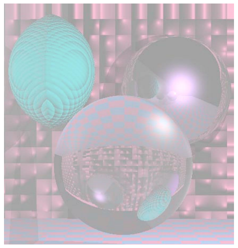
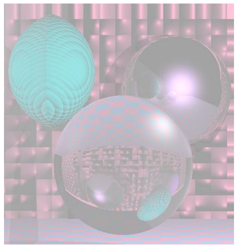


CS G140
Graduate Computer Graphics

Prof. Harriet Fell

Spring 2011

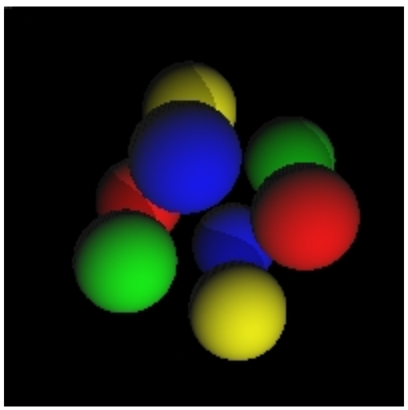
Lecture 2 – January 26, 2011

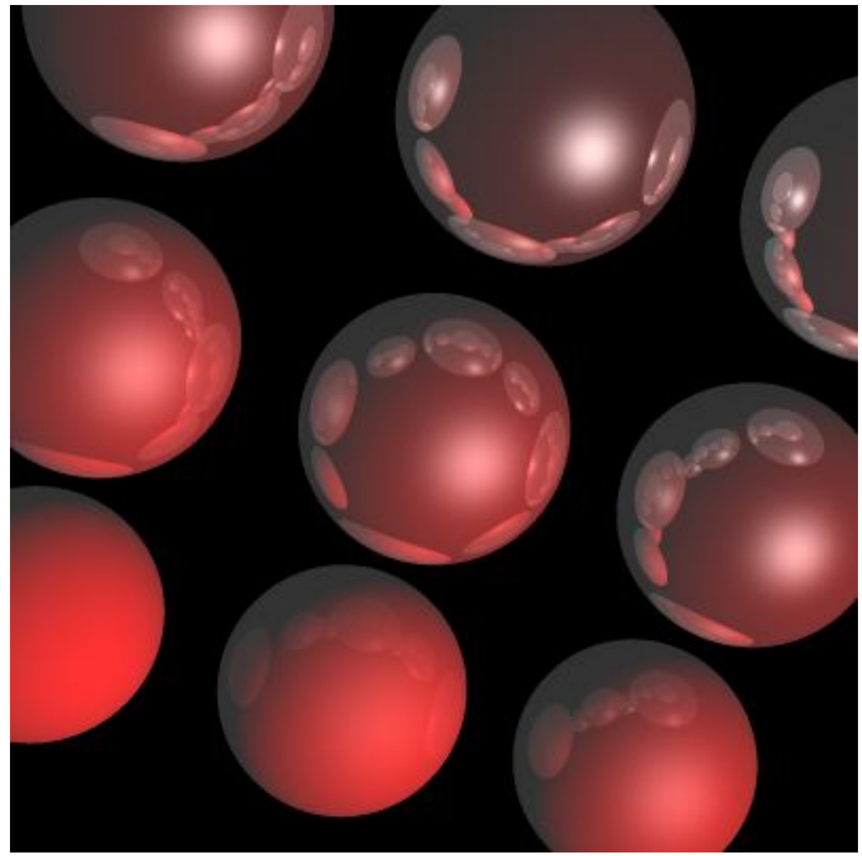


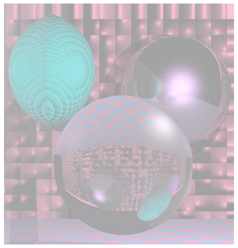
Today’s Topics

- Ray Tracing
  - Ray-Sphere Intersection
  - Light: Diffuse Reflection
  - Shadows
  - Phong Shading
- More Math
  - Matrices
  - Transformations
  - Homogeneous Coordinates

Ray Tracing a World of Spheres

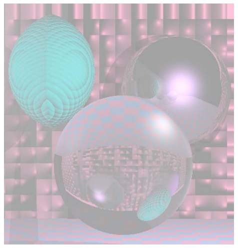




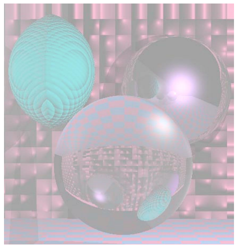


What is a Sphere

```
Vector3D   center;    // 3 doubles
double     radius;
double     R, G, B;   // for RGB colors between 0 and 1
double     kd;        // diffuse coeficient
double     ks;        // specular coefficient
int        specExp;   // specular exponent 0 if ks = 0
(double    ka;        // ambient light coefficient)
double     kgr;       // global reflection coefficient
double     kt;        // transmitting coefficient
int        pic;       // > 0 if picture texture is used
```



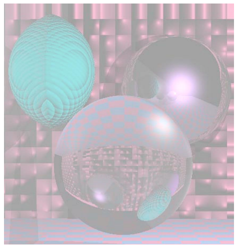
```
-.01  .01   500 800 // transform theta phi mu distance
1 // antialias
1 // numlights
100 500 800 // Lx, Ly, Lz
9 // numspheres
//cx    cy  cz  radius  R   G   B  ka  kd  ks  specExp  kgr  kt  pic
-100 -100   0  40      .9   0   0  .2  .9  .0     4       0    0   0
-100    0   0  40      .9   0   0  .2  .8  .1     8      .1    0   0
-100  100   0  40      .9   0   0  .2  .7  .2    12      .2    0   0
   0 -100   0  40      .9   0   0  .2  .6  .3    16      .3    0   0
   0    0   0  40      .9   0   0  .2  .5  .4    20      .4    0   0
   0  100   0  40      .9   0   0  .2  .4  .5    24      .5    0   0
 100 -100   0  40      .9   0   0  .2  .3  .6    28      .6    0   0
 100    0   0  40      .9   0   0  .2  .2  .7    32      .7    0   0
 100  100   0  40      .9   0   0  .2  .1  .8    36      .8    0   0
```



