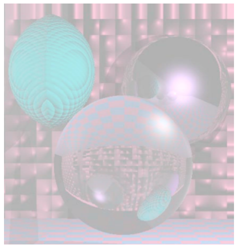
CS G140
Graduate Computer Graphics

Prof. Harriet Fell

Spring 2009

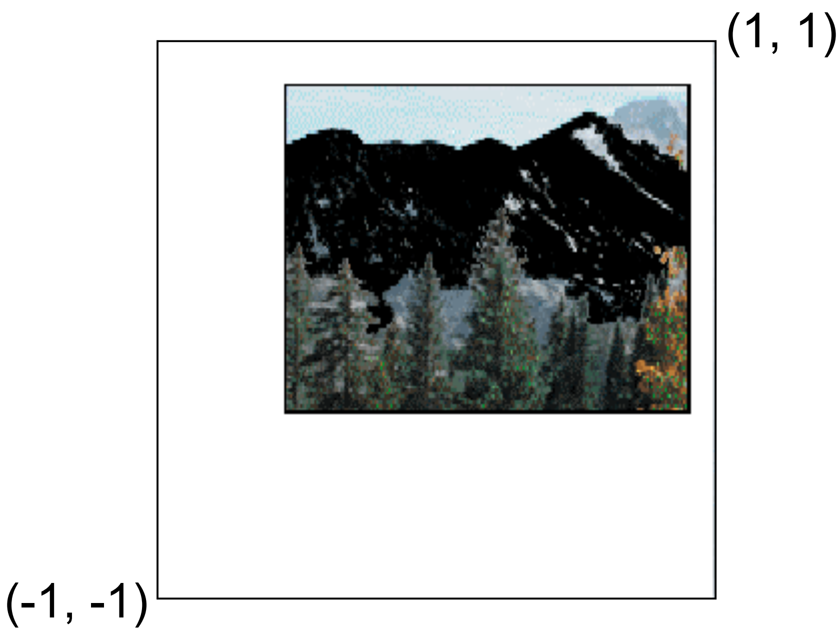
Lecture 3 - January 21, 2009



Today's Topics

- From 3D to 2D
- 2 Dimensional Viewing Transformation
  http://www.siggraph.org/education/materials/HyperGraph/viewing/view2d/2dview0.htm
- Viewing – from Shirley *et al.* Chapter 7
- Recursive Ray Tracing
  - Reflection
  - Refraction

Scene is from my photo of Estes Park – Harriet Fell
Kitchen window from http://www.hoagy.org/cityscape/graphics/cityscapeAtNight.jpg


(1, 1)
(-1, -1)

from a 3D World to a 2D Screen

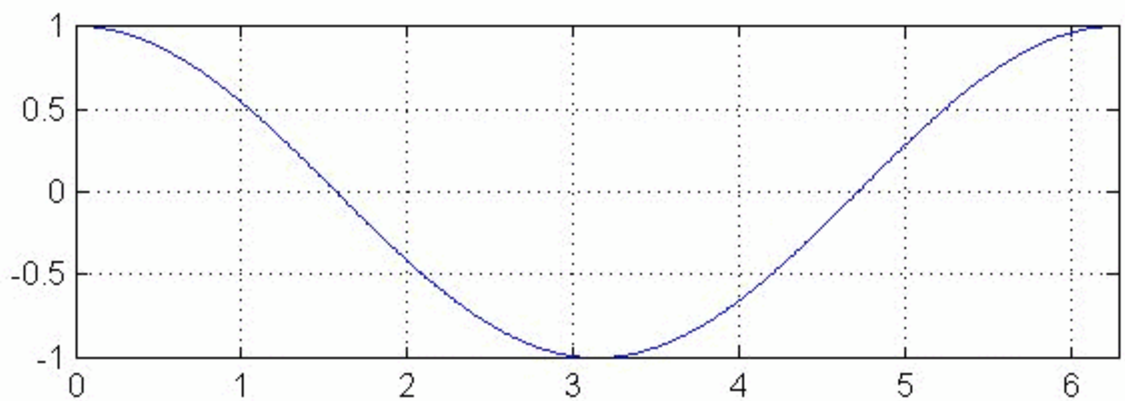
When we define an image in some world coordinate system, to display that image we must somehow map the image to the physical output device.

1. Project 3D world down to a 2D window (WDC).
2. Transform WDC to a Normalized Device Coordinates Viewport (NDC).
3. Transform (NDC) to 2D physical device coordinates (PDC).

2 Dimensional Viewing Transformation

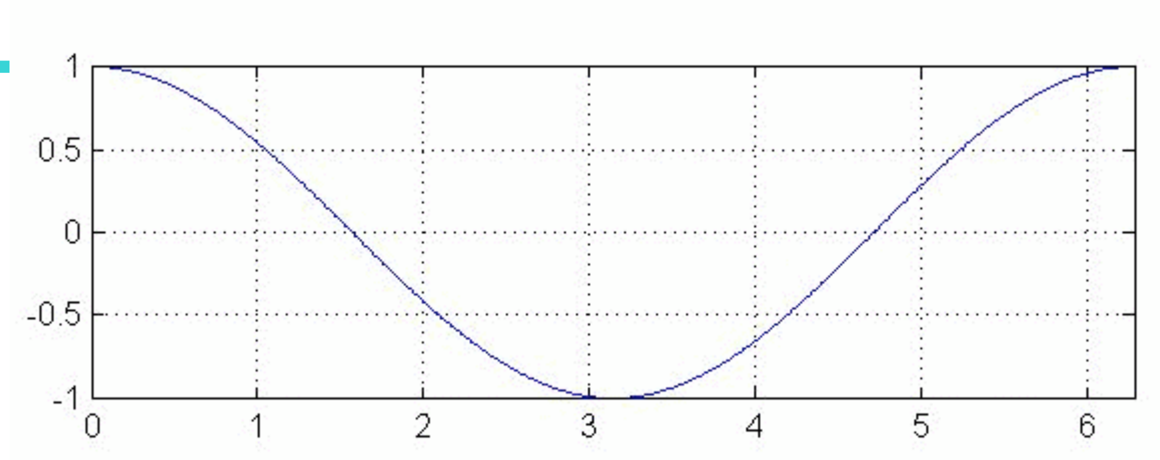
http://www.siggraph.org/education/materials/HyperGraph/viewing/view2d/2dview0.htm

- Window
  - Example: Want to plot x vs. cos(x) for x between 0.0 and 2Pi. The values of cos x will be between -1.0 and +1.0. So we want the window as shown here.

