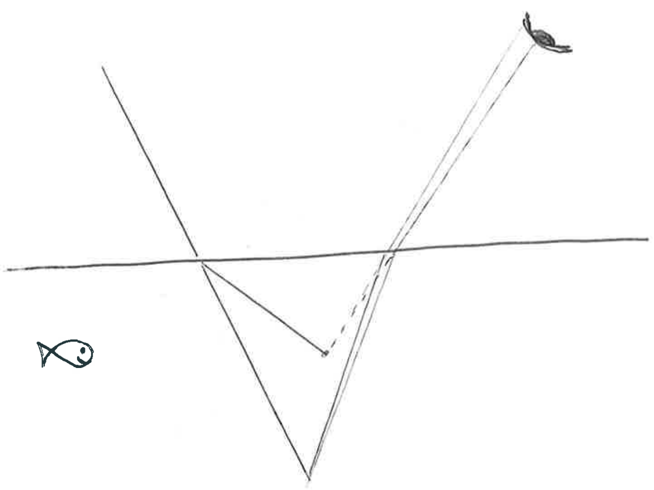
$$\frac{\delta t}{\delta y_1} = \frac{\sin\theta_1}{v_1} - \frac{\sin\theta_2}{v_2} = 0$$

$$n_1 = \frac{c}{v_1}$$

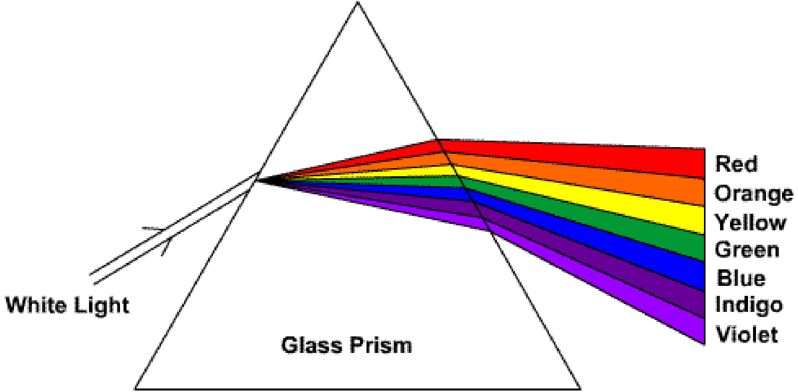
$$\frac{n_1 \sin\theta_1}{c} = \frac{n_2 \sin\theta_2}{c}$$

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

Snell's Law

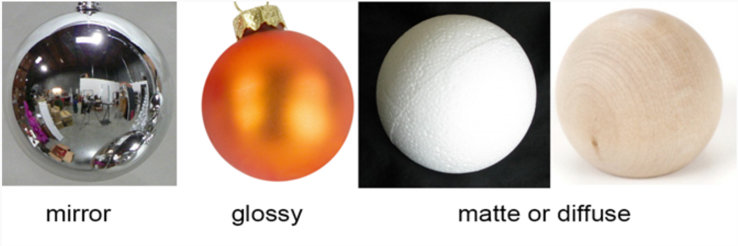


Snell's Law



$$\eta_1(\lambda)\sin(\theta_1) = \eta_2(\lambda)\sin(\theta_2))$$

Surfaces



- Surfaces are usually cathegorised on a continum
- Mirror $\Rightarrow$ Glossy $\Rightarrow$ Diffuse

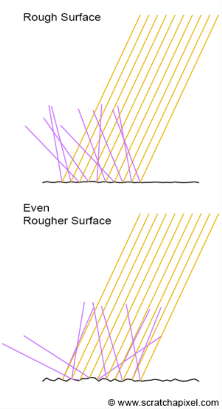
Mirrors [1]

- We only see one ray from each point
- Perfect reflection

[1] Cloud Gate Chicago ("The Egg") Image URL

Glossy Surfaces

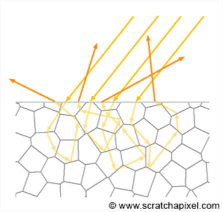
- Surfaces look glossy because of imperfections
- Random surface pertubations



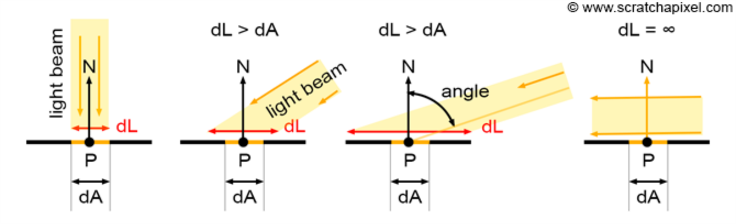
Glossy Surfaces

Diffuse Surfaces

- Often due to internal structures inside object
- Light exits object at angles independent from input angle
- Sometimes we have structured internals such that they are not independent
  - sub-surface scattering (skin)