World of Spheres

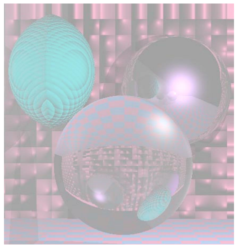
```
Vector3D VP;                    // the viewpoint
int numLights;
Vector3D theLights[5];          // up to 5 white lights
double ka;                      // ambient light coefficient
int numSpheres;
Sphere theSpheres[20];          // 20 sphere max

int ppmT[3];                    // ppm texture files
View sceneView;                 // transform data
double distance;                // view plane to VP
bool antialias;                 // if true antialias
```

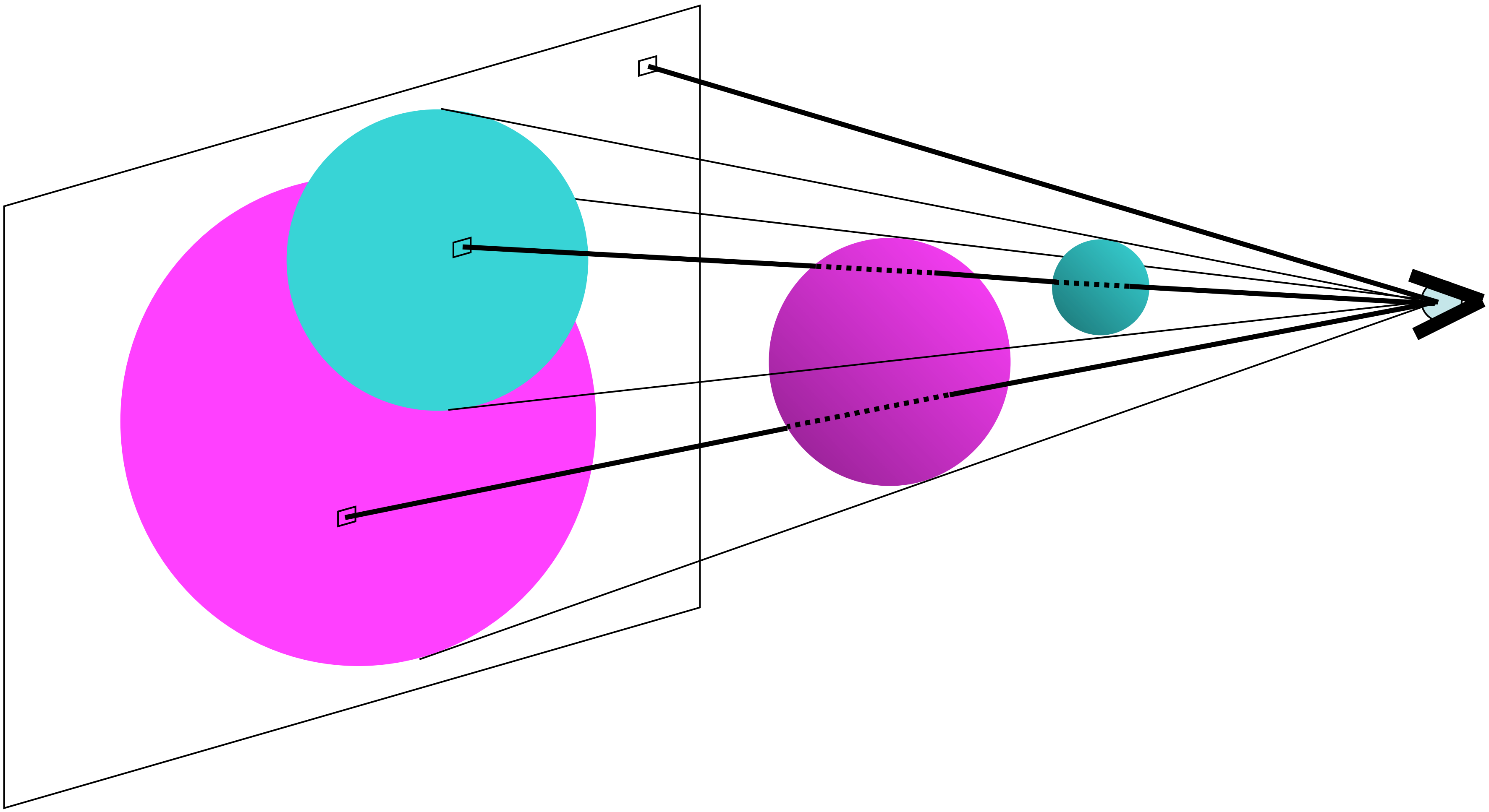



Simple Ray Casting for Detecting Visible Surfaces

```
select window on viewplane and center of projection
for (each scanline in image) {
    for (each pixel in the scanline) {
        determine ray from center of projection
            through pixel;
        for (each object in scene) {
            if (object is intersected and
                is closest considered thus far)
                    record intersection and object name;
        }
        set pixel's color to that of closest object intersected;
    }
}
```