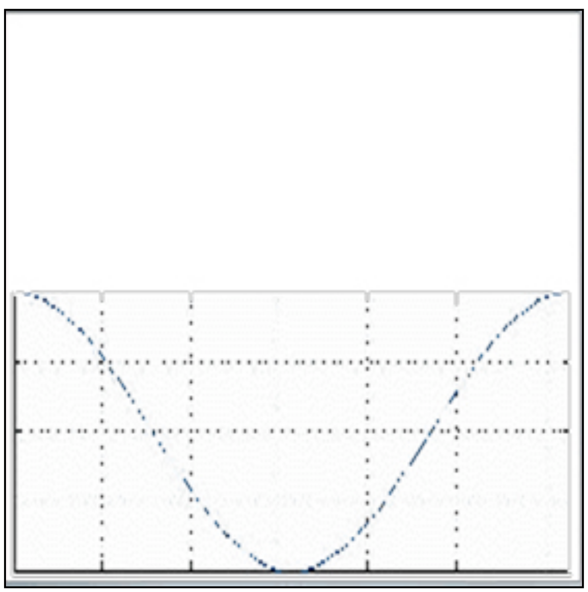


2D Viewing Transformation

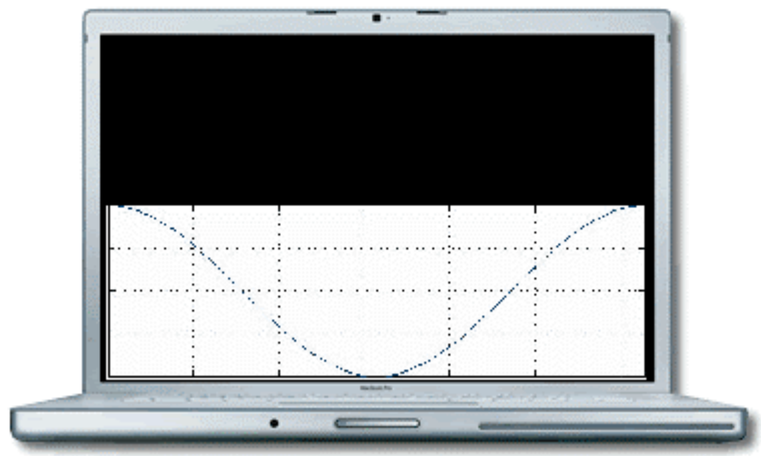
World



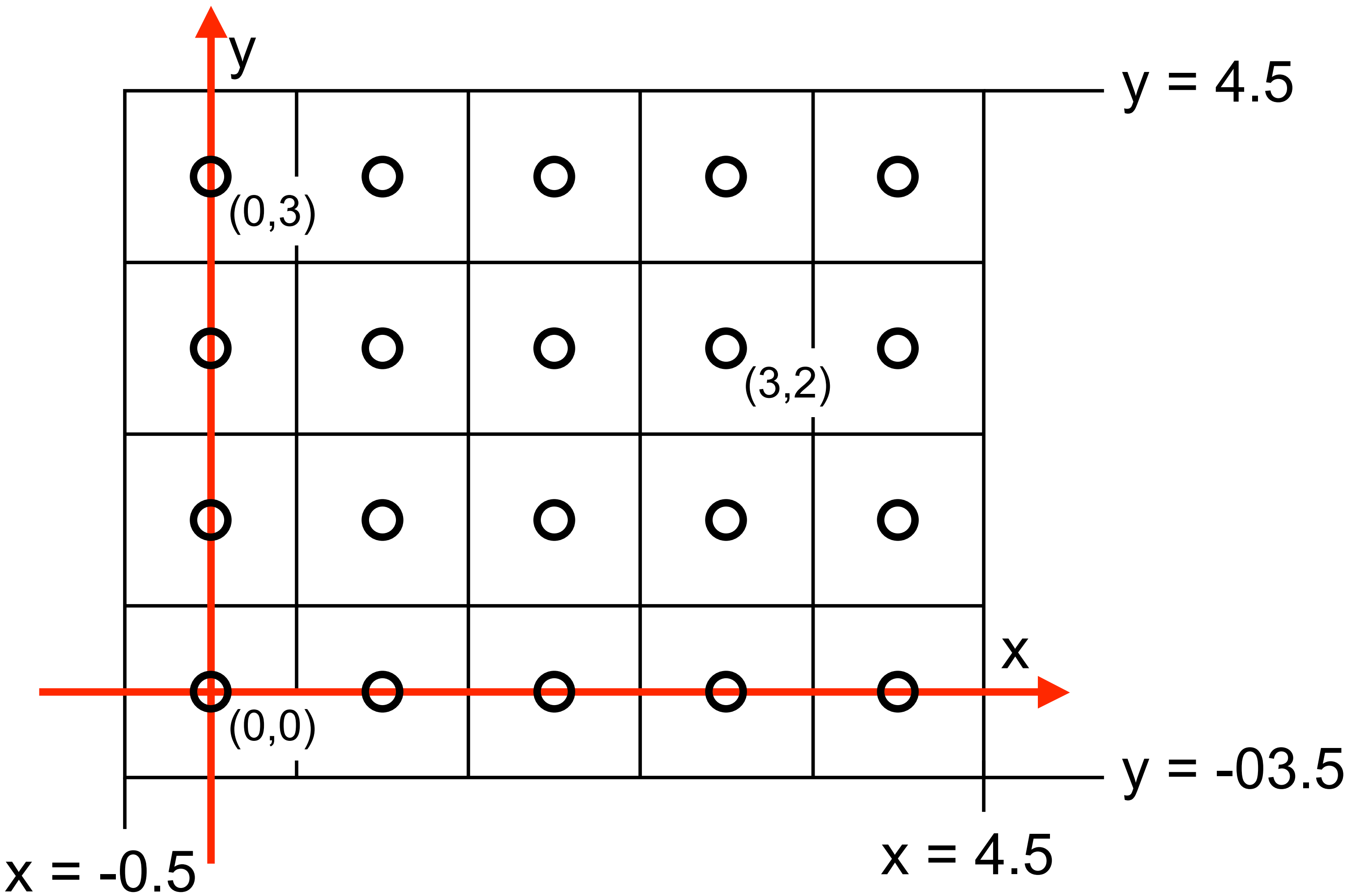
Viewport



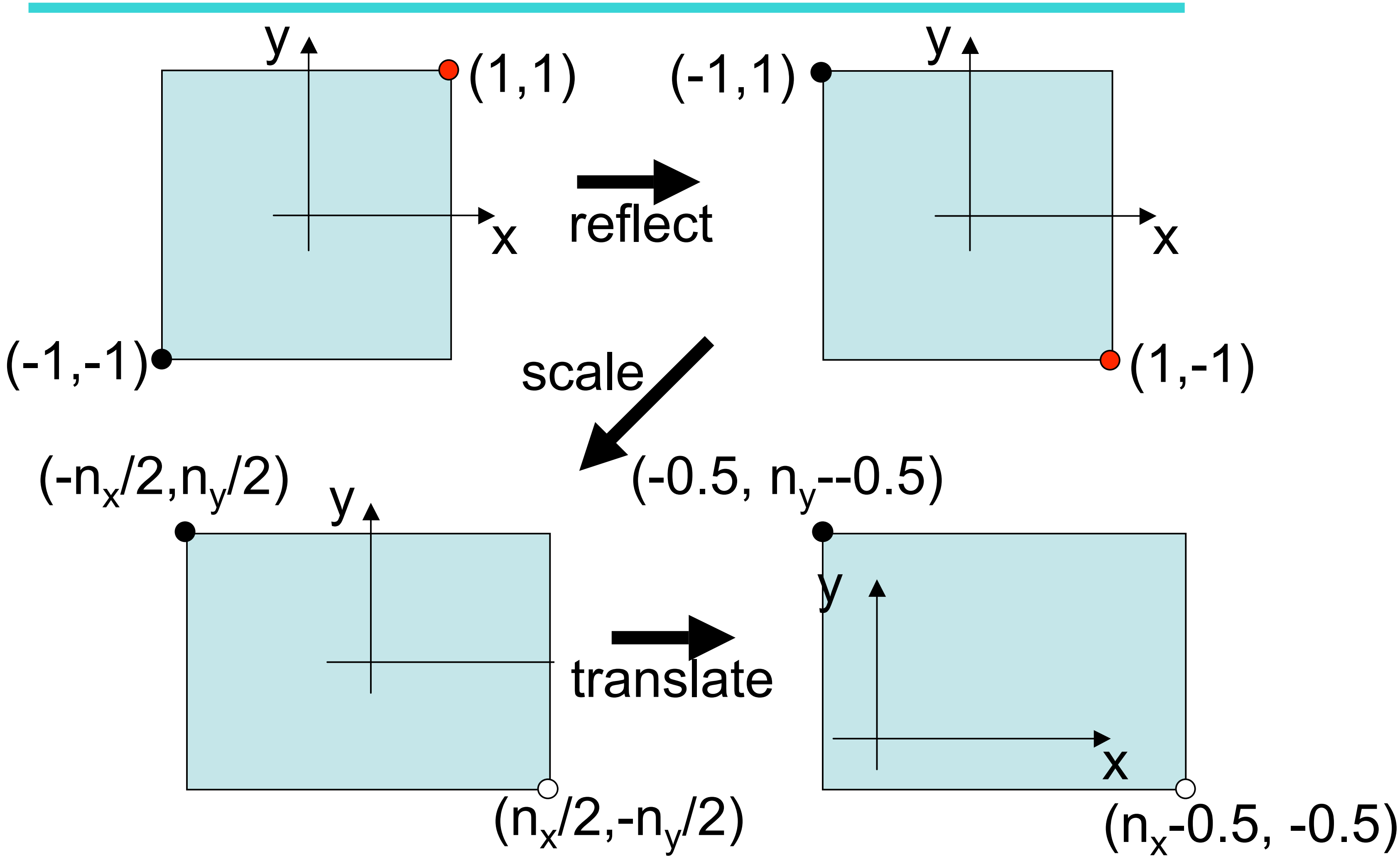
Device



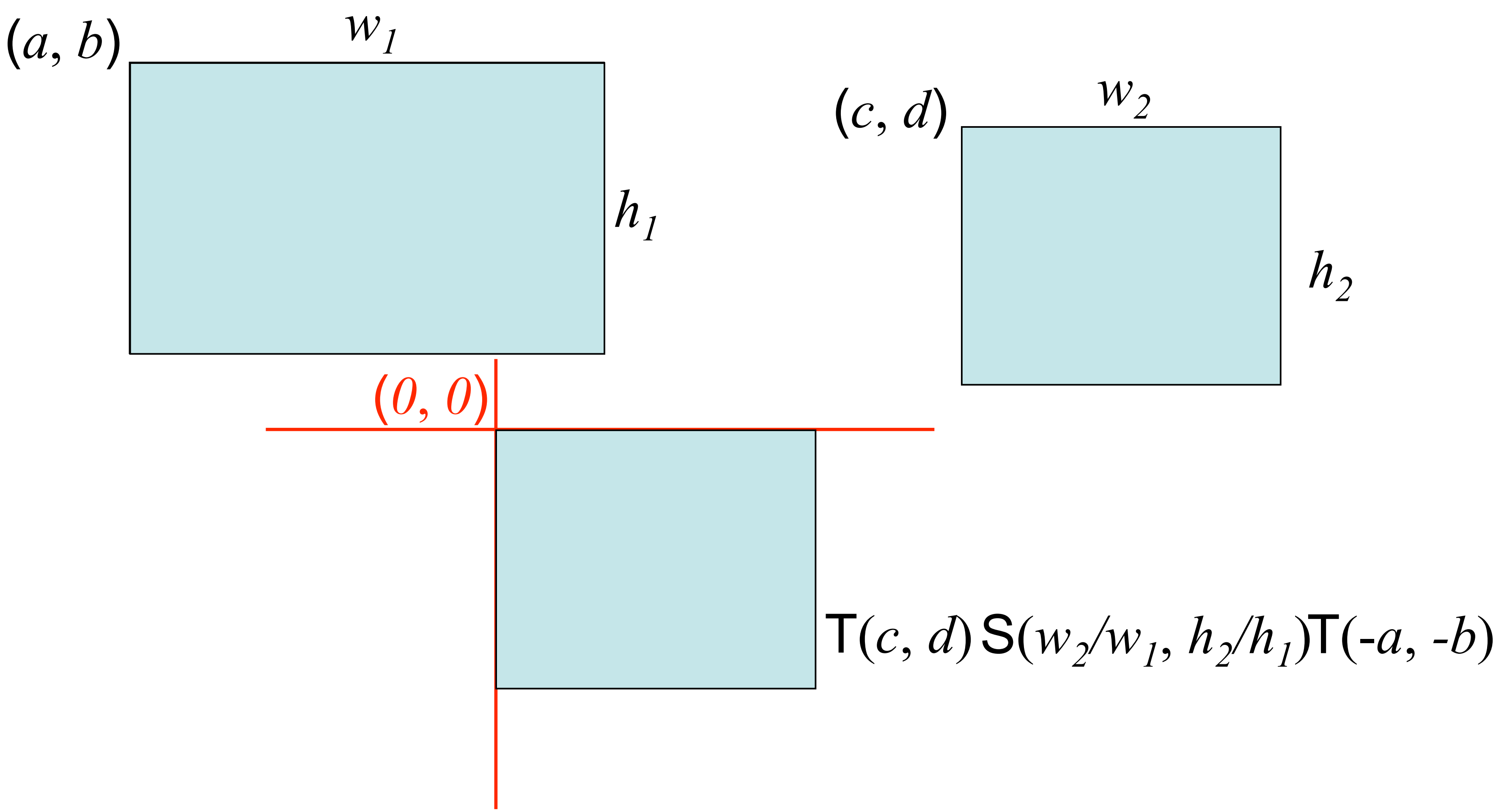
Pixel Coordinates



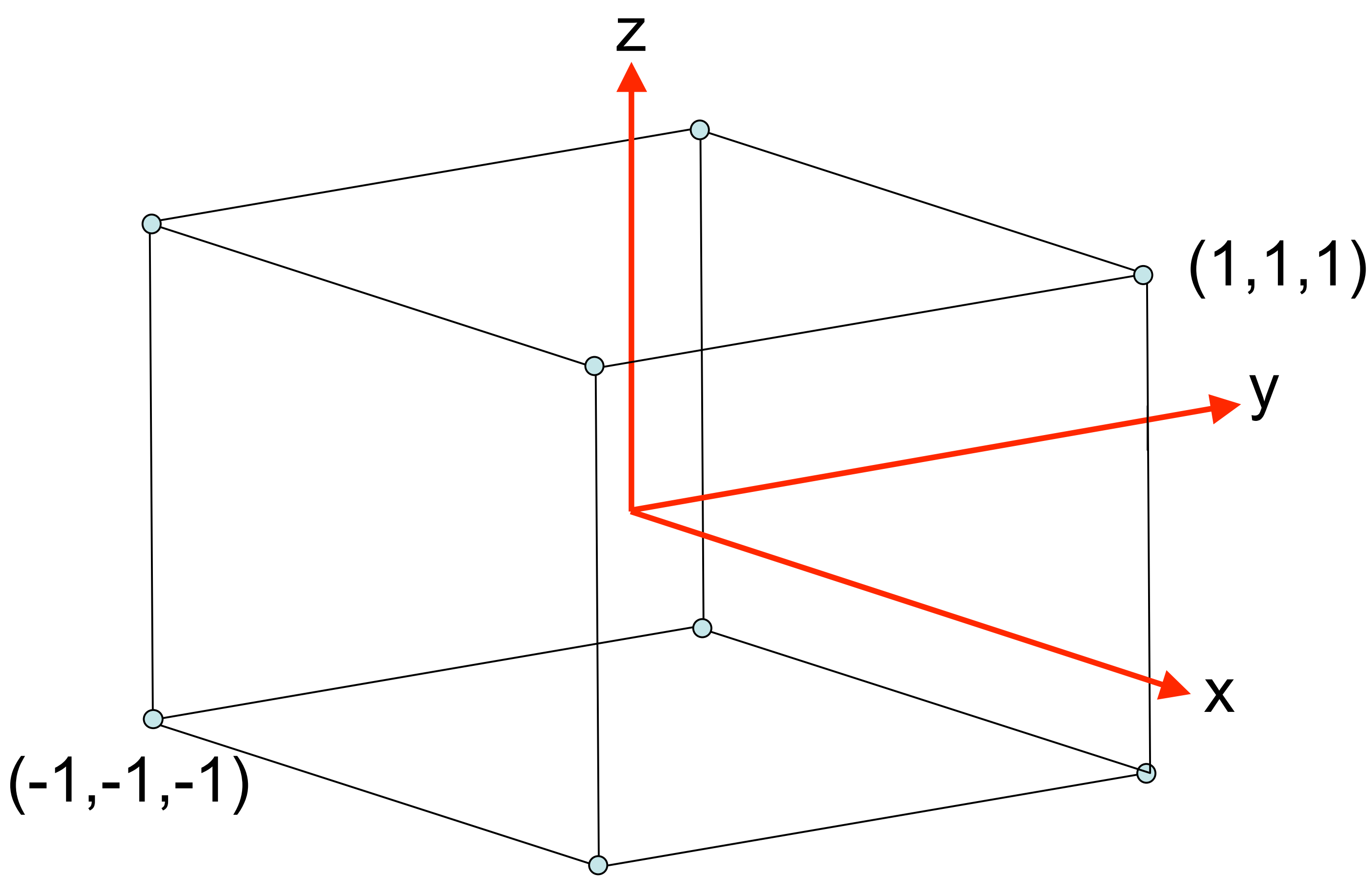
Canonical View to Pixels



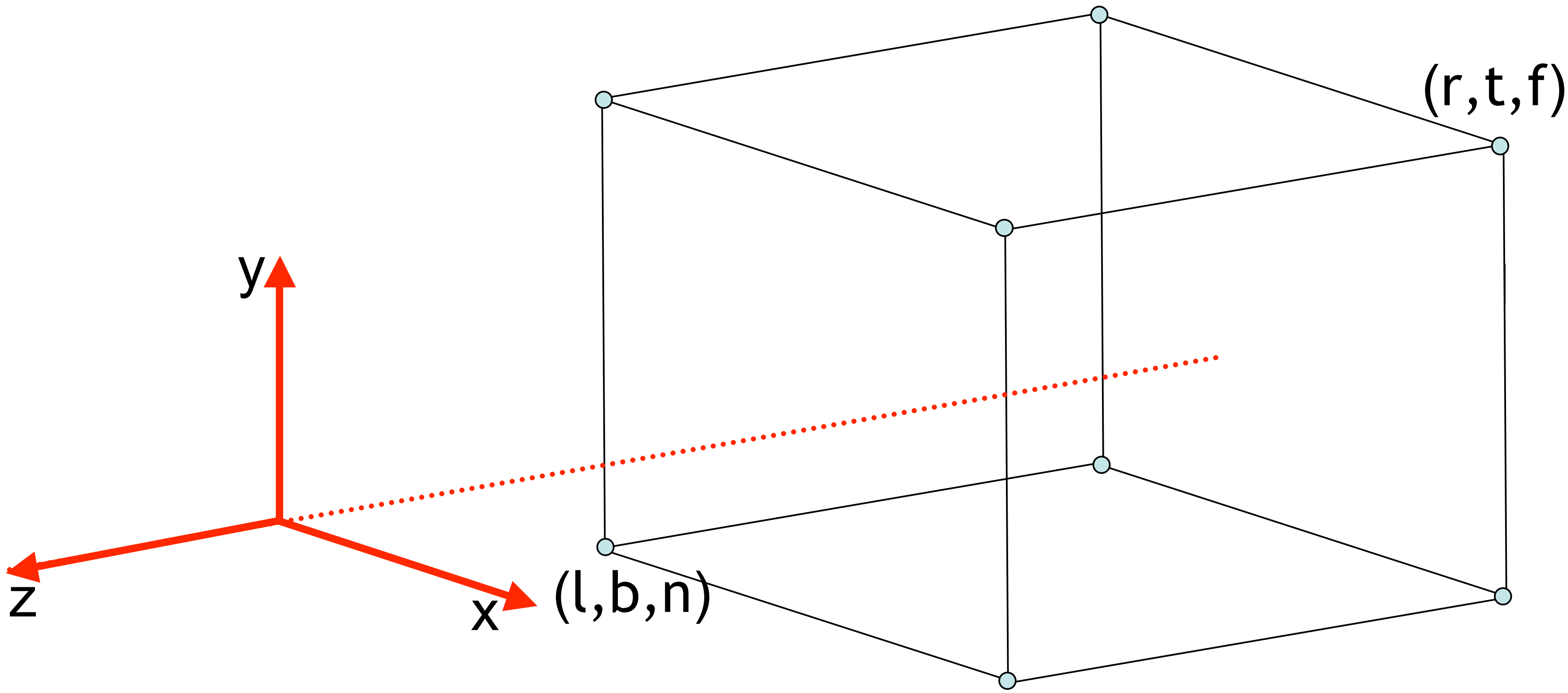
2D Rectangle to Rectangle



Canonical View Volume



Orthographic Projection



left bottom near right top far

Orthographic Projection Math

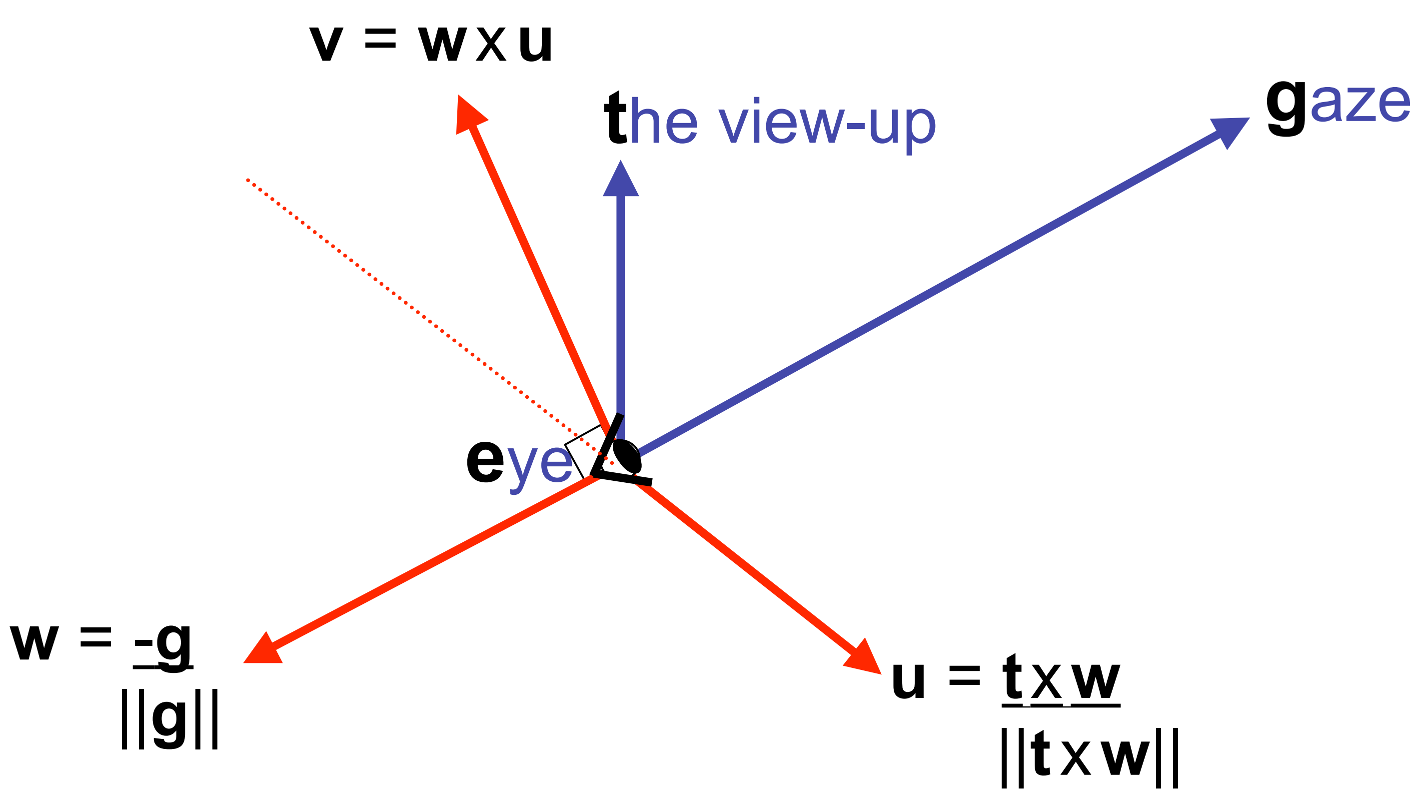
$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Orthographic Projection Math

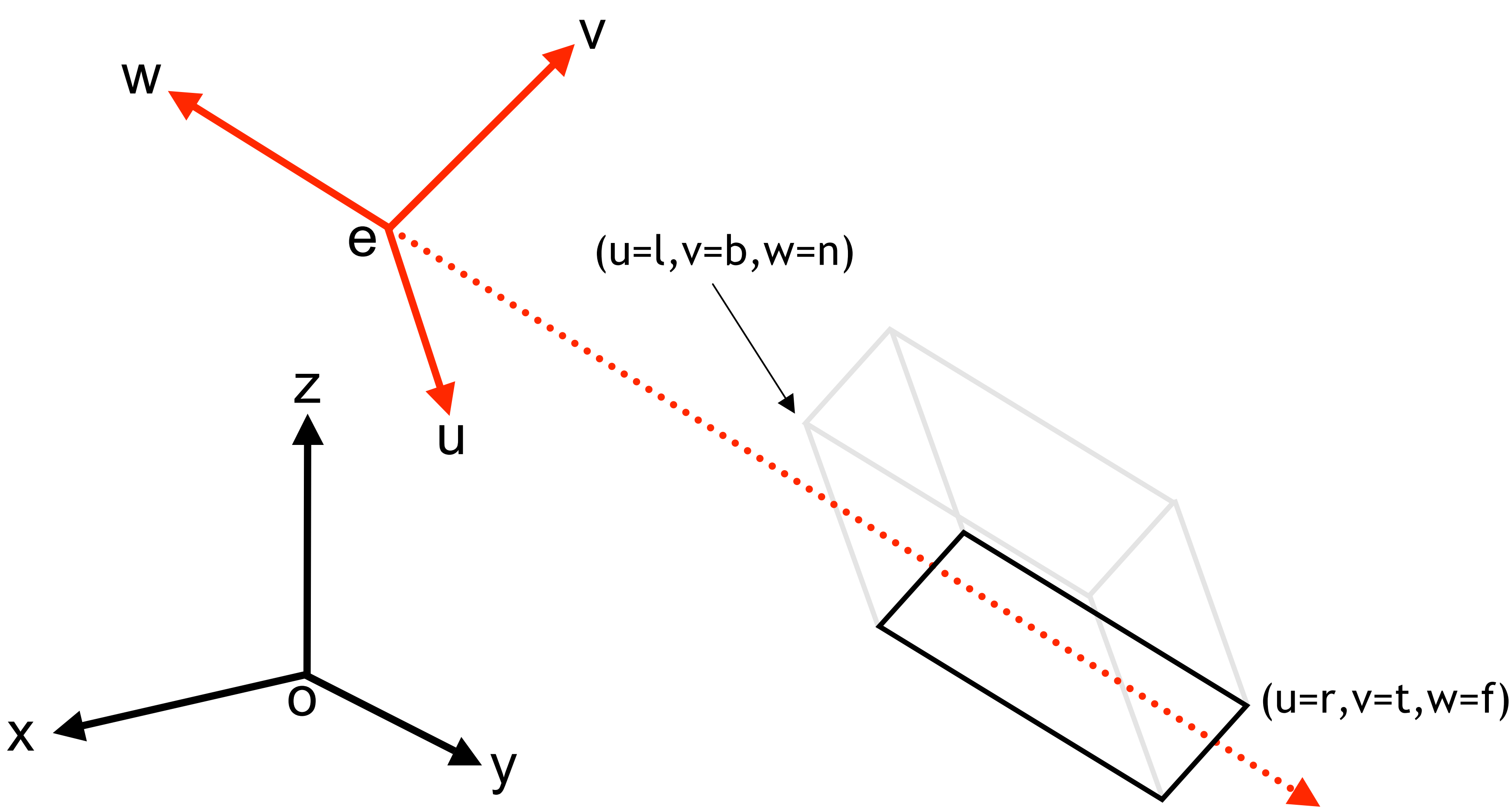
$$M_o = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & 0 \\ 0 & \frac{n_y}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{r - l} & 0 & 0 & 0 \\ 0 & \frac{2}{t - b} & 0 & 0 \\ 0 & 0 & \frac{2}{n - f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ z_{canonical} \\ 1 \end{bmatrix} = M_o \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Arbitrary View Positions



Arbitrary Position Geometry



Arbitrary Position Transformation

Move e to the origin and align (u, v, w) with (x, y, z).