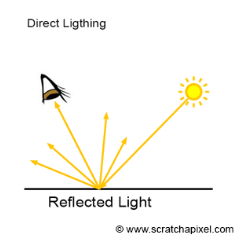
Diffuse Surfaces (foreshortening term)

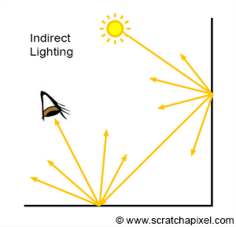


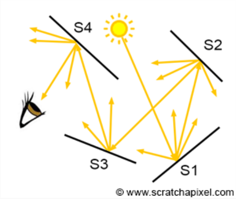
- the surface would get more light once the light is angled
- to avoid this effect, keep the surface const.

$$\mathrm{d}A = \cos(n, l)\mathrm{d}L = n^{\mathrm{T}} l \mathrm{d}L$$

Lights

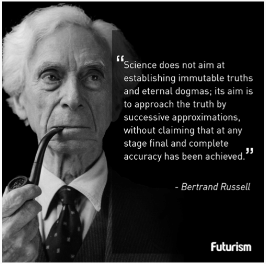

Direct Ligthing
Reflected Light
© www.scratchapixel.com


Indirect
Lighting
© www.scratchapixel.com


S4
S2
S3
S1
© www.scratchapixel.com

Approximations

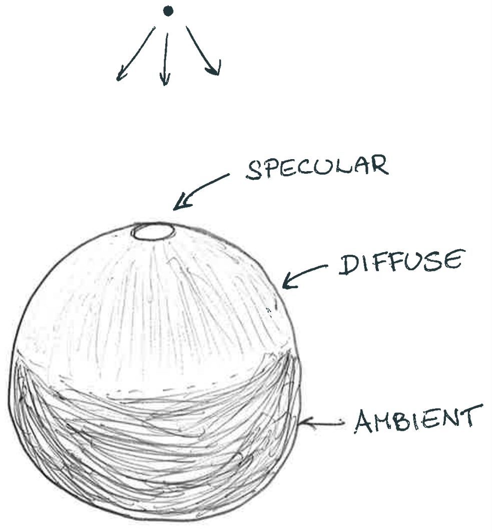
- Lights interaction with a surface is simple in *forward* tracing
- Rather complicated *backward* tracing
- Approximate all lights as different types of direct light



#Science #Truth

Lights

Lights Ball

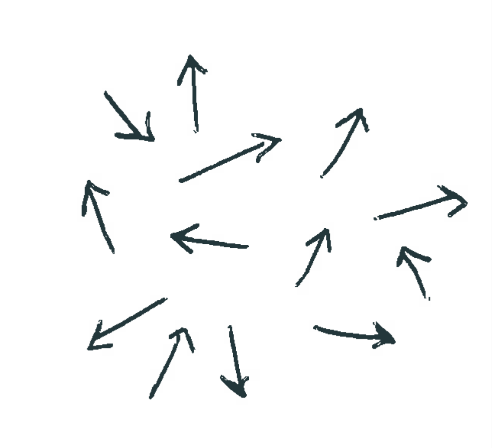


Light Factorisation

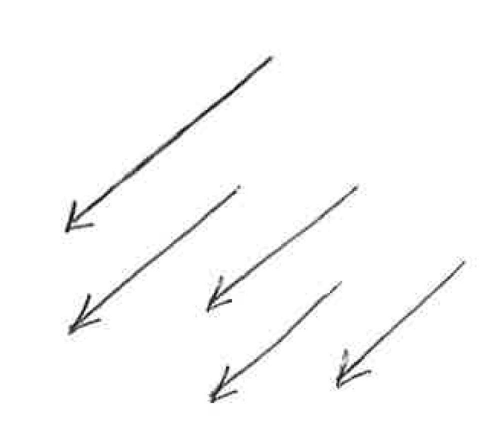
$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

- There is only one type of light
- Approximation: Factorise into Ambient, Diffuse and Specular

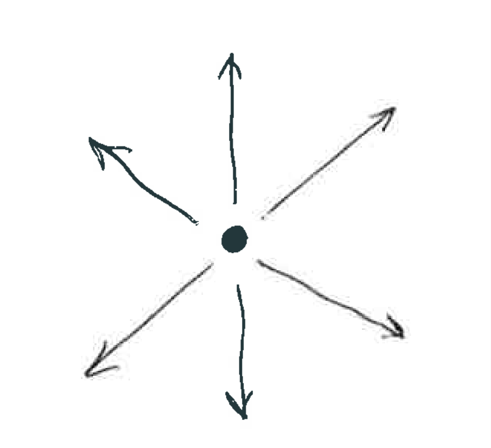
Light Factorisation (Ambient)