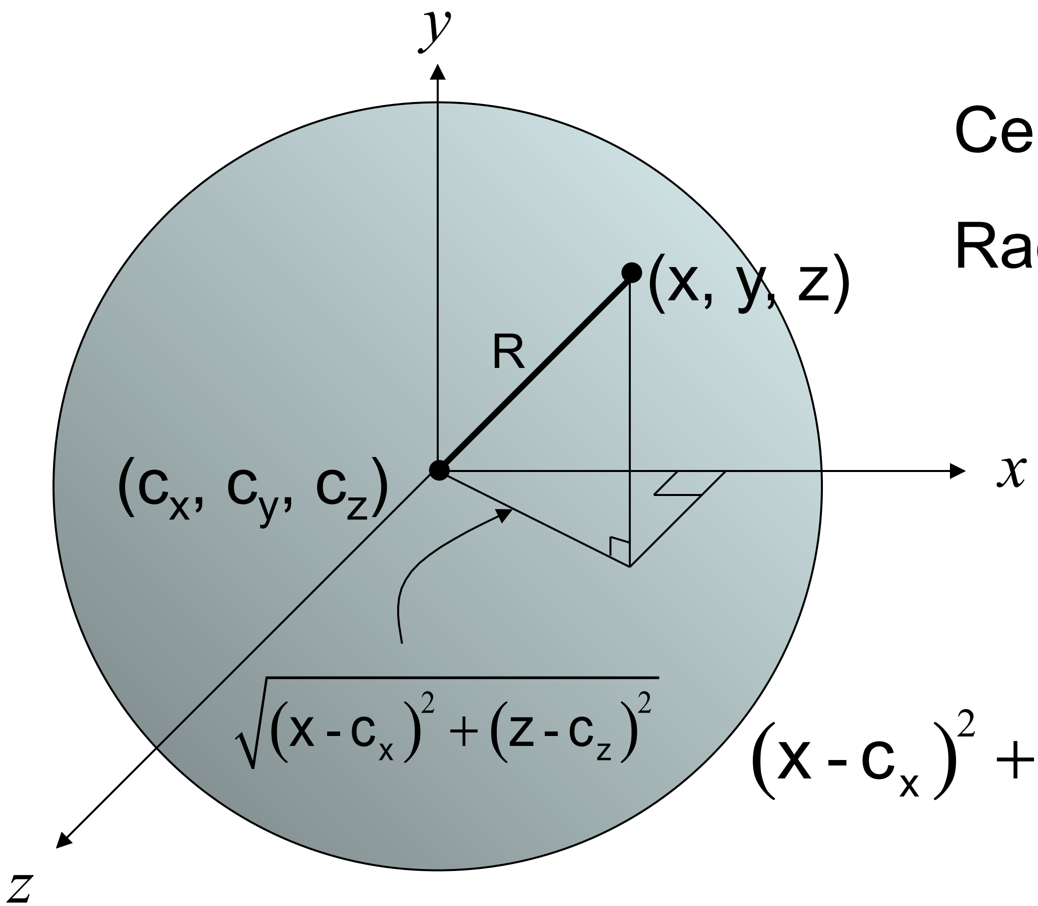
Ray Trace 1
Finding Visible Surfaces



Ray-Sphere Intersection

- Given
  - Sphere
    - Center (c_x, c_y, c_z)
    - Radius, R
  - Ray from P_0 to P_1
    - $P_0 = (x_0, y_0, z_0)$ and $P_1 = (x_1, y_1, z_1)$
  - View Point
    - (V_x, V_y, V_z)
- Project to window from $(0,0,0)$ to $(w,h,0)$

Sphere Equation



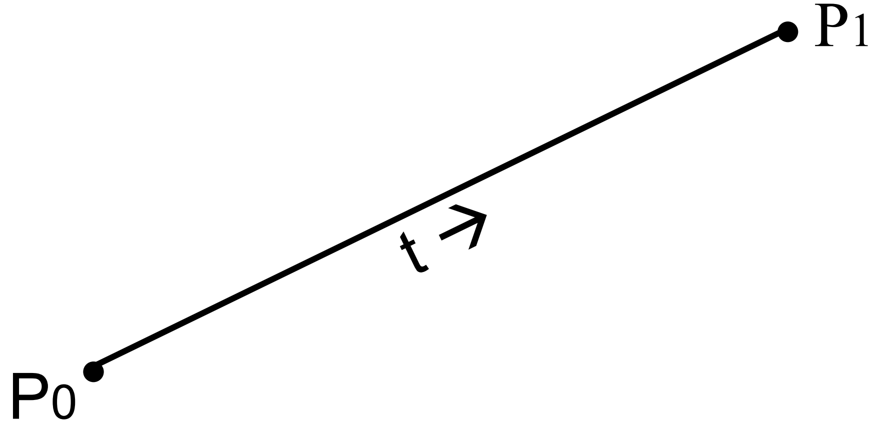
Center C = (c_x, c_y, c_z)

Radius R

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = R^2$$

Ray Equation

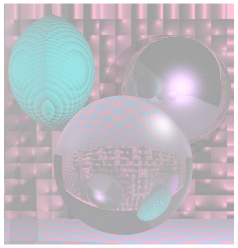
$P_0 = (x_0, y_0, z_0)$ and $P_1 = (x_1, y_1, z_1)$



The ray from P_0 to P_1 is given by:

$P(t) = (1 - t)P_0 + tP_1$ $\quad\quad 0 <= t <= 1$

$\quad = P_0 + t(P_1 - P_0)$



Intersection Equation

$P(t) = P_0 + t(P_1 - P_0)$ $\qquad\qquad$ $0 <= t <= 1$

is really three equations

$x(t) = x_0 + t(x_1 - x_0)$

$y(t) = y_0 + t(y_1 - y_0)$

$z(t) = z_0 + t(z_1 - z_0)$ $\qquad\qquad$ $0 <= t <= 1$

Substitute x(t), y(t), and z(t) for x, y, z, respectively in

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = R^2$$

$$((x_0 + t(x_1 - x_0)) - c_x)^2 + ((y_0 + t(y_1 - y_0)_1) - c_y)^2 + ((z_0 + t(z_1 - z_0)) - c_z)^2 = R^2$$

Solving the Intersection Equation

$$\left(\left(x_0 + t(x_1 - x_0)\right) - c_x\right)^2 + \left(\left(y_0 + t(y_1 - y_0)_1\right) - c_y\right)^2 + \left(\left(z_0 + t(z_1 - z_0)\right) - c_z\right)^2 = R^2$$

is a quadratic equation in variable t.

For a fixed pixel, VP, and sphere,

$\boxed{x_0, y_0, z_0,}$ $\boxed{x_1, y_1, z_1,}$ $\boxed{c_x, c_y, c_z, \text{ and } R}$

are all constants.

We solve for t using the quadratic formula.