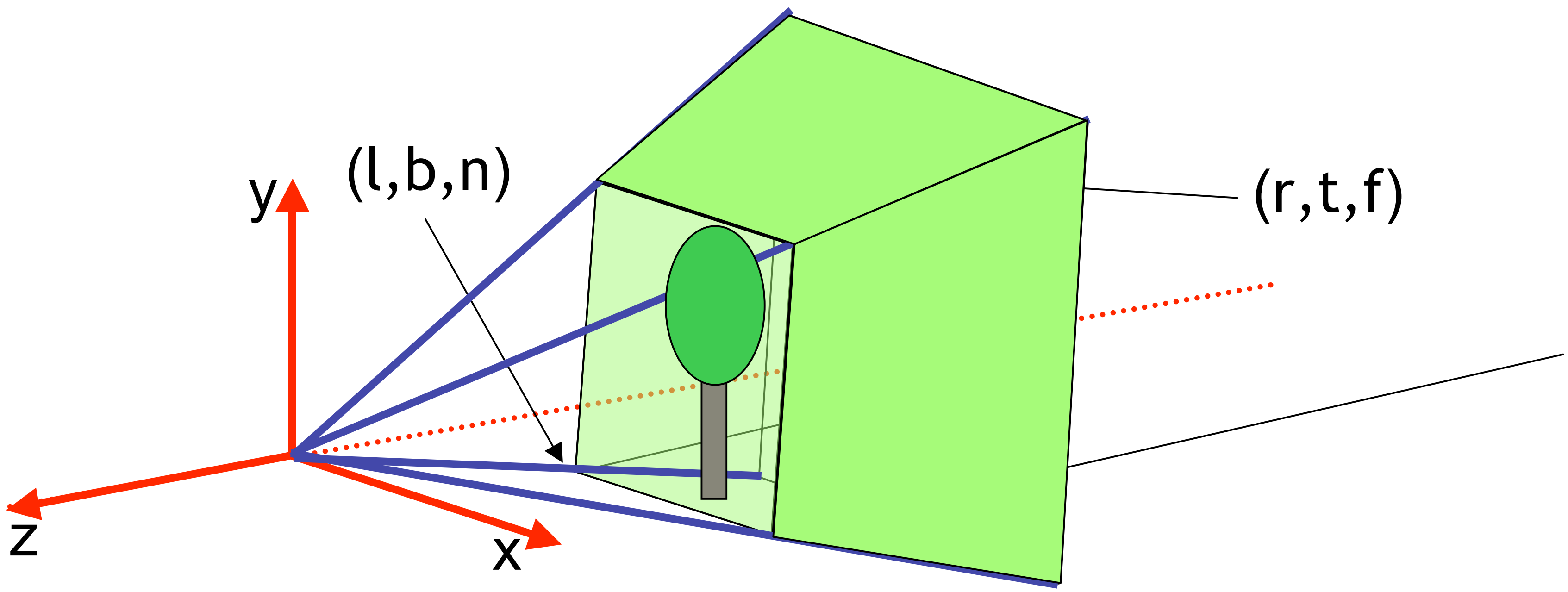
$$M_v = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compute $M = M_o M_v$.

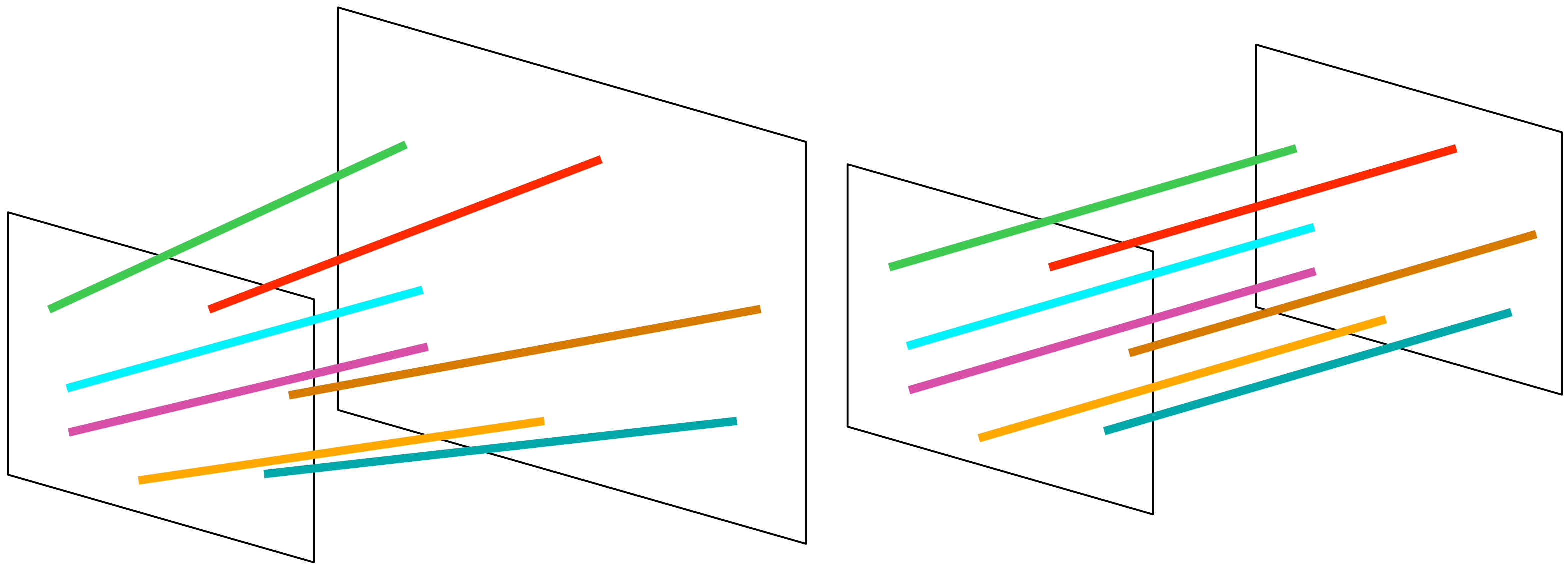
For each line segment (a,b)

$$p = Ma,\ q = Mb,\ drawline(p,q).$$

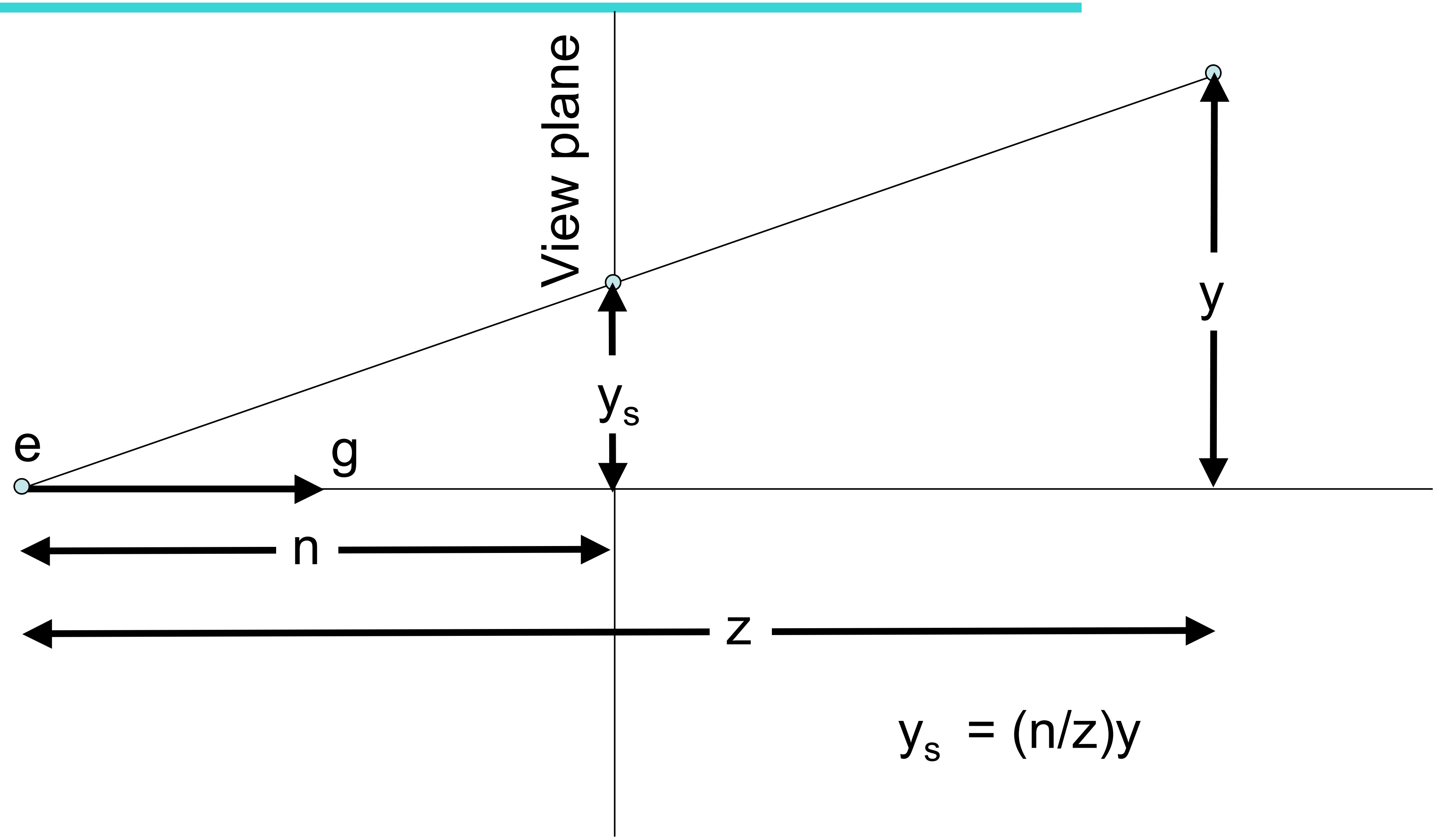
Perspective Projection



Lines to Lines



Perspective Geometry



$y_s = (n/z)y$

Perspective Transformation

The perspective transformation should take

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} nx/z \\ ny/z \\ p(z) \\ 1 \end{bmatrix}$$

Where

$$p(n) = n$$

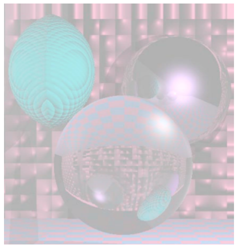
$$p(f) = f$$

and

$$n \geq z_1 > z_2 \geq f$$

implies $p(z_1) > p(z_2)$.

$P(z) = n + f - fn/z$ satisfies these requirements.



Perspective Transformation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} nx/z \\ ny/z \\ n+f-fn/z \\ 1 \end{bmatrix}$$

is not a linear transformation.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} nx \\ ny \\ nz+fz-fn \\ z \end{bmatrix}$$

is a linear transformation.

The Whole Truth about Homogeneous Coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftrightarrow \left\{ \begin{bmatrix} hx \\ hy \\ h \end{bmatrix} \middle| \, h \neq 0 \right\}$$

A point in 2-space corresponds to a line through the origin in 3-space minus the origin itself.

A point in 3-space corresponds to a line through the origin in 4-space minus the origin itself.

Homogenize

$$\begin{bmatrix} 6 \\ 14 \\ 2 \end{bmatrix} \xrightarrow{\text{homogenize}} \begin{bmatrix} 3 \\ 7 \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 3 \\ 7 \end{bmatrix} \qquad \begin{bmatrix} 27 \\ 63 \\ 9 \end{bmatrix} \xrightarrow{\text{homogenize}} \begin{bmatrix} 3 \\ 7 \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 3 \\ 7 \end{bmatrix}$$

$$\begin{bmatrix} 22 \\ 121 \\ 77 \\ 11 \end{bmatrix} \xrightarrow{\text{homogenize}} \begin{bmatrix} 2 \\ 11 \\ 7 \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 2 \\ 11 \\ 7 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 5.5 \\ 3.5 \\ .5 \end{bmatrix} \xrightarrow{\text{homogenize}} \begin{bmatrix} 2 \\ 11 \\ 7 \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 2 \\ 11 \\ 7 \end{bmatrix}$$

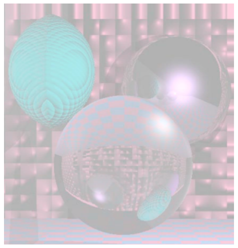
Perspective Transformation Matrix

$$M_p = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad M_p \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} nx \\ ny \\ nz + fz - fn \\ z \end{bmatrix}$$

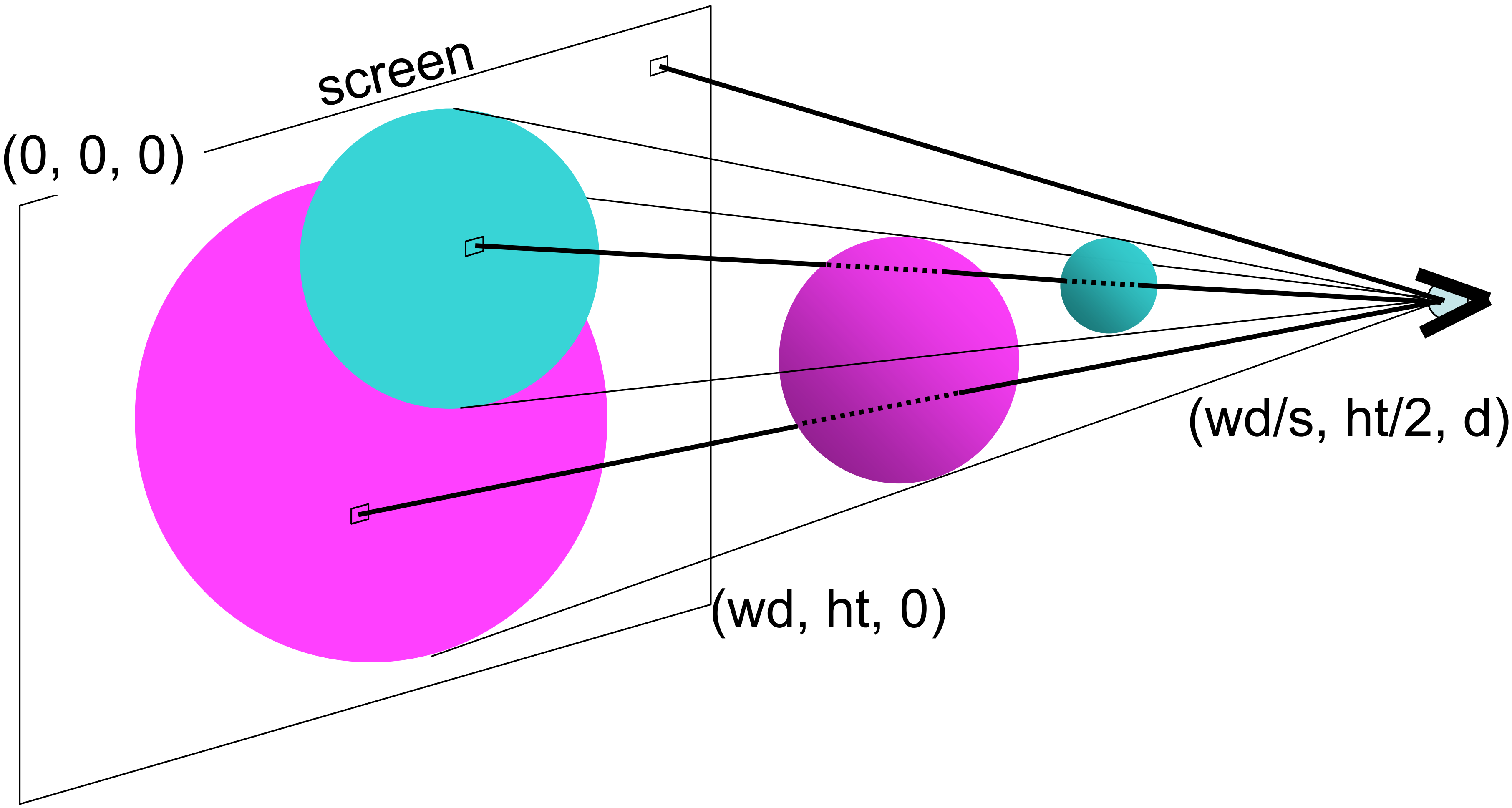
Compute $M = M_o M_p M_v$.