Light Factorisation (Diffuse)



Light Factorisation (Point)



Hadamard Product

$$\mathbf{a} = \mathbf{b} \circ \mathbf{c}$$

$$(\mathbf{a})_i = (\mathbf{b})_i \cdot (\mathbf{c})_i$$

Colour

$$\mathbf{i} = \mathbf{m} \circ \mathbf{s}$$

s light intensity
m material properties
i colour of reflected light

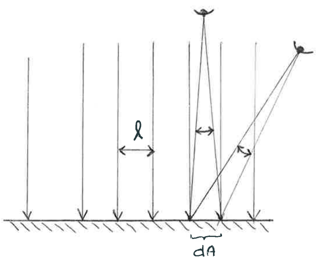
Parametrisation

Material and Light

Notation	Description
$\mathbf{s}_{amb}$	Ambient intensity
$\mathbf{s}_{diff}$	Diffuse intensity
$\mathbf{s}_{spec}$	Specular intensity
$\mathbf{s}_{pos}$	Light source position

Notation	Description
$\mathbf{m}_{amb}$	Ambient material
$\mathbf{m}_{diff}$	Diffuse material
$\mathbf{m}_{spec}$	Specular material
m_{shi}	"Shininess"
$\mathbf{m}_{emi}$	Emitting

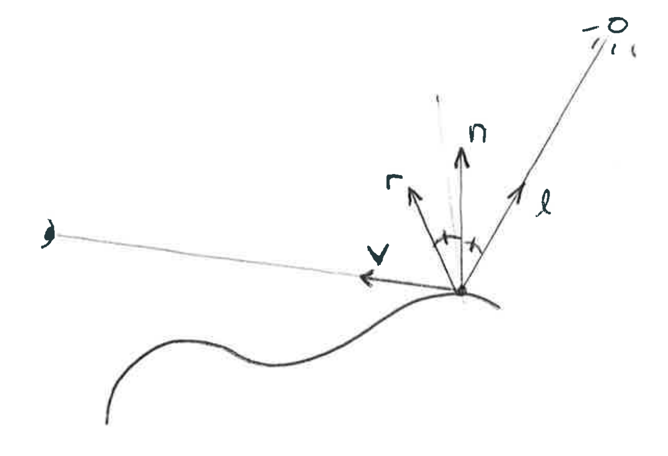
Diffuse/Lambertian



- View independent
- Lambertian Surface (Looks the same from all directions)

$$\mathbf{i}_{diff} = \max\left((0, \mathbf{n}^{\mathrm{T}}\mathbf{l})\right) \mathbf{m}_{diff} \circ \mathbf{s}_{diff}$$

Specular

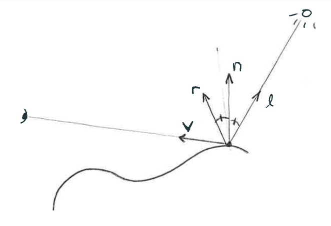


Specular

- View dependent
- Non-linear "highlights"

$\mathbf{r} = 2(\mathbf{n}^{\mathrm{T}}\mathbf{l})\mathbf{n} - \mathbf{l}$

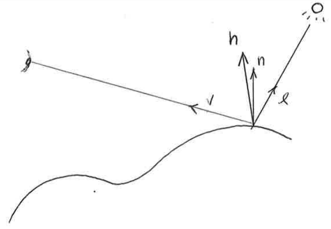
Phong $i_{spec} = (\mathbf{r}^{\mathrm{T}}\mathbf{v})^{m_{shi}}$



Blinn

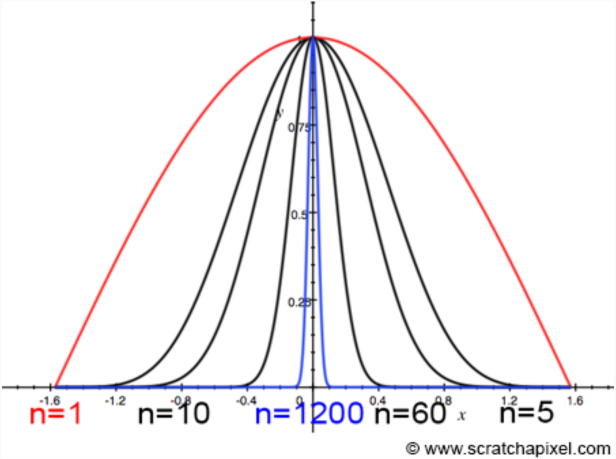
$$i_{spec} = (\mathbf{n}^{\mathrm{T}}\mathbf{h})^{m_{shi}}$$