The Quadratic Coefficients

$$\left(\left(x_0 + t\left(x_1 - x_0\right)\right) - c_x\right)^2 + \left(\left(y_0 + t\left(y_1 - y_0\right)_1\right) - c_y\right)^2 + \left(\left(z_0 + t\left(z_1 - z_0\right)\right) - c_z\right)^2 = R^2$$

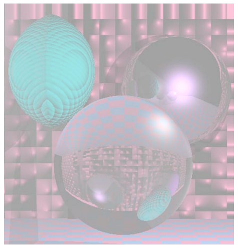
Set $d_x = x_1 - x_0$

$d_y = y_1 - y_0$

$d_z = z_1 - z_0$

Now find the the coefficients:

$$At^2 + Bt + C = 0$$



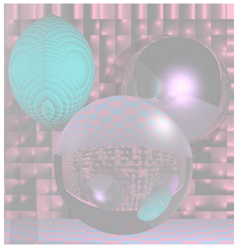
Computing Coefficients

$$\left(\left(x_0 + t(x_1 - x_0)\right) - c_x\right)^2 + \left(\left(y_0 + t(y_1 - y_0)\right) - c_y\right)^2 + \left(\left(z_0 + t(z_1 - z_0)\right) - c_z\right)^2 = R^2$$

$$\left(\left(x_0 + td_x\right) - c_x\right)^2 + \left(\left(y_0 + td_y\right) - c_y\right)^2 + \left(\left(\left(z_0 + td_z\right) - c_z\right)^2 = R^2$$

$$\begin{aligned}
&(x_0 + td_x)^2 - 2c_x(x_0 + td_x) + c_x{}^2 + \\
&(y_0 + td_y)^2 - 2c_y(y_0 + td_y) + c_y{}^2 + \\
&(z_0 + td_z)^2 - 2c_z(z_0 + td_z) + c_z{}^2 - R^2 = 0
\end{aligned}$$

$$\begin{aligned}
&x_0{}^2 + 2x_0td_x + t^2d_x{}^2 - 2c_xx_0 - 2c_xtd_x + c_x{}^2 + \\
&y_0{}^2 + 2y_0td_y + t^2d_y{}^2 - 2c_yy_0 - 2c_ytd_y + c_y{}^2 + \\
&z_0{}^2 + 2z_0td_z + t^2d_z{}^2 - 2c_zz_0 - 2c_ztd_z + c_z{}^2 - R^2 = 0
\end{aligned}$$



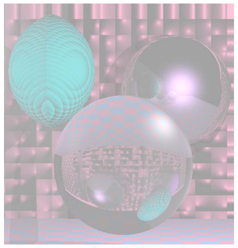
The Coefficients

$$x_0^2 + 2x_0td_x + t^2d_x^2 - 2c_xx_0 - 2c_xtd_x + c_x^2 +$$
$$y_0^2 + 2y_0td_y + t^2d_y^2 - 2c_yy_0 - 2c_ytd_y + c_y^2 +$$
$$z_0^2 + 2z_0td_z + t^2d_z^2 - 2c_zz_0 - 2c_ztd_z + c_z^2 - R^2 = 0$$

$$A = d_x^2 + d_y^2 + d_z^2$$

$$B = 2d_x(x_0 - c_x) + 2d_y(y_0 - c_y) + 2d_z(z_0 - c_z)$$

$$C = c_x^2 + c_y^2 + c_z^2 + x_0^2 + y_0^2 + z_0^2 + -2(c_xx_0 + c_yy_0 + c_zz_0) - R^2$$



Solving the Equation

$$At^2 + Bt + C = 0$$

$$\text{discriminant} = D(A,B,C) = B^2 - 4AC$$

$$D(A,B,C)\begin{cases} < 0 & \text{no intersection} \\ = 0 & \text{ray is tangent to the sphere} \\ > 0 & \text{ray intersects sphere in two points} \end{cases}$$

The intersection nearest P_0 is given by:

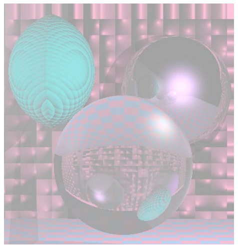
$$t = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

To find the coordinates of the intersection point:

$$x = x_0 + td_x$$

$$y = y_0 + td_y$$

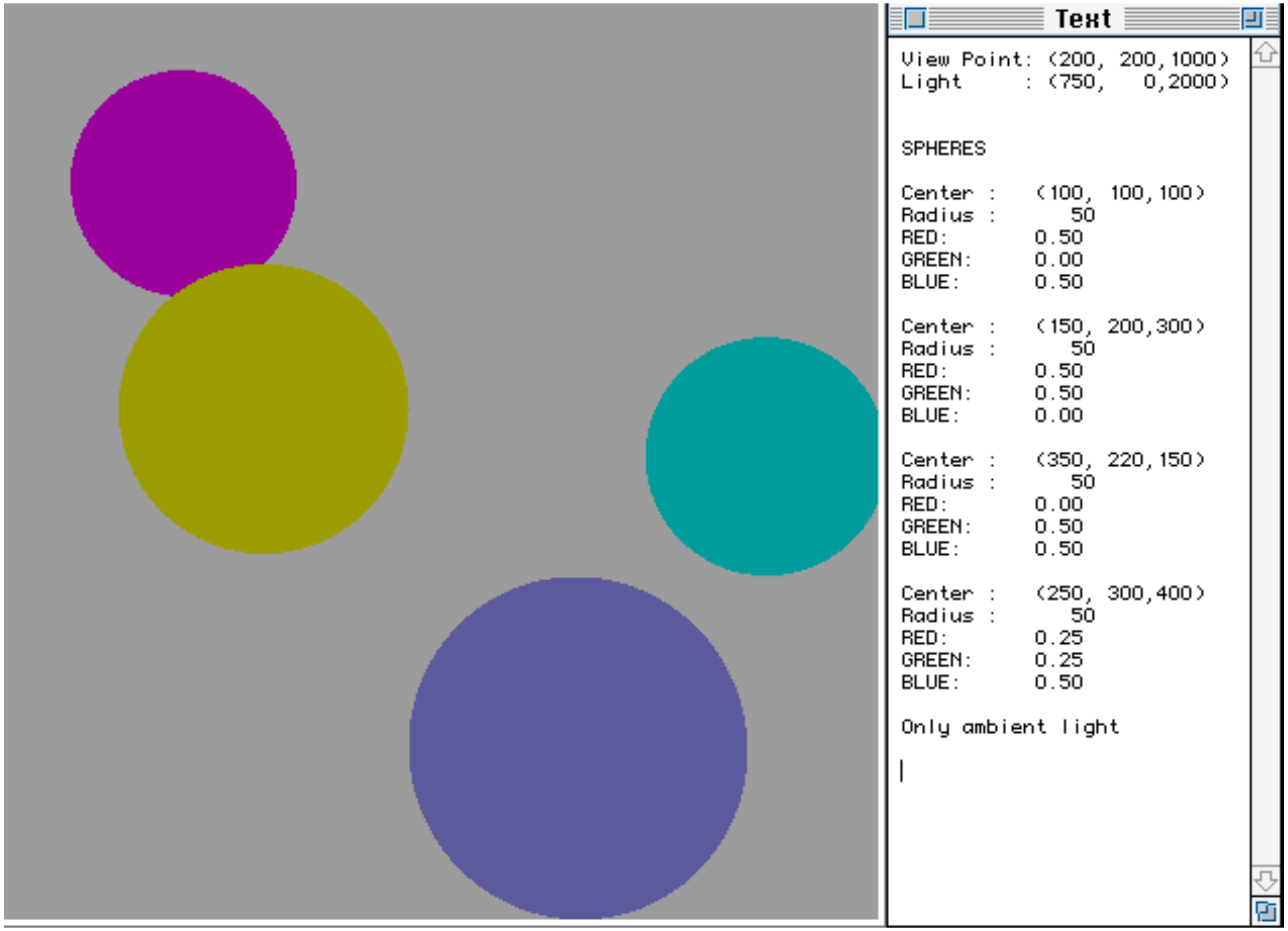
$$z = z_0 + td_z$$

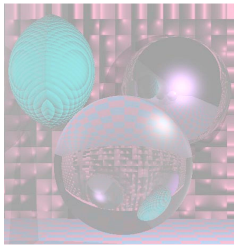


First Lighting Model

- Ambient light is a global constant.
  Ambient Light = $k_a (A_R, A_G, A_B)$
  k_a is in the "World of Spheres"
  $0 \le k_a \le 1$
  (A_R, A_G, A_B) = average of the light sources
    $(A_R, A_G, A_B) = (1, 1, 1)$ for white light
- Color of object $S = (S_R, S_G, S_B)$
- Visible Color of an object S with only ambient light
  $C_S = k_a (A_R S_R, A_G S_G, A_B S_B)$
- For white light
  $C_S = k_a (S_R, S_G, S_B)$

Visible Surfaces
Ambient Light





Second Lighting Model

- Point source light $L = (L_R, L_G, L_B)$ at (L_x, L_y, L_z)
- Ambient light is also present.
- Color at **point p** on an object S with ambient & diffuse reflection

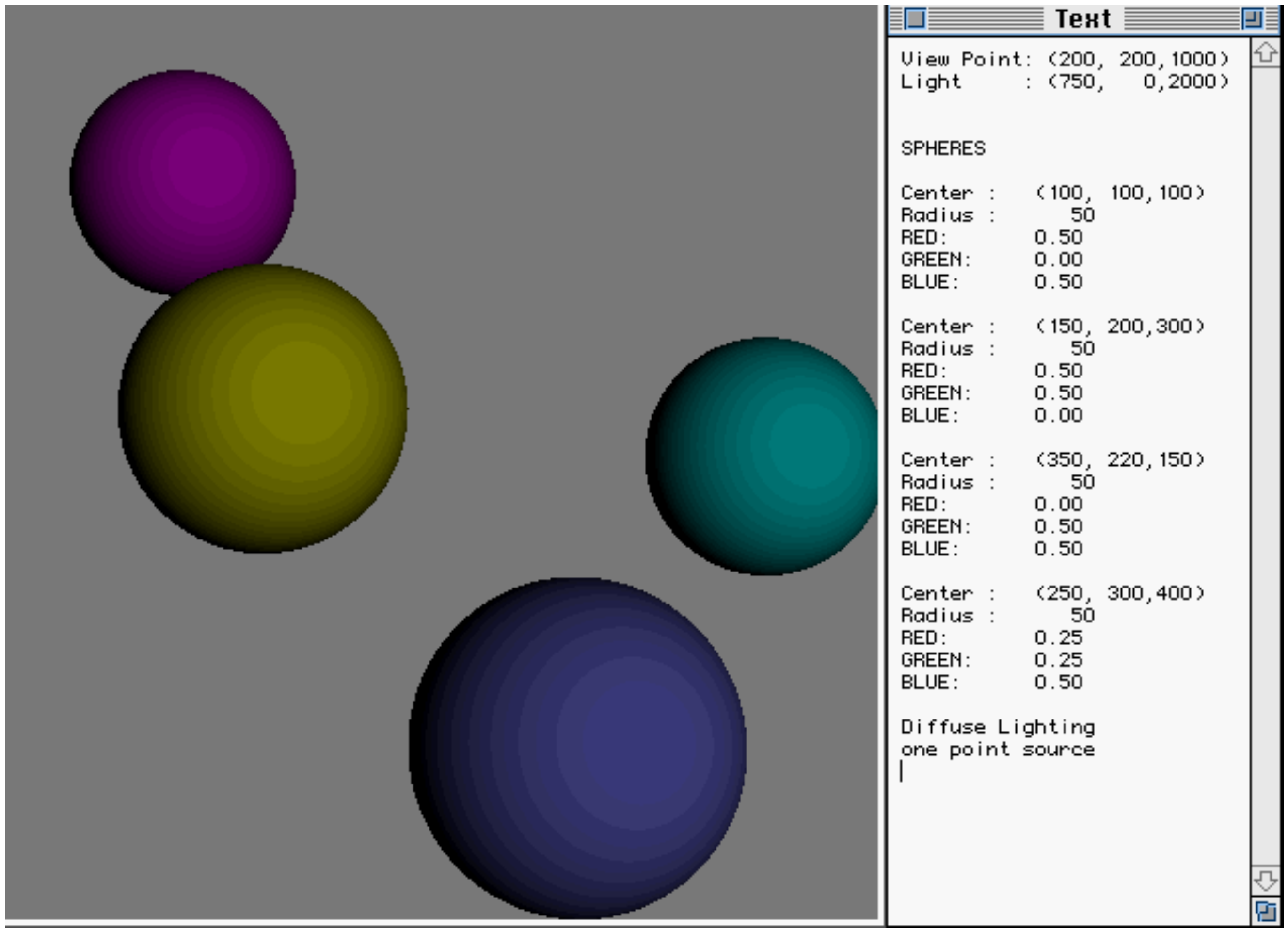
$$C_p = k_a (A_R S_R, A_G S_G, A_B S_B) + k_d k_p (L_R S_R, L_G S_G, L_B S_B)$$