For each line segment (a,b)

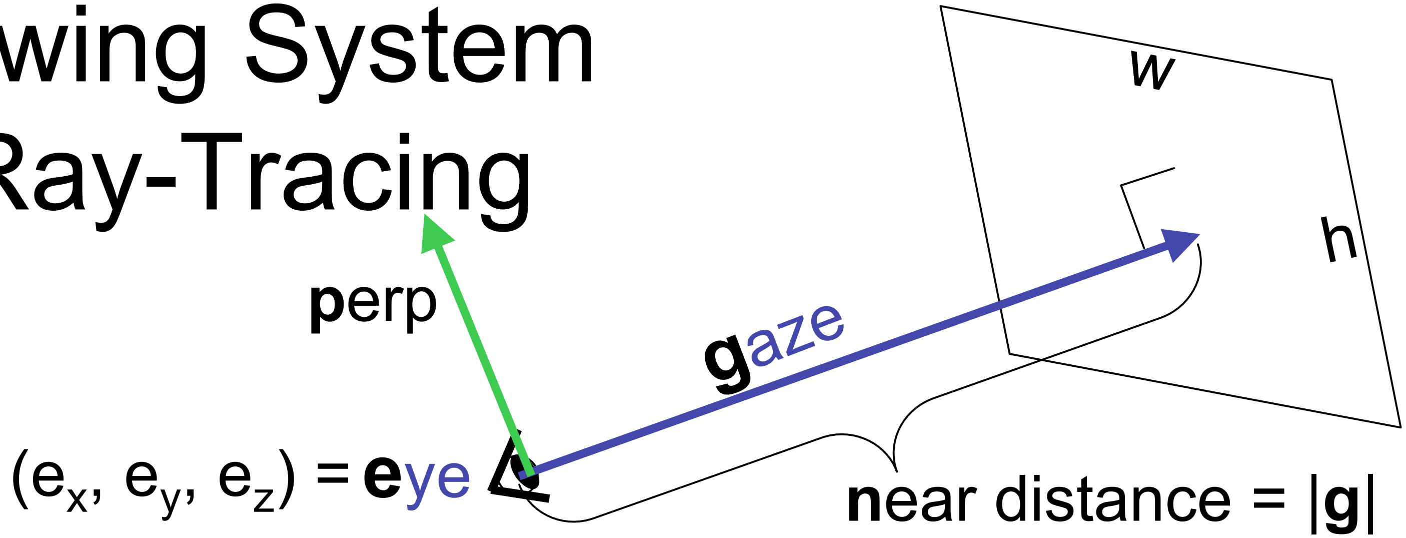
$$p = Ma,\ q = Mb,\ drawline(homogenize(p),\ homogenize(q)).$$



Viewing for Ray-Tracing
Simplest Views

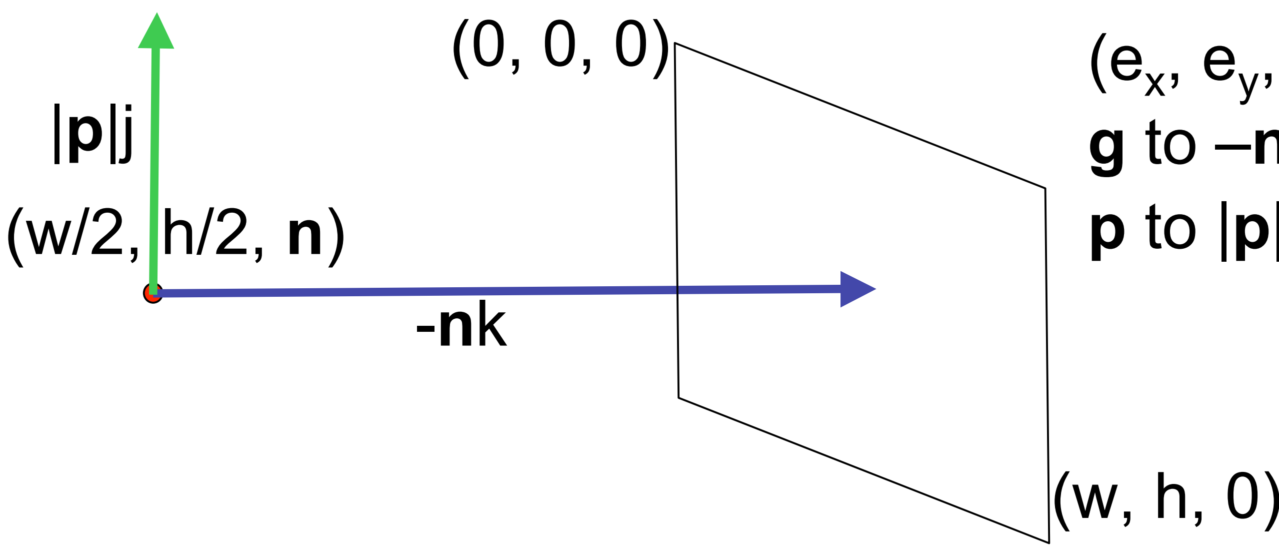


A Viewing System for Ray-Tracing



You need a transformation that sends

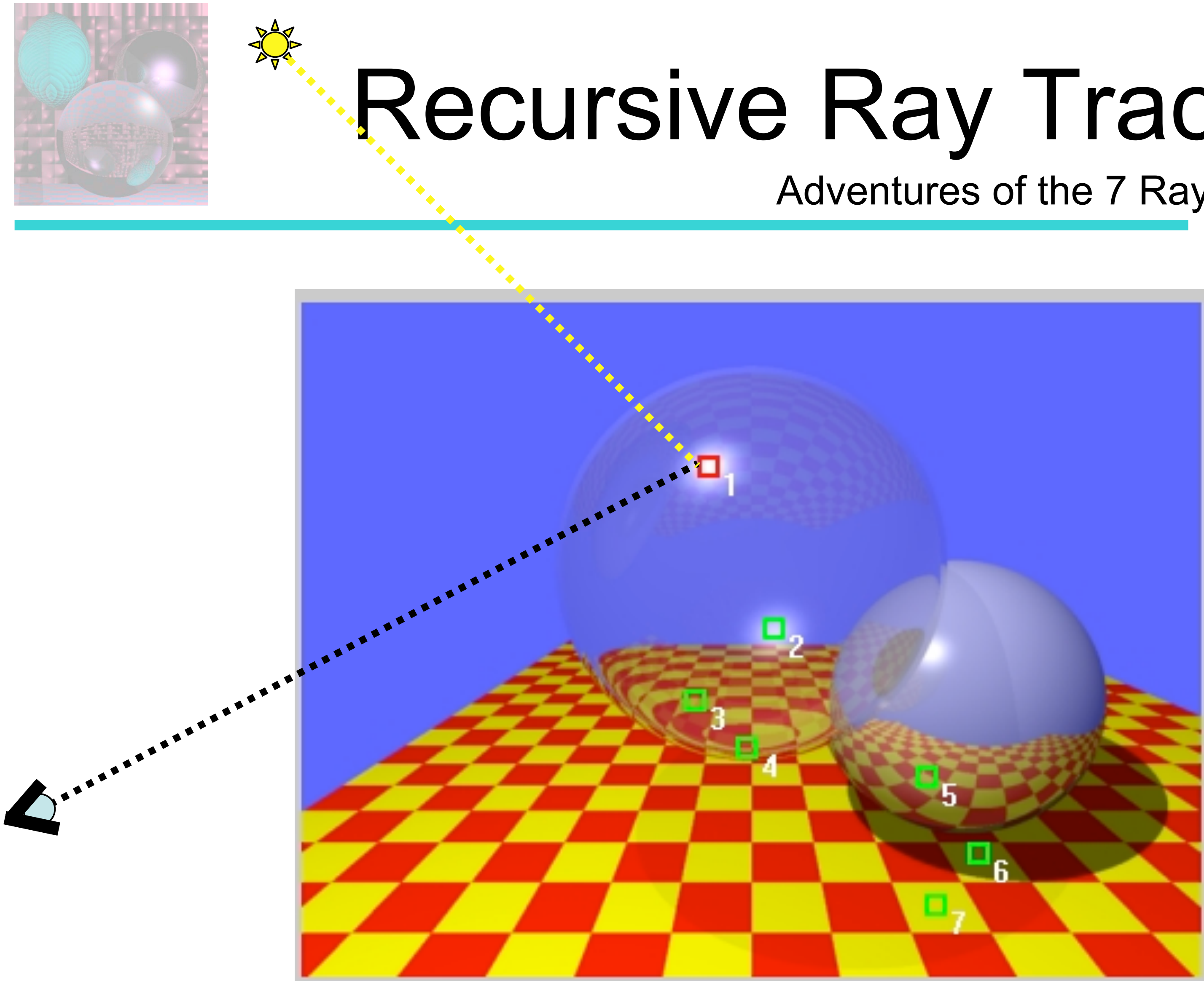
- (e_x, e_y, e_z) to (w/2, h/2, **n**)
- **g** to –**n**k
- **p** to |**p**|j