$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{((\mathbf{l} + \mathbf{v})^{\mathrm{T}}(\mathbf{l} + \mathbf{v}))^{\frac{1}{2}}}$$



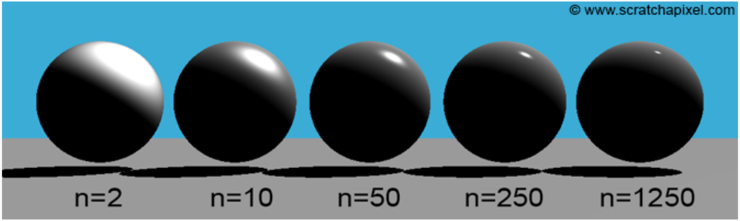
- Specular colour $\mathbf{i}_{spec} = \max((0, i_{spec}))\,\mathbf{m}_{spec} \circ \mathbf{s}_{spec}$

Specular

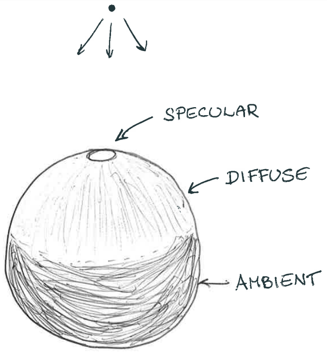


y
0.75
0.5
0.25
-1.6
-1.2
-0.8
-0.4
0
0.4
0.8
1.2
1.6
x
n=1
n=10
n=1200
n=60
n=5
© www.scratchapixel.com

Specular



Ambient



- Accounts for indirect light
- Not particularly realistic

$$\mathbf{i}_{amb} = \mathbf{m}_{amb} \circ \mathbf{s}_{amb}$$

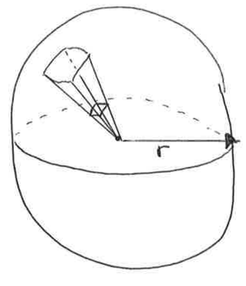
Distance Attenuation

$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

- Law of conservation of matter: $A = 4\pi r^2$
- Distance attenuation

$$d = \left(s_c + s_l \cdot r + s_q \cdot r^2\right)^{-1}$$
$$r = \left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}}$$

Distance Attenuation



Distance Attenuation

$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

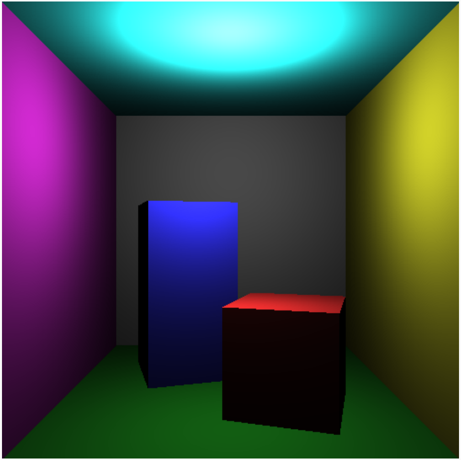
- Law of conservation of matter: $A = 4\pi r^2$
- Distance attenuation

$$d = \left(s_c + s_l \cdot r + s_q \cdot r^2\right)^{-1}$$
$$r = \left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}}$$

All Together

$$
\begin{aligned}
\mathbf{i}_{tot} &= f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec}) \\
&= \mathbf{m}_{emi} + \sum_{i=0}^{N-1} \Big(\mathbf{m}_{amb} \circ \mathbf{s}^{i}_{amb} \\
&+ \frac{\max((\mathbf{n}^{\mathrm{T}}\mathbf{l}^{i}), 0)\mathbf{m}_{diff} \circ \mathbf{s}^{i}_{diff} + \max((\mathbf{n}^{\mathrm{T}}\mathbf{h}^{i}), 0)^{m_{shi}} \mathbf{m}_{spec} \circ \mathbf{s}^{i}_{spec}}{s^{i}_{c} + s^{i}_{l}\left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}} + s^{i}_{q}\left((\mathbf{s}^{i}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}^{i}_{pos} - \mathbf{p})\right)^{\frac{1}{2}}} \Bigg)
\end{aligned}
$$

Lab



Code

```
/*Per Pixel*/
/*compute primary ray*/
for(int i=0;i<N_primitives;i++)
  {
    /*1. compute intersection
      2. check closest*/
  }
for(int i=0;i<N_lights;i++)
  {
    /*compute light*/
  }
```

eof