- For white light, L = (1, 1, 1)

$$C_p = k_a (S_R, S_G, S_B) + k_d k_p (S_R, S_G, S_B)$$

- k_p depends on the **point p** on the object and (L_x, L_y, L_z)
- k_d depends on the object (sphere)
- k_a is global
- $k_a + k_d <= 1$

Diffuse Light

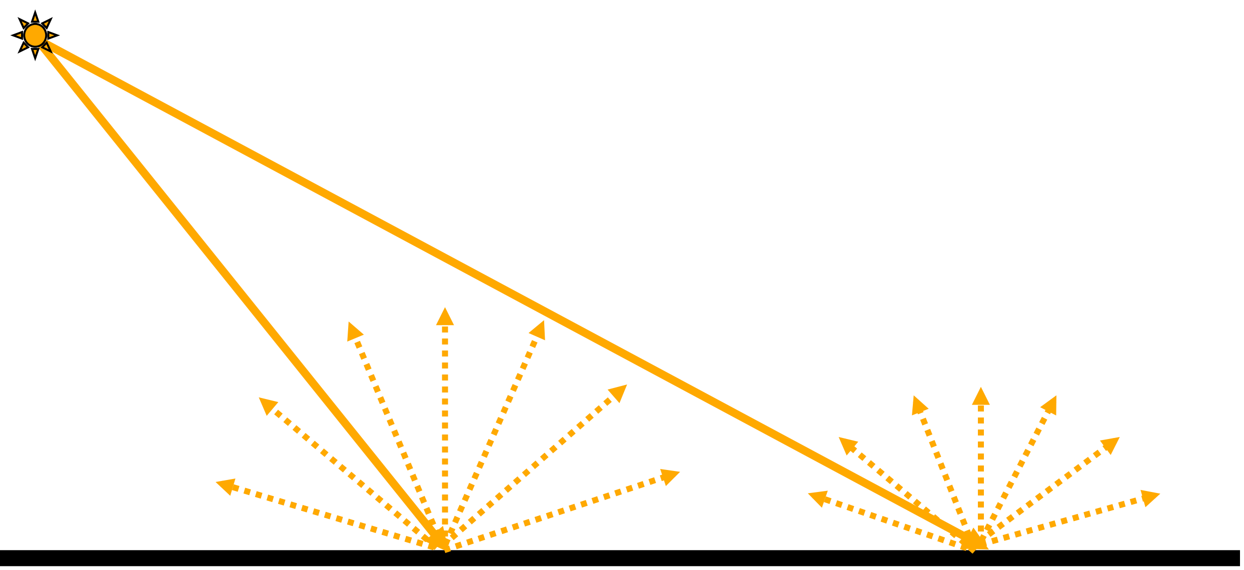


Lambertian Reflection Model
Diffuse Shading

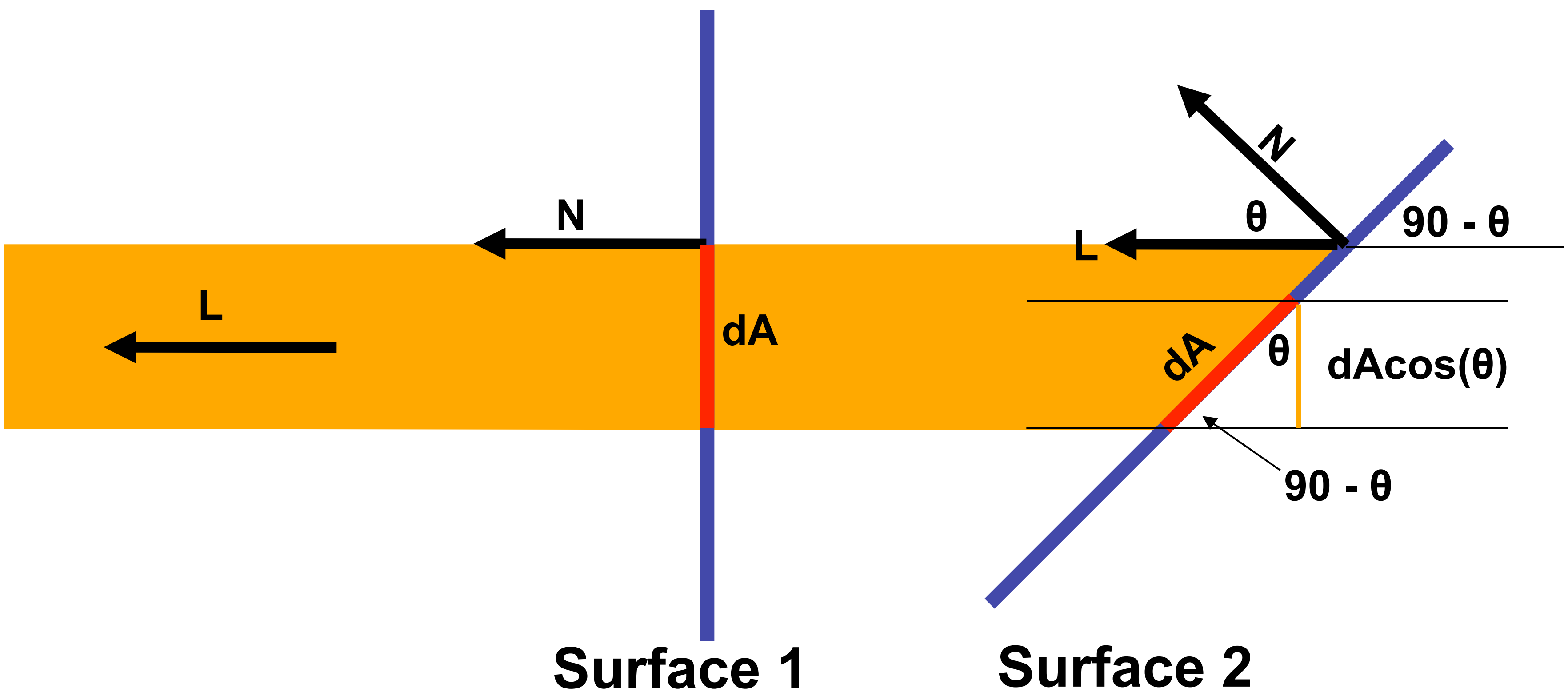
- For matte (non-shiny) objects
- Examples
  - Matte paper, newsprint
  - Unpolished wood
  - Unpolished stones
- Color at a point on a matte object does not change with viewpoint.