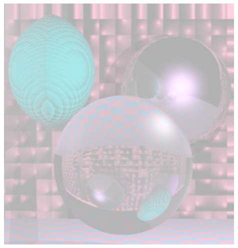
Time for a Break

Recursive Ray Tracing

Adventures of the 7 Rays - Watt

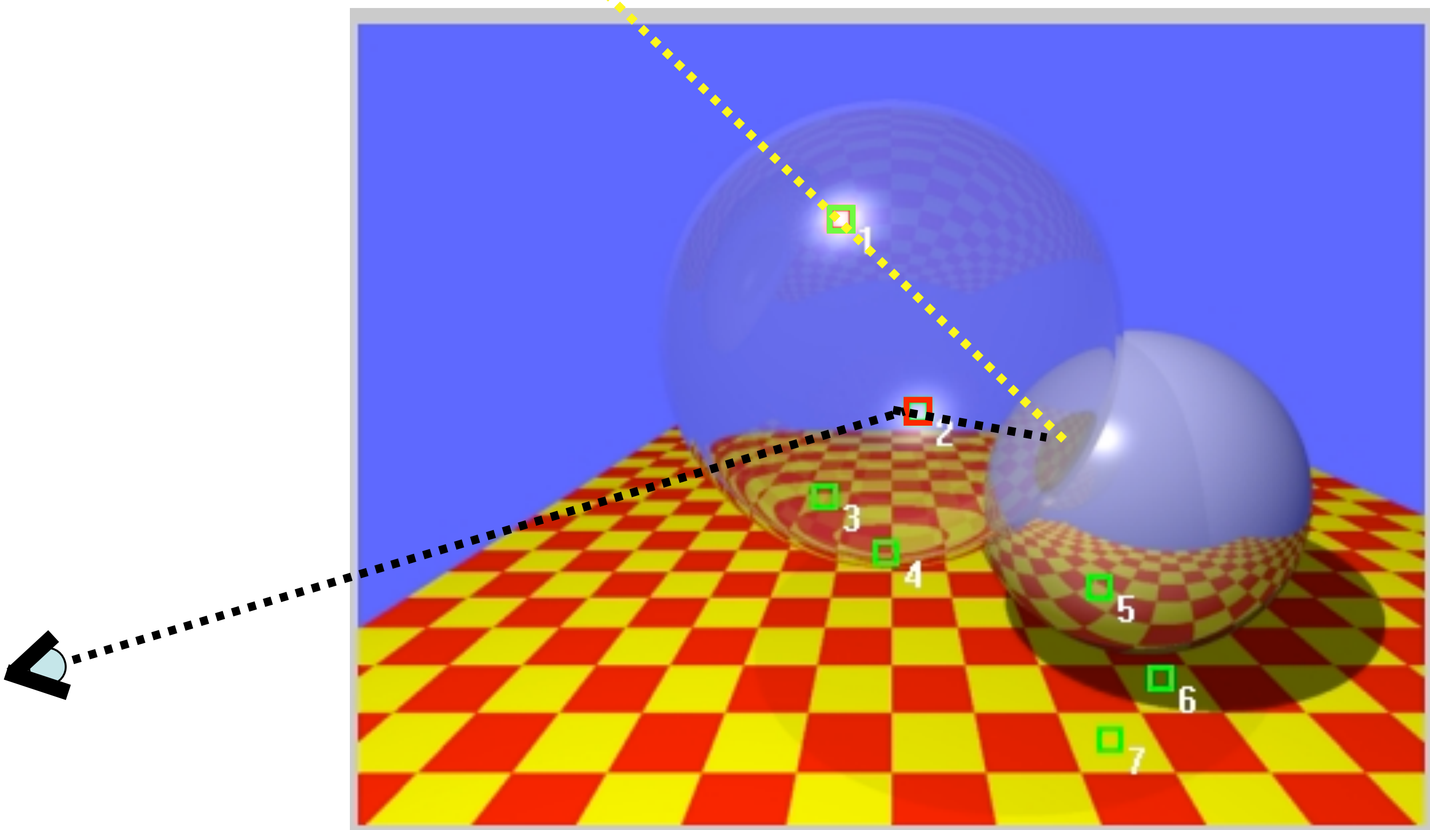


Specular Highlight on Outside of Shere

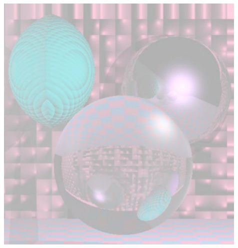


Recursive Ray Tracing

Adventures of the 7 Rays - Watt

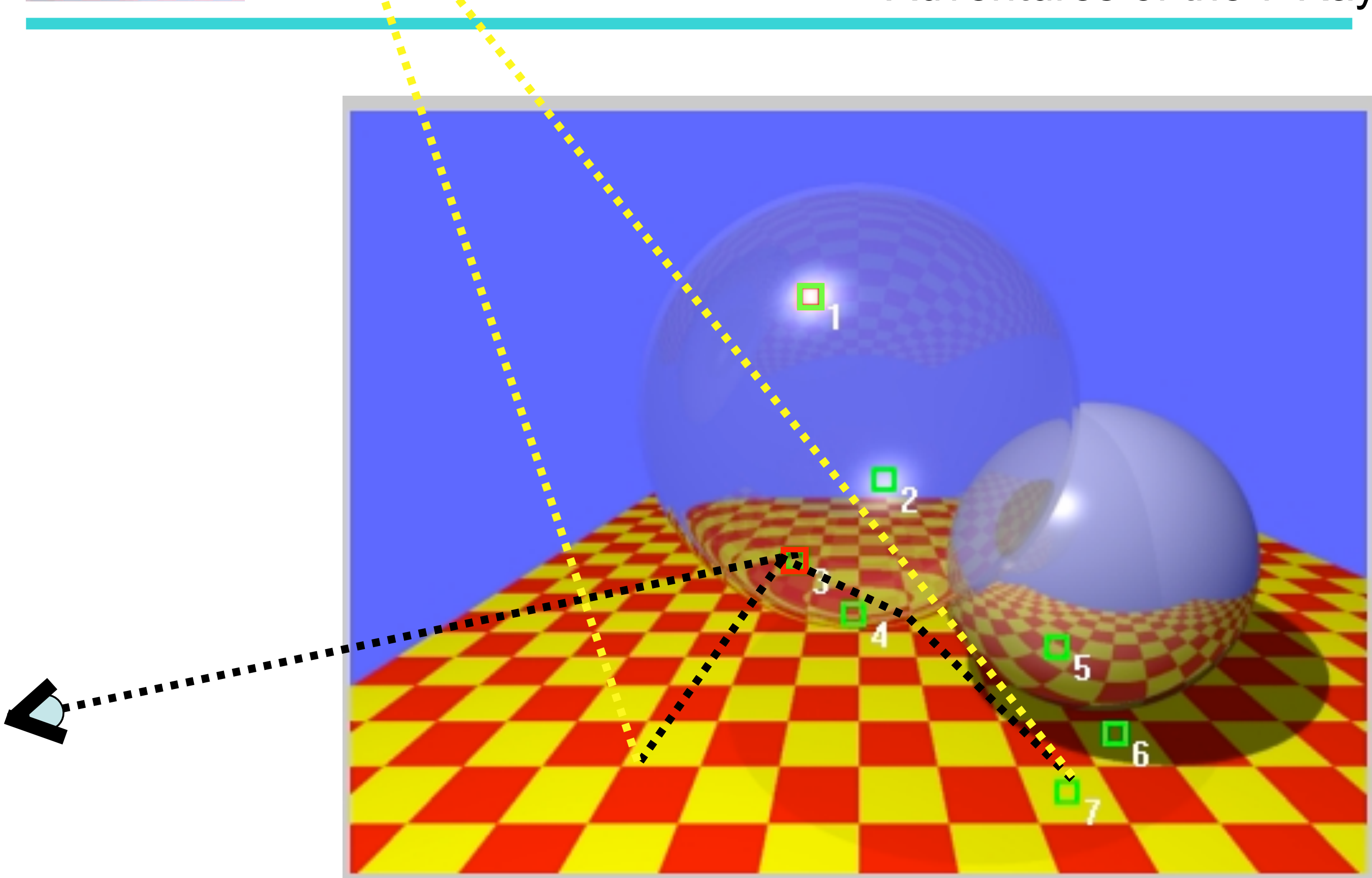


Specular Highlight on Inside of Sphere



Recursive Ray Tracing

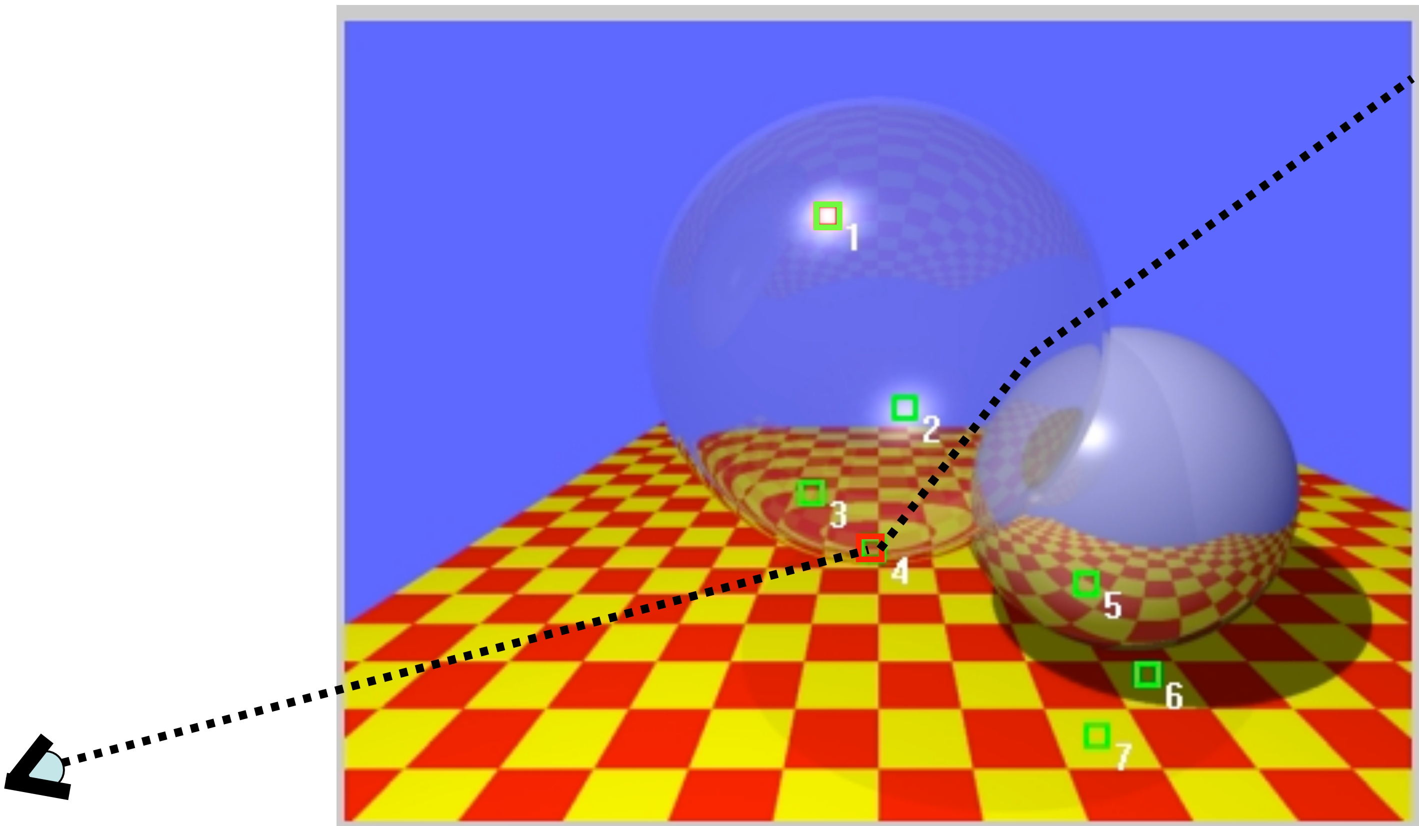
Adventures of the 7 Rays - Watt



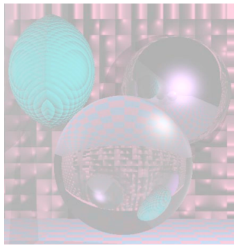
Reflection and Refraction of Checkerboard

Recursive Ray Tracing

Adventures of the 7 Rays - Watt

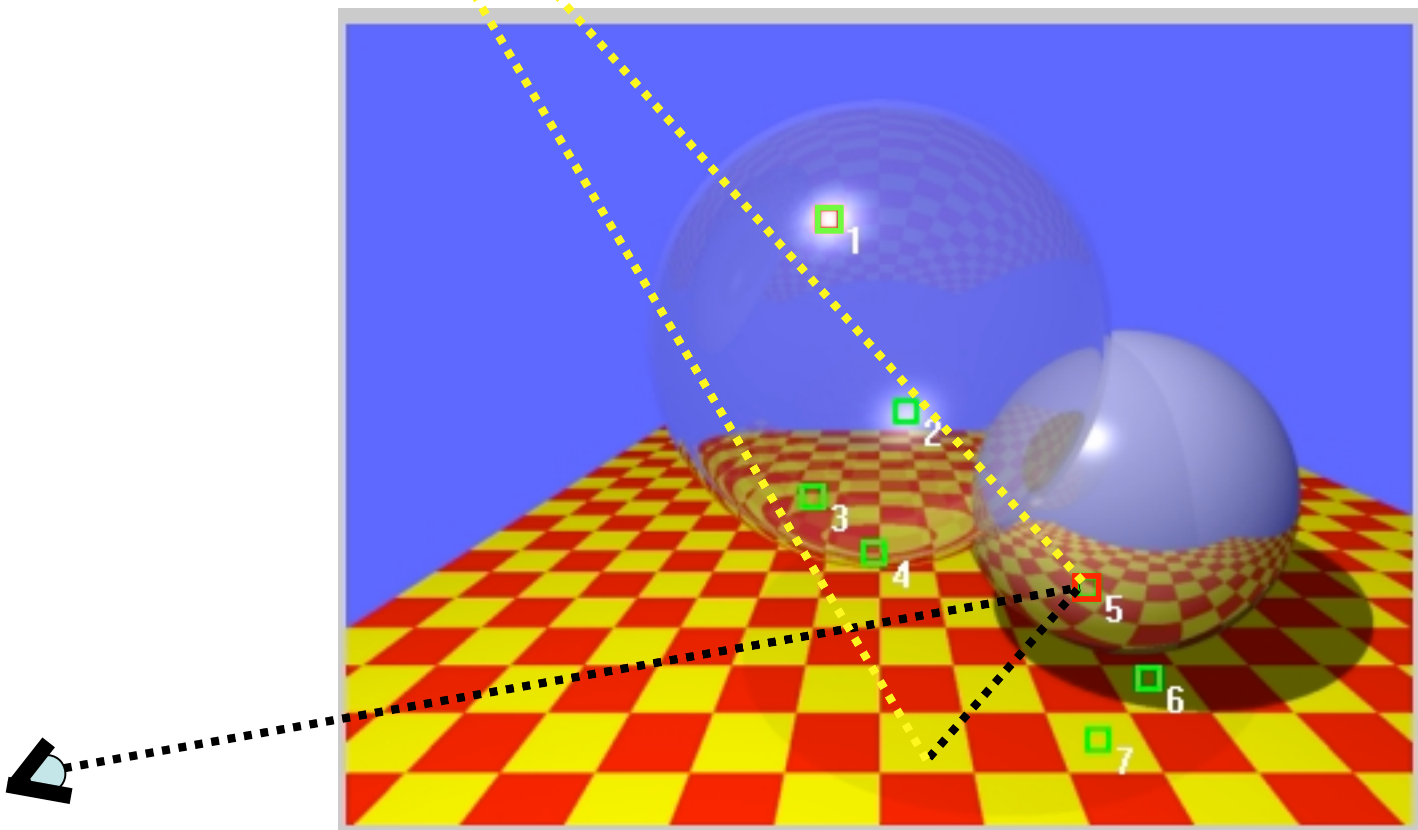


Refraction Hitting Background



Recursive Ray Tracing

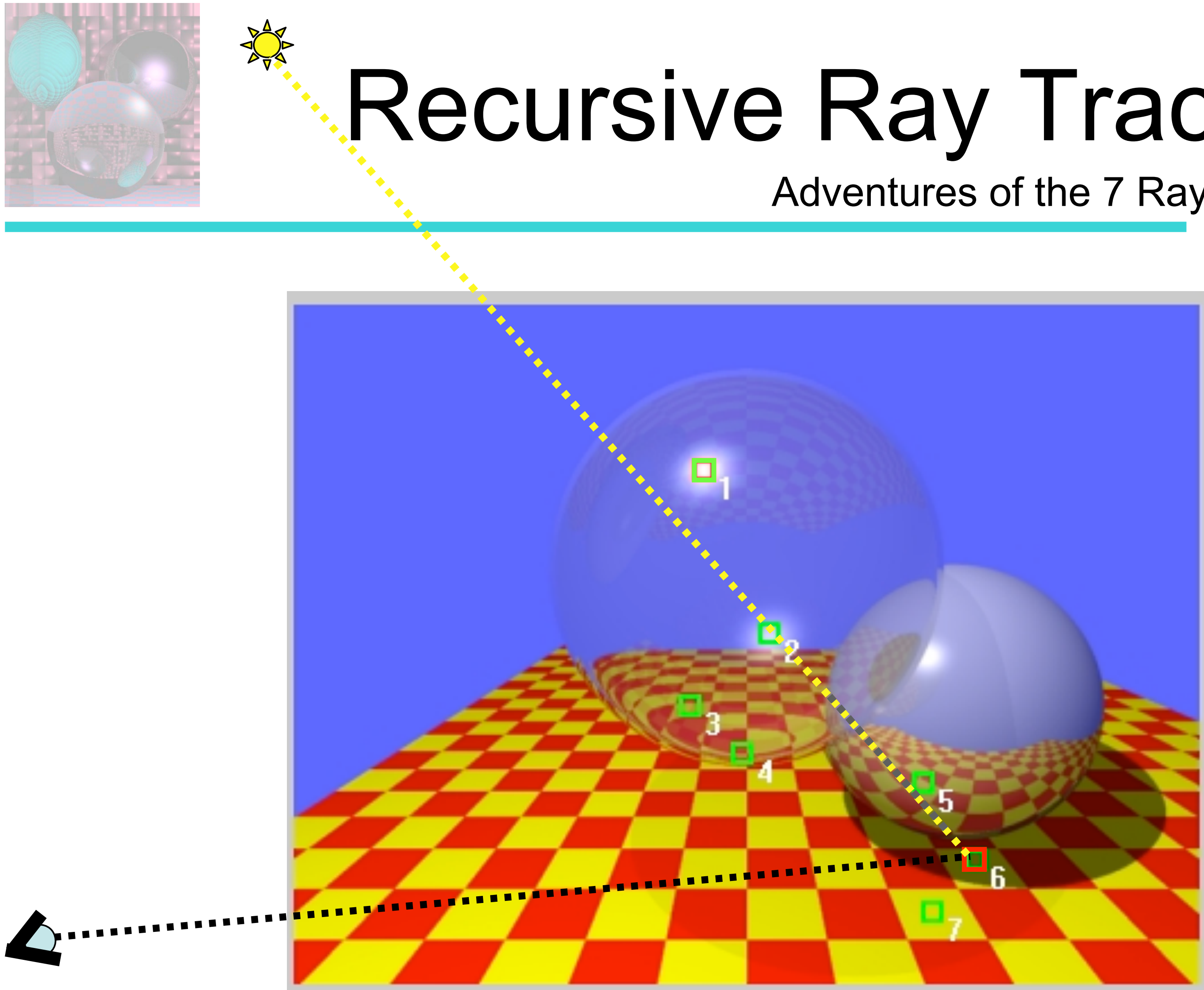
Adventures of the 7 Rays - Watt



Local Diffuse Plus Reflection from Checkerboard

Recursive Ray Tracing

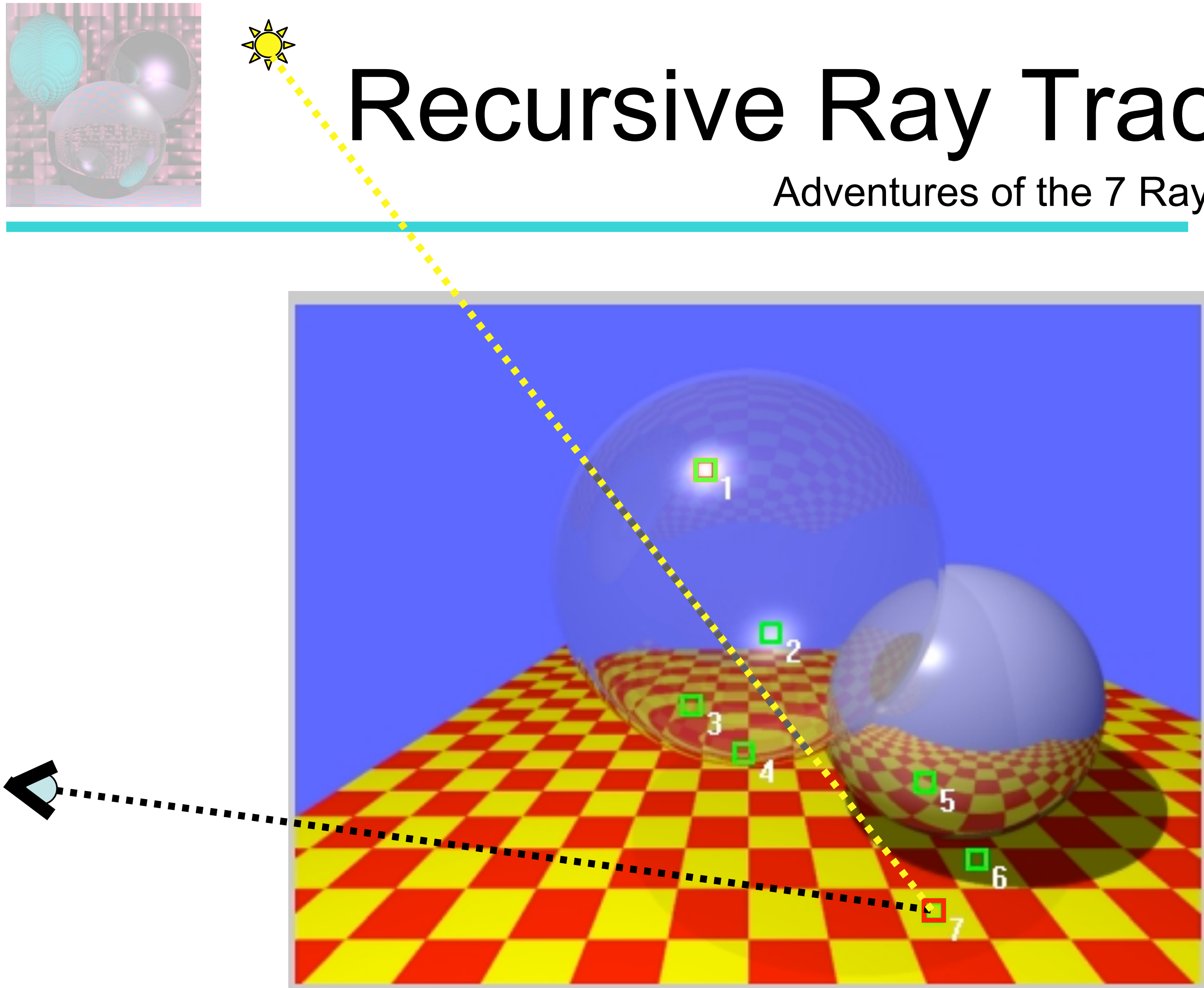
Adventures of the 7 Rays - Watt



Local Diffuse in Complete Shadow

Recursive Ray Tracing

Adventures of the 7 Rays - Watt



Local Diffuse in Shadow from Transparent Sphere

Recursive Ray-Tracing

- How do we know which rays to follow?
- How do we compute those rays?
- How do we organize code so we can follow all those different rays?

```
select center of projection(cp) and window on view plane;
for (each scan line in the image ) {
  for (each pixel in scan line ) {
    determine  ray from the cp through the pixel;
    pixel = RT_trace(ray, 1);}}

// intersect ray with objects; compute shade at closest intersection
// depth is current depth in ray tree
RT_color RT_trace (RT_ray ray; int depth){
  determine closest intersection of ray with an object;
  if (object hit) {
    compute normal at intersection;
    return RT_shade (closest object hit, ray, intersection, normal,
                            depth);}
  else
    return BACKGROUND_VALUE;
}
```

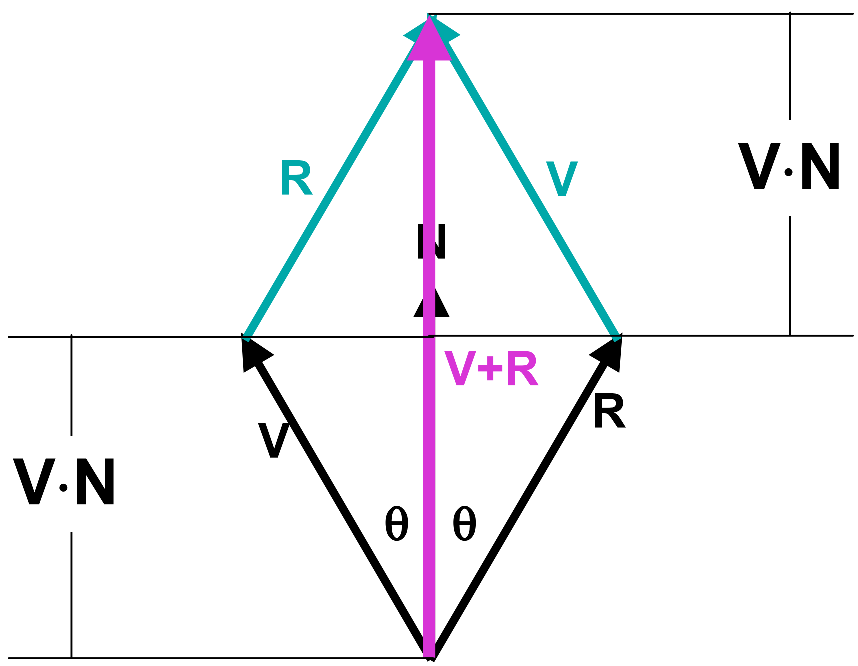
```
// Compute shade at point on object,
// tracing rays for shadows, reflection, refraction.
RT_color RT_shade (
  RT_object object,   // Object intersected
  RT_ray ray,         // Incident ray
  RT_point point,     // Point of intersection to shade
  RT_normal normal,// Normal at point
  int depth )         // Depth in ray tree
{
RT_color  color;          // Color of ray
RT_ray rRay, tRay, sRay;// Reflected, refracted, and shadow ray
  color = ambient term ;
  for ( each light  ) {
    sRay = ray from point to light ;
    if (  dot product of normal and direction to light is positive  ){
      compute how much light is blocked by opaque and
      transparent surfaces, and use to scale diffuse and specular
      terms before adding them to color;}}
```

```
    if ( depth < maxDepth ) {      // return if depth is too deep
        if ( object is reflective ) {
            rRay = ray in reflection direction from point;
            rColor = RT_trace(rRay, depth + 1);
            scale rColor by specular coefficient and add to color;
        }
        if ( object  is transparent  ) {
            tRay = ray in refraction direction from point;
            if ( total internal reflection does not occur  ) {
                tColor = RT_trace(tRay, depth + 1);
                scale tColor by transmission coefficient
                and add to color;
            }
        }
    }
    return color;   // Return the color of the ray
}
```