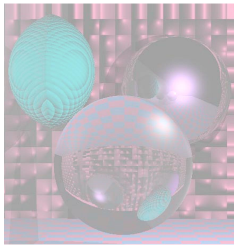
Physics of Lambertian Reflection

- Incoming light is partially absorbed and partially transmitted equally in all directions

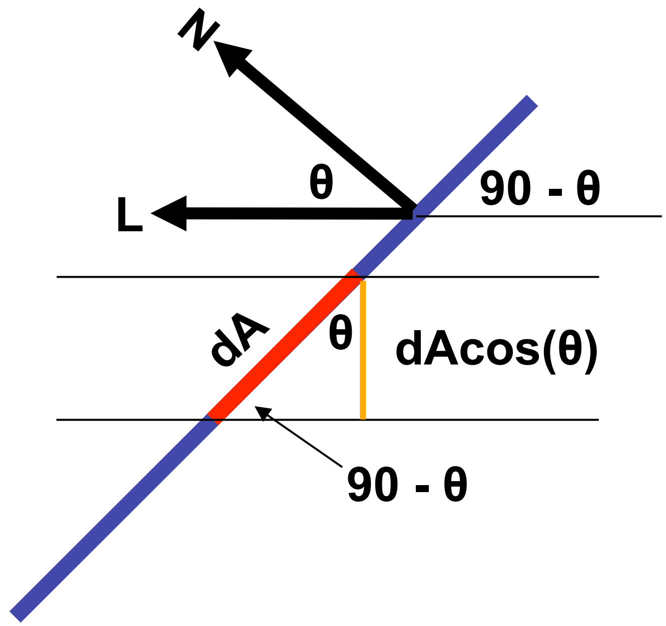


Geometry of Lambert's Law





$Cos(\theta)=N \bullet L$