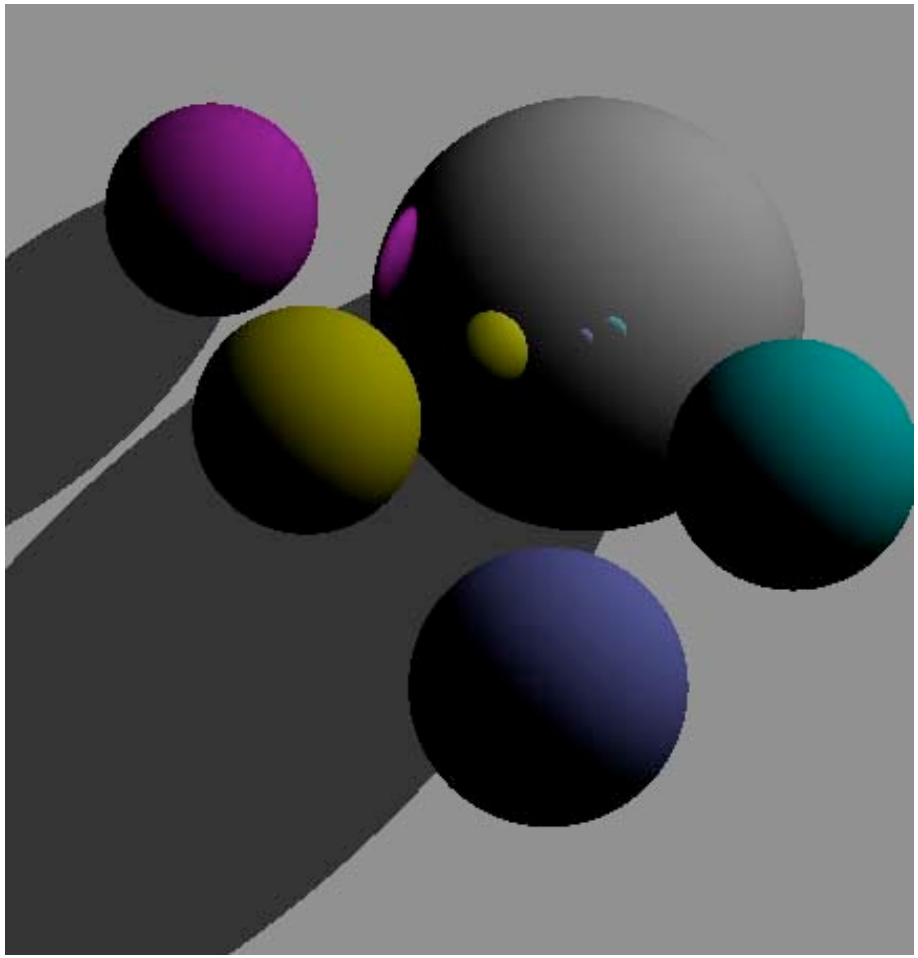
Computing **R**

V + **R** = (2 **V**·**N**) **N**

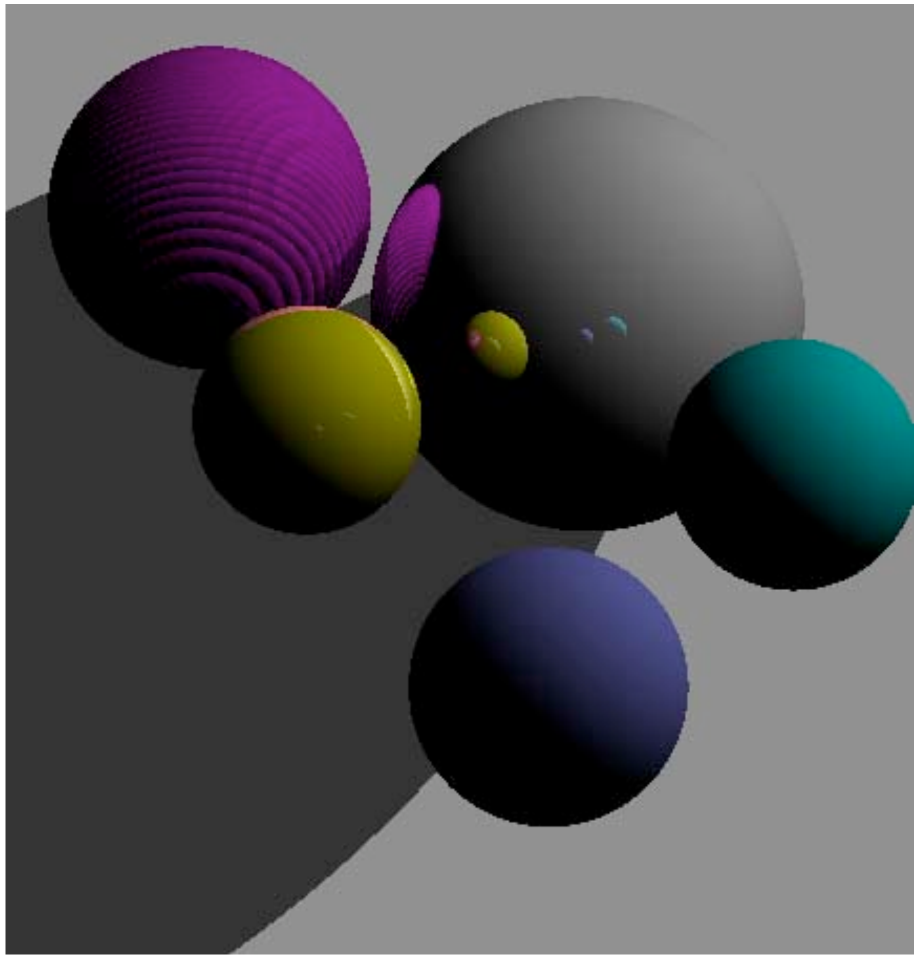
R = (2 **V**·**N**) **N** - **V**



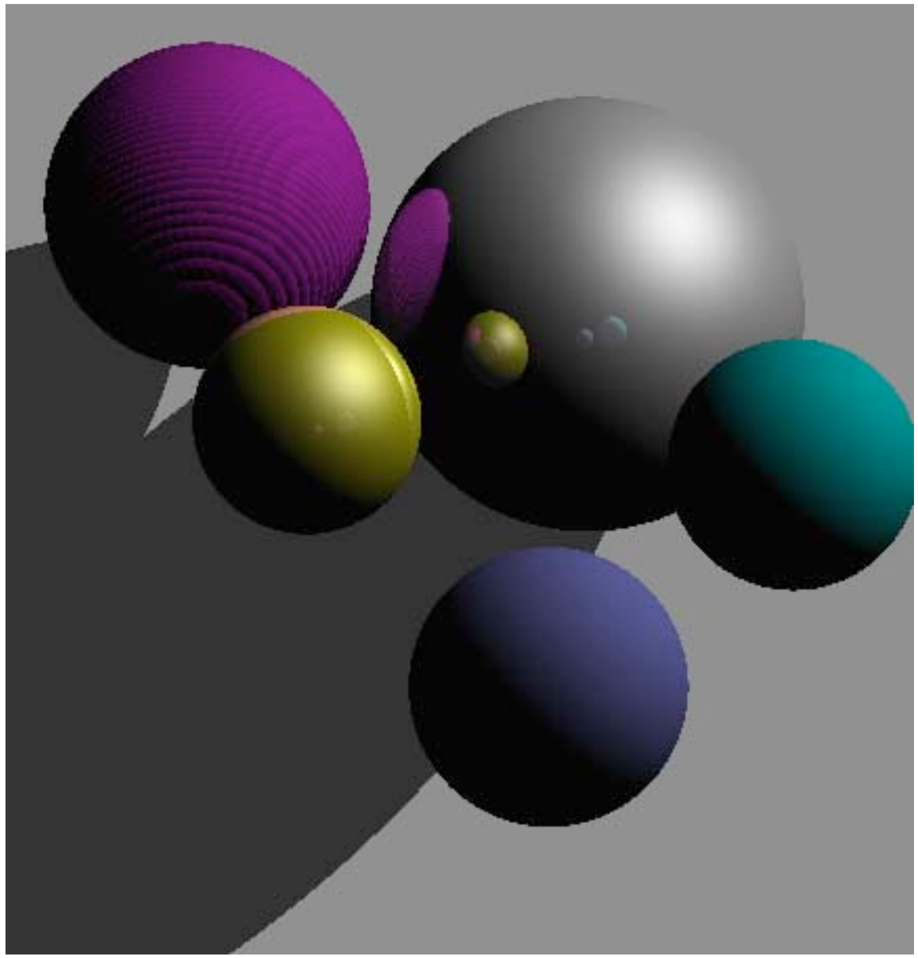
Reflections, no Highlight



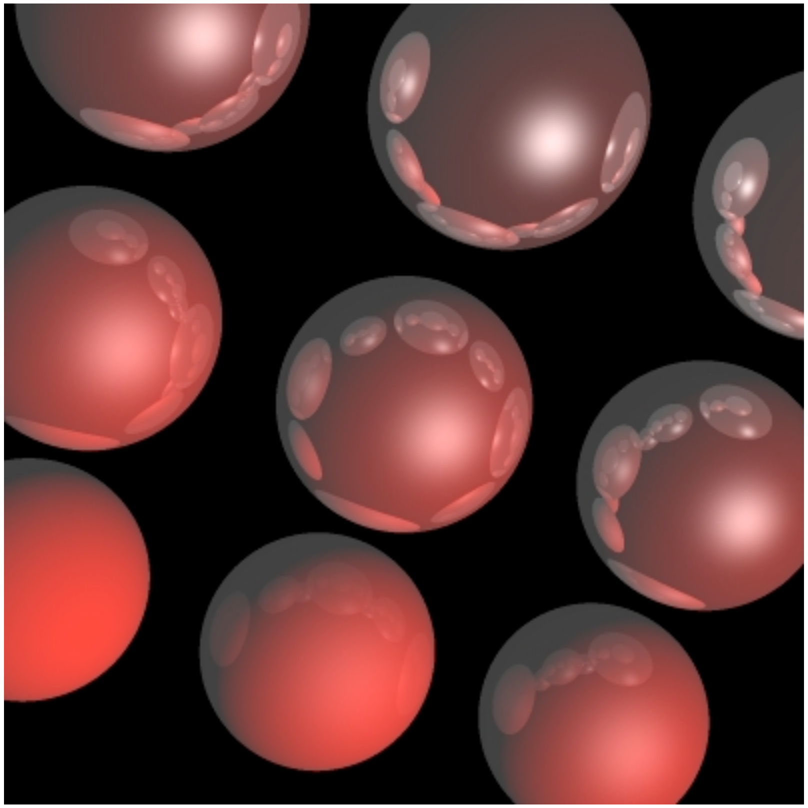
Second Order Reflection



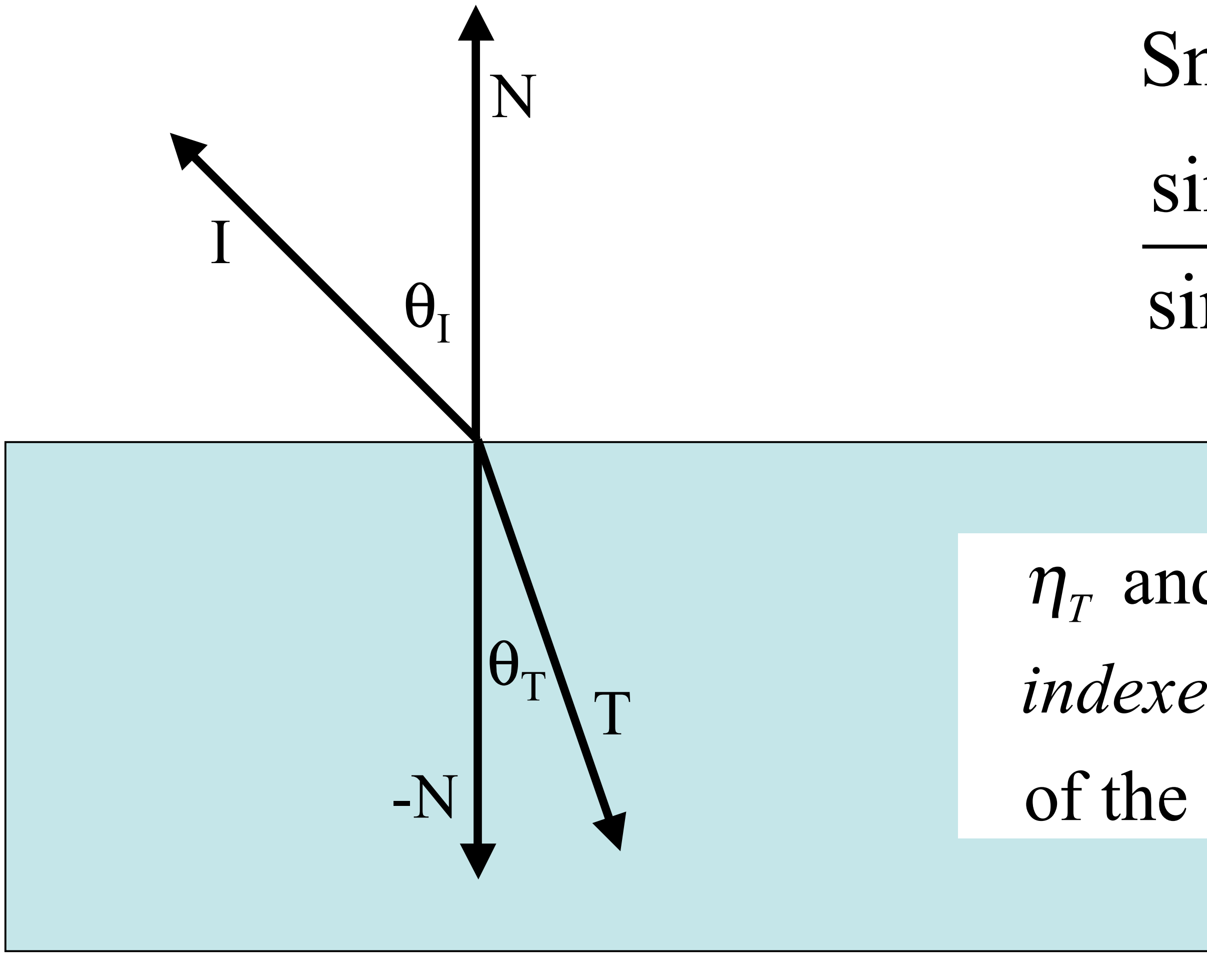
Refelction with Highlight



Nine Red Balls



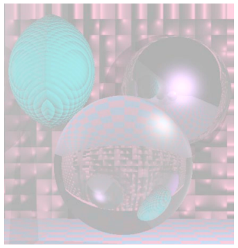
Refraction



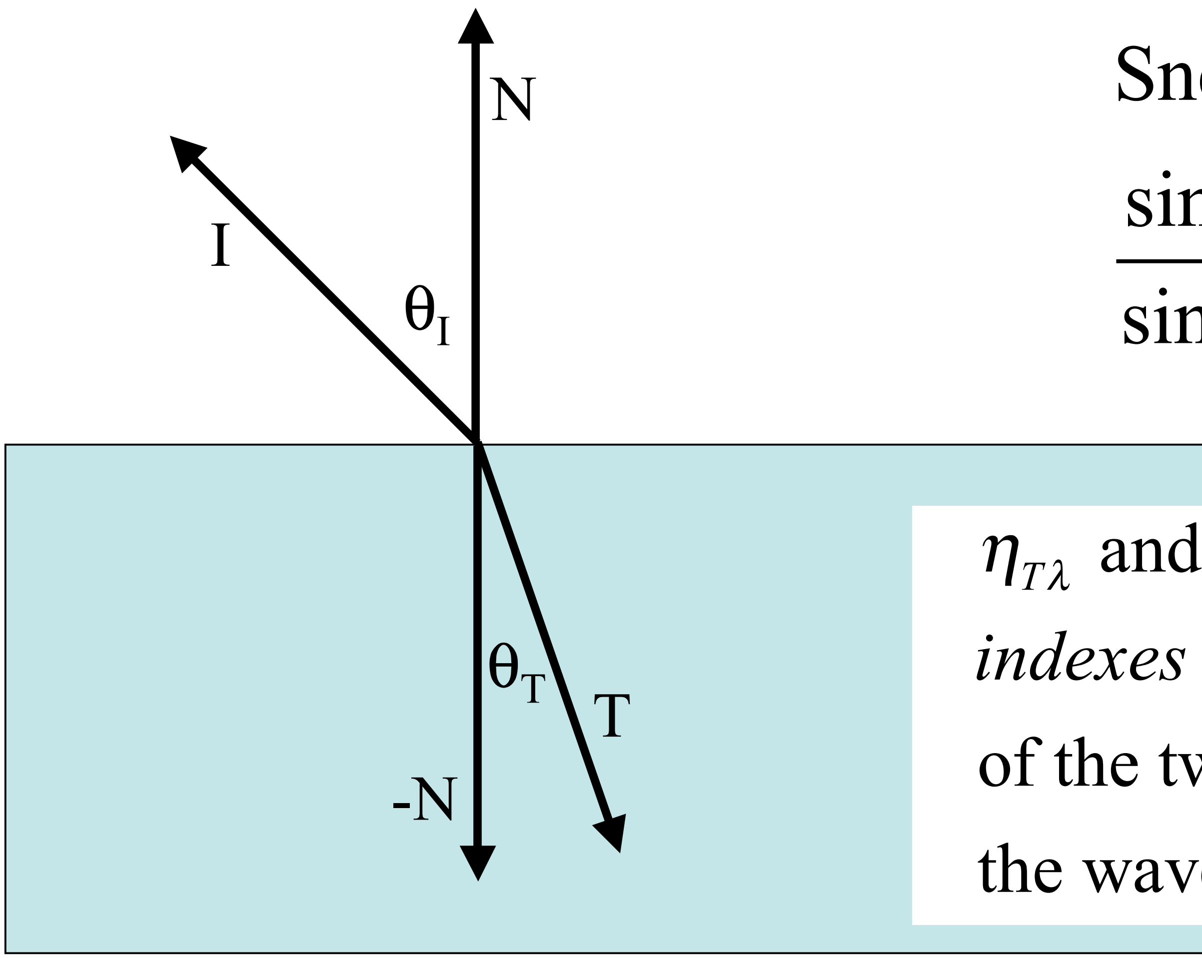
Snell's Law

$$\frac{\sin(\theta_I)}{\sin(\theta_T)} = \frac{\eta_T}{\eta_I}$$

η_T and η_I are the *indexes of refraction* of the two mediums.



Refraction and Wavelength

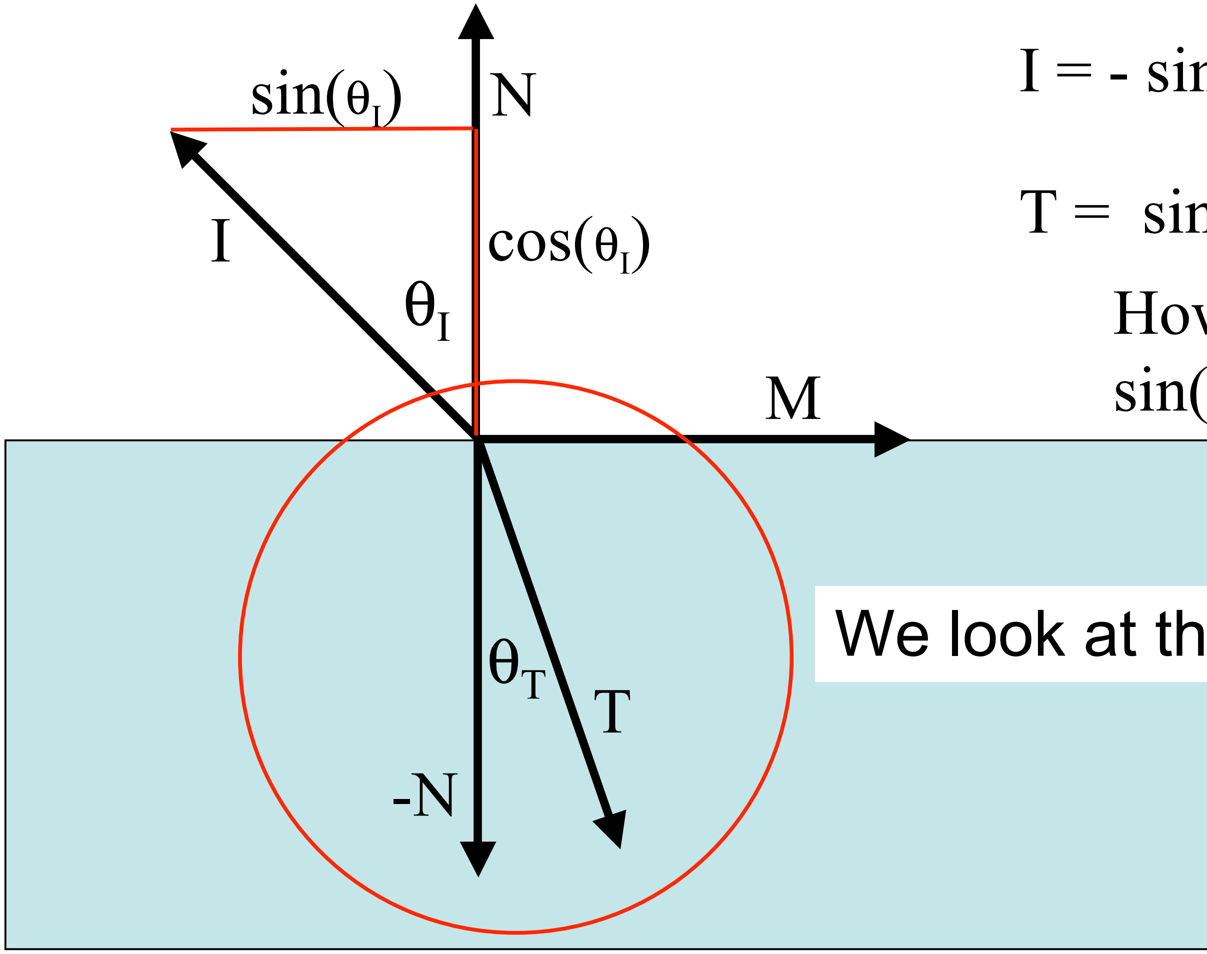


Snell's Law

$$\frac{\sin(\theta_I)}{\sin(\theta_T)} = \frac{\eta_{T\lambda}}{\eta_{I\lambda}}$$

$\eta_{T\lambda}$ and $\eta_{I\lambda}$ are the *indexes of refraction* of the two mediums for the wavelength of light λ.

Computing T



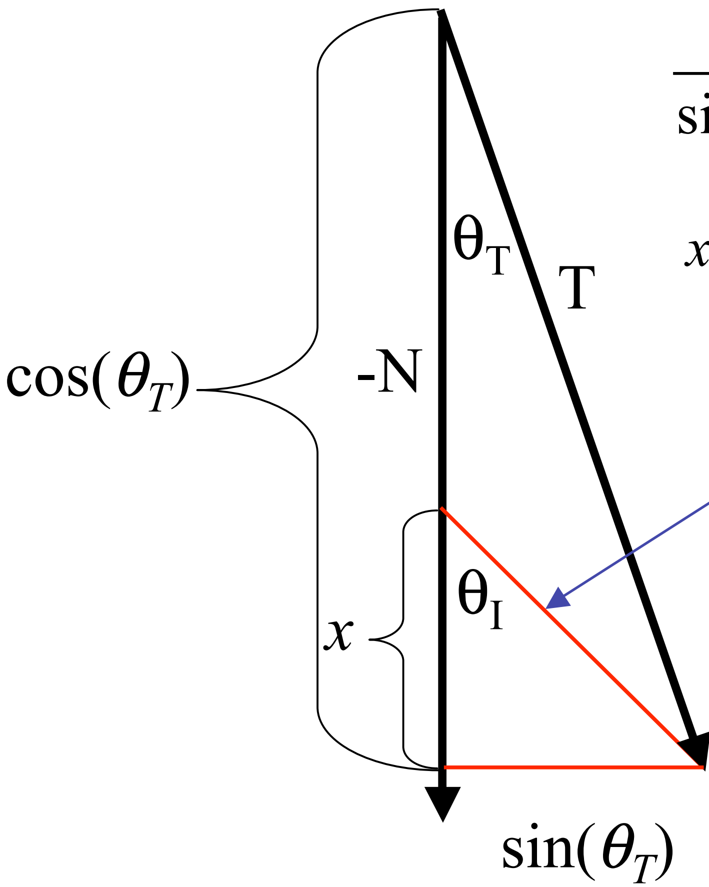
$I = -\sin(\theta_I)M + \cos(\theta_I)N$

$T = \sin(\theta_T)M - \cos(\theta_T)N$

How do we compute M, $\sin(\theta_T)$, and $\cos(\theta_T)$?

We look at this in more detail

Computing T

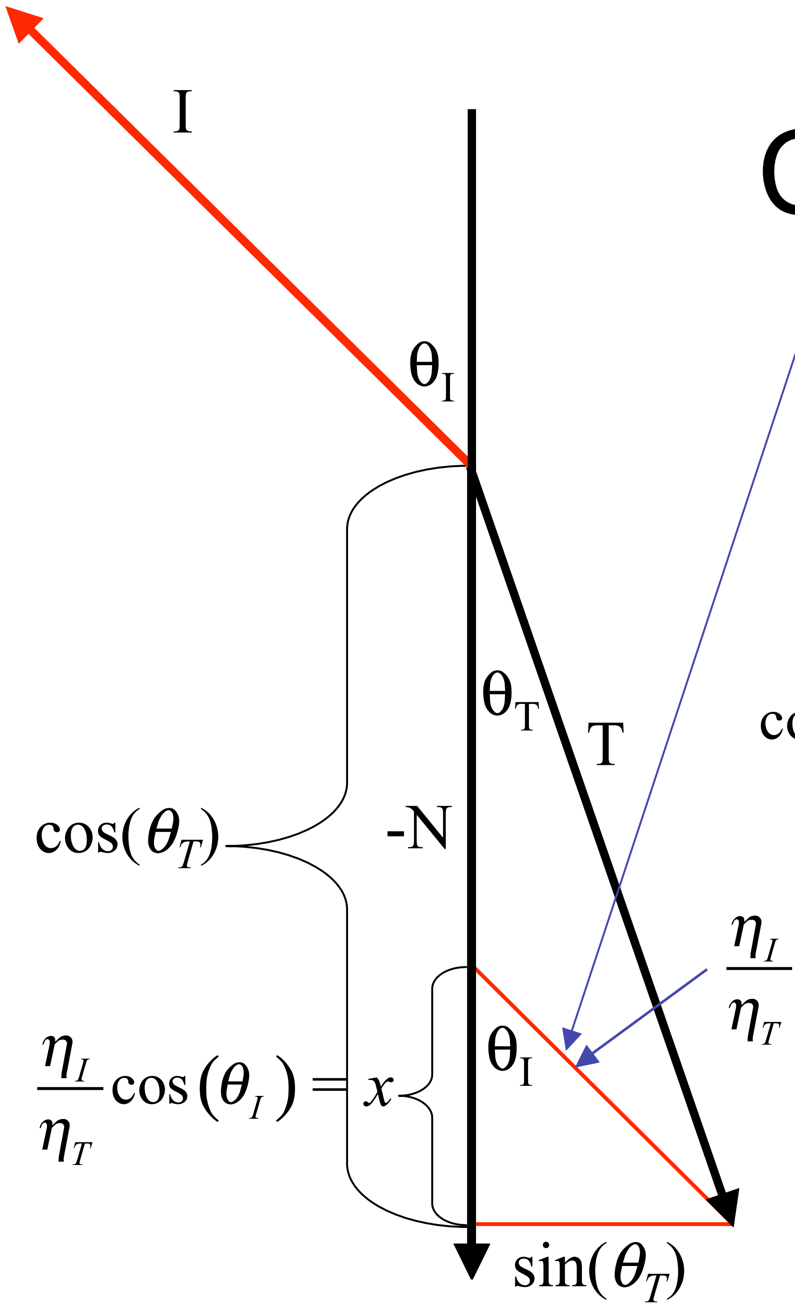


$$\frac{x}{\sin(\theta_T)} = \cot(\theta_I) = \frac{\cos(\theta_I)}{\sin(\theta_I)}$$

$$x = \cos(\theta_I)\frac{\sin(\theta_T)}{\sin(\theta_I)} = \frac{\eta_I}{\eta_T}\cos(\theta_I) \quad \text{by Snell's law.}$$

$$= \frac{x}{\cos(\theta_I)} = \frac{\eta_I}{\eta_T}$$

Computing T



Parallel to I

$$T = -\left(\cos(\theta_T) - \frac{\eta_I}{\eta_T}\cos(\theta_I)\right)N - \frac{\eta_I}{\eta_T}I$$

$$\cos(\theta_T) = \sqrt{1-\sin^2(\theta_T)} = \sqrt{1-\left(\frac{\eta_I}{\eta_T}\right)^2 \sin^2(\theta_I)}$$

$$= \sqrt{1-\left(\frac{\eta_I}{\eta_T}\right)^2 \left(1-\cos^2(\theta_I)\right)}$$

$$= \sqrt{1-\left(\frac{\eta_I}{\eta_T}\right)^2 \left(1-(N \bullet I)^2\right)}$$

Total Internal Reflection

$$\cos(\theta_T) = \sqrt{1 - \left(\frac{\eta_I}{\eta_T}\right)^2 \left(1 - (N \bullet I)^2\right)}$$

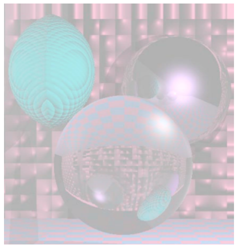
When is $\cos(\theta_T)$ defined?

When $1 - \left(\frac{\eta_I}{\eta_T}\right)^2 \left(1 - (N \bullet I)^2\right) \geq 0.$

If $\eta_I > \eta_T$ and $N \bullet I$ is close to 0, $\cos(\theta_T)$ may not be defined.

Then there is no transmitting ray and we have

total internal reflection.



Index of Refraction

The speed of all electromagnetic radiation in vacuum is the same, approximately 3×108 meters per second, and is denoted by c. Therefore, if v is the phase velocity of radiation of a specific frequency in a specific material, the refractive index is given by

$$\eta = \frac{c}{v}$$

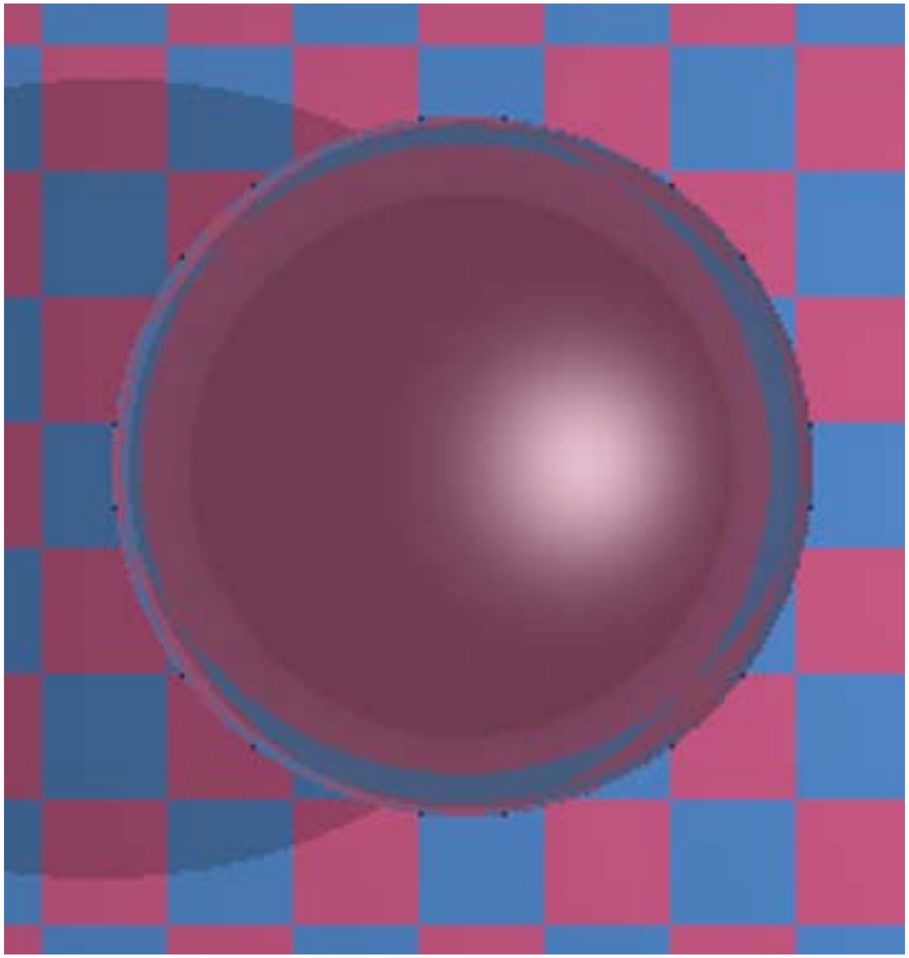
http://en.wikipedia.org/wiki/Refractive_index

Indices of Refraction

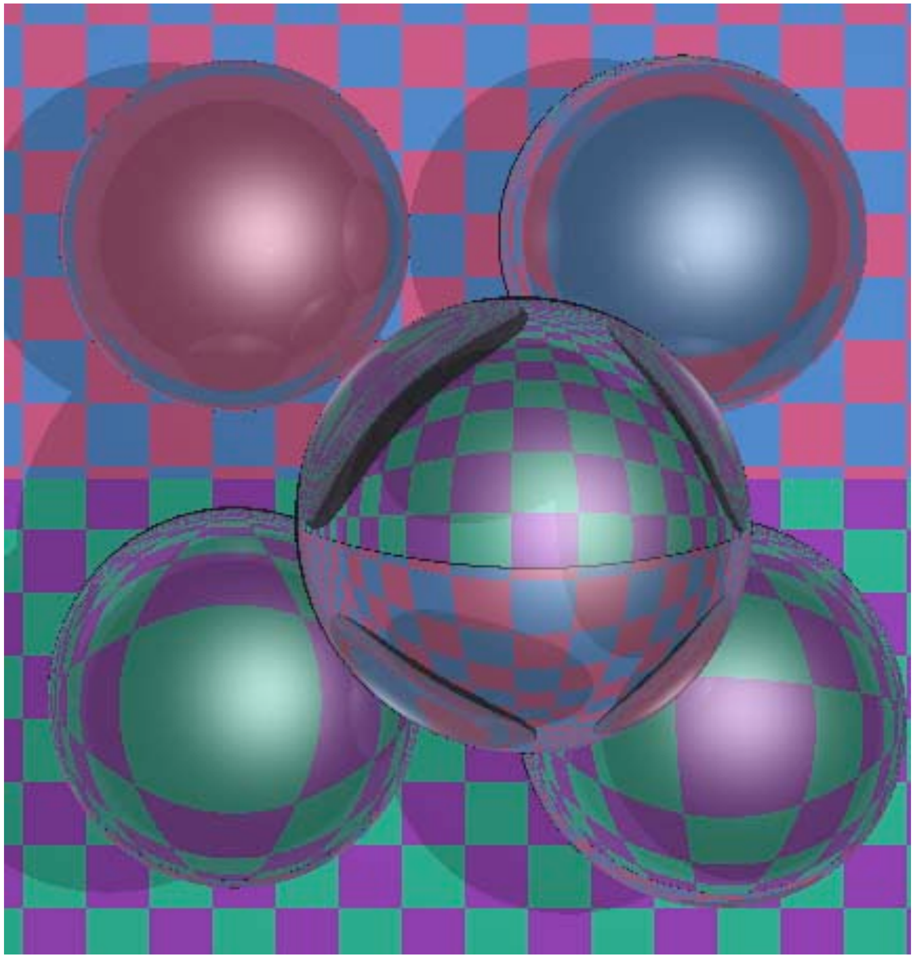
Material	**η at λ=589.3 nm**
vacuum	1 (exactly)
helium	1.000036
air at STP	1.0002926
water ice	1.31
liquid water (20°C)	1.333
ethanol	1.36
glycerine	1.4729
rock salt	1.516
glass (typical)	1.5 to 1.9
cubic zirconia	2.15 to 2.18
diamond	2.419

http://en.wikipedia.org/wiki/List_of_indices_of_refraction

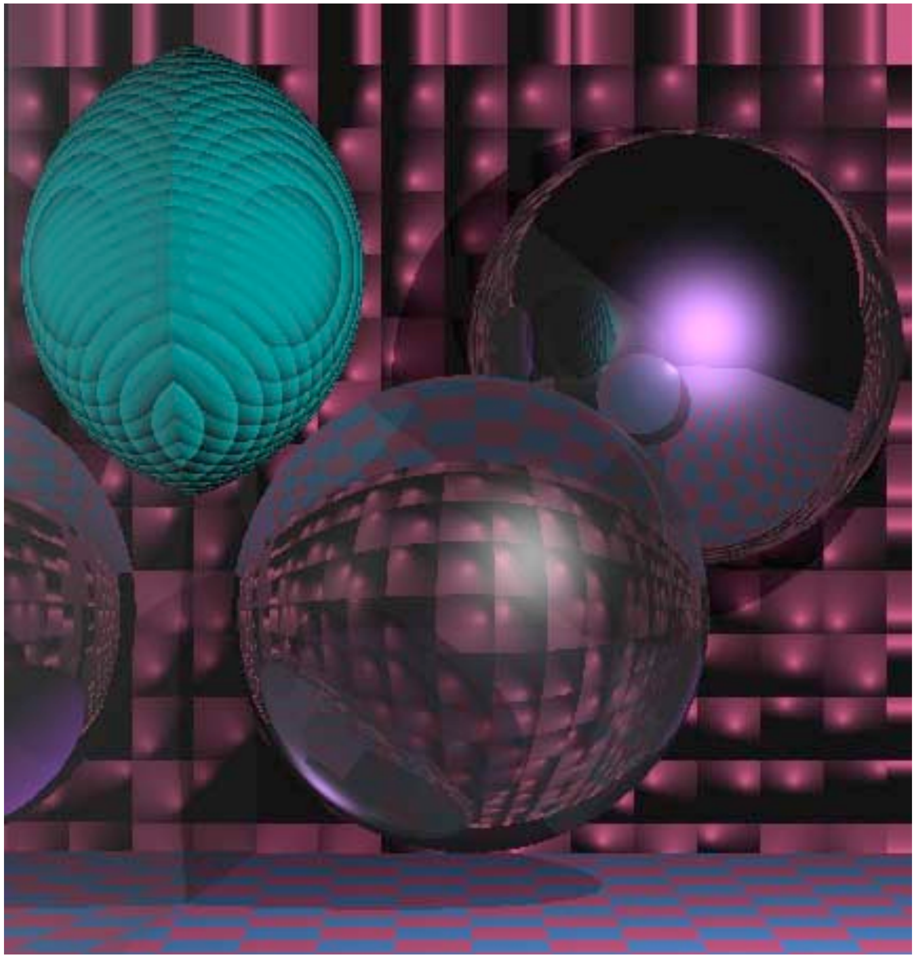
One Glass Sphere



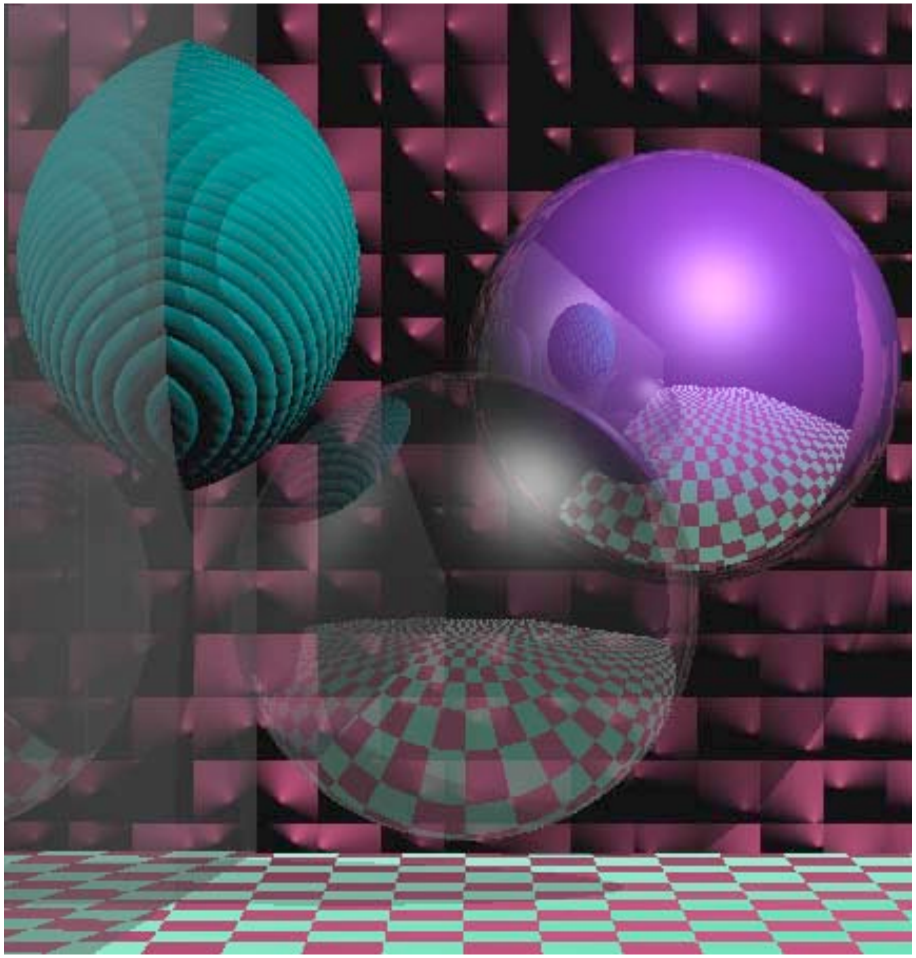
Five Glass Balls

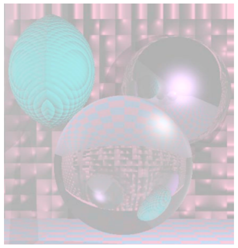


A Familiar Scene

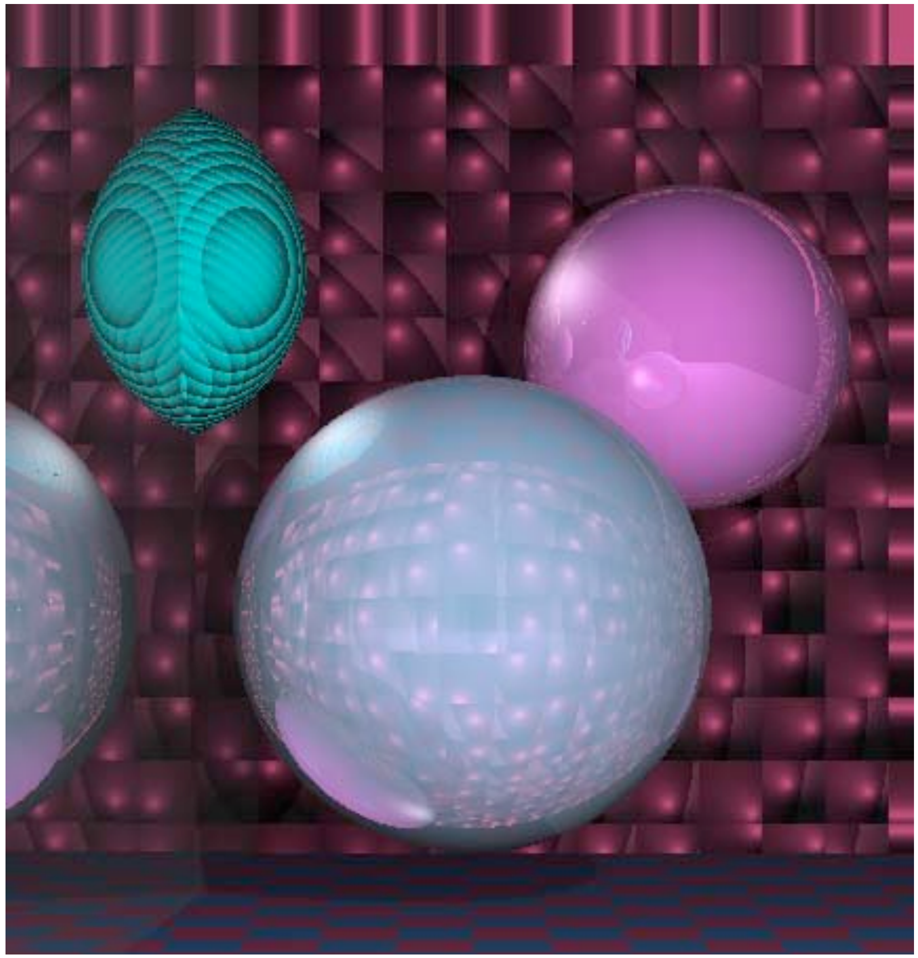


Bubble





Milky Sphere



Lens - Carl Andrews 1999