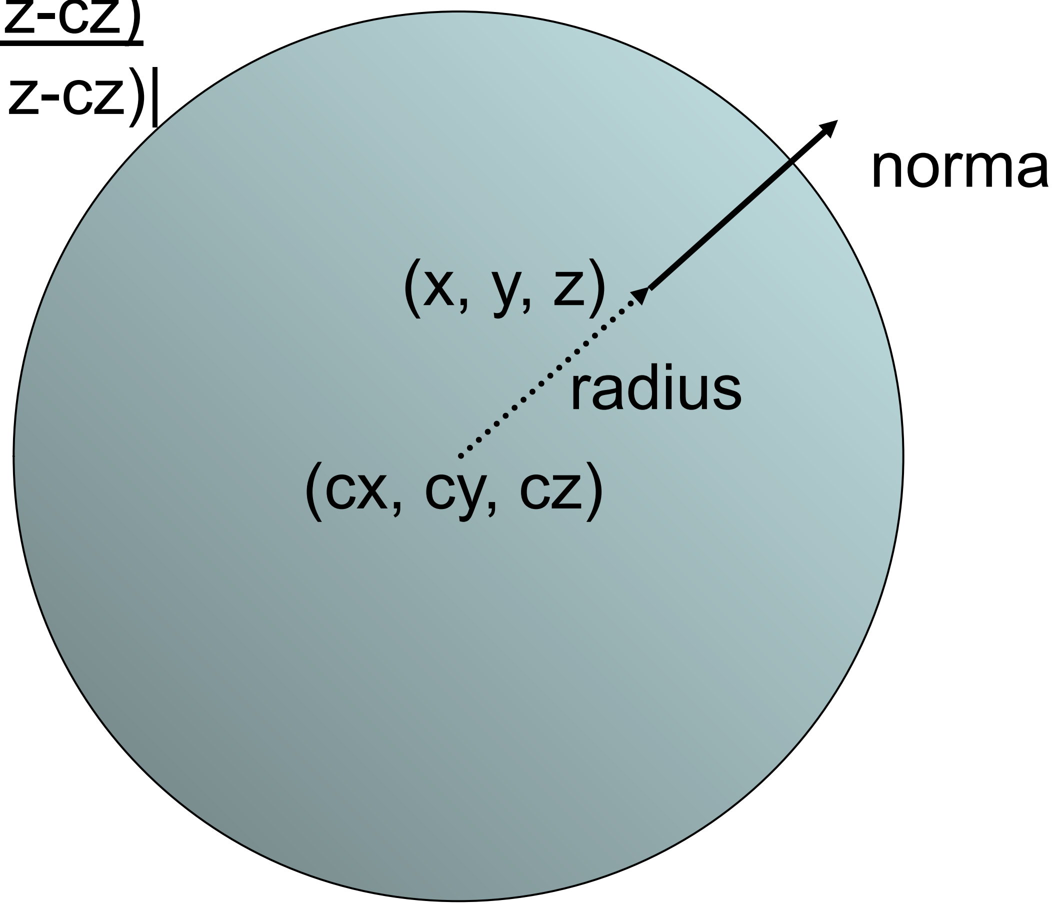


Surface 2

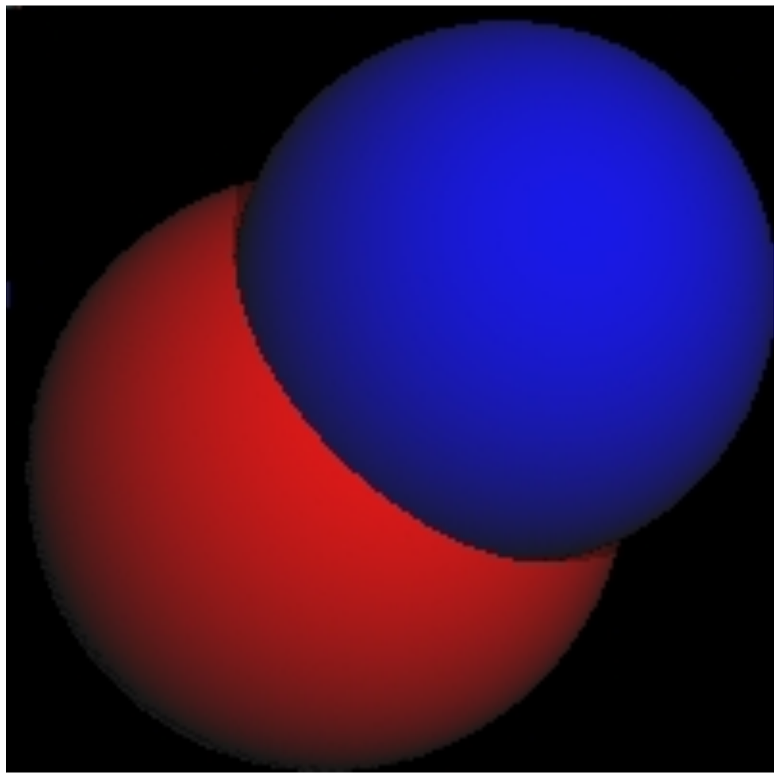
Cp= ka (SR, SG, SB) + kd **N•L** (SR, SG, SB)

Finding N

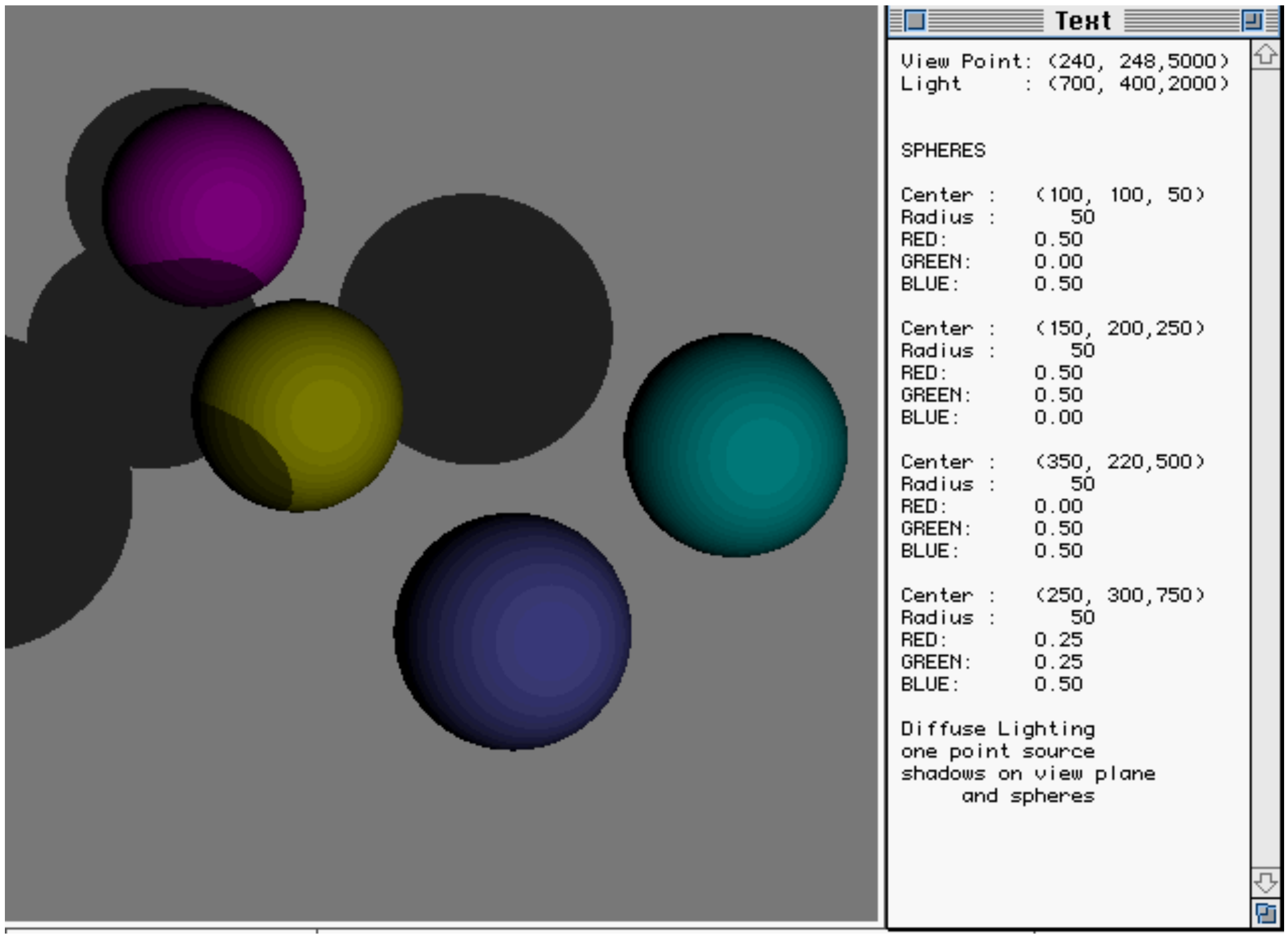
$$\mathbf{N} = \frac{(x-cx,\ y-cy,\ z-cz)}{|(x-cx,\ y-cy,\ z-cz)|}$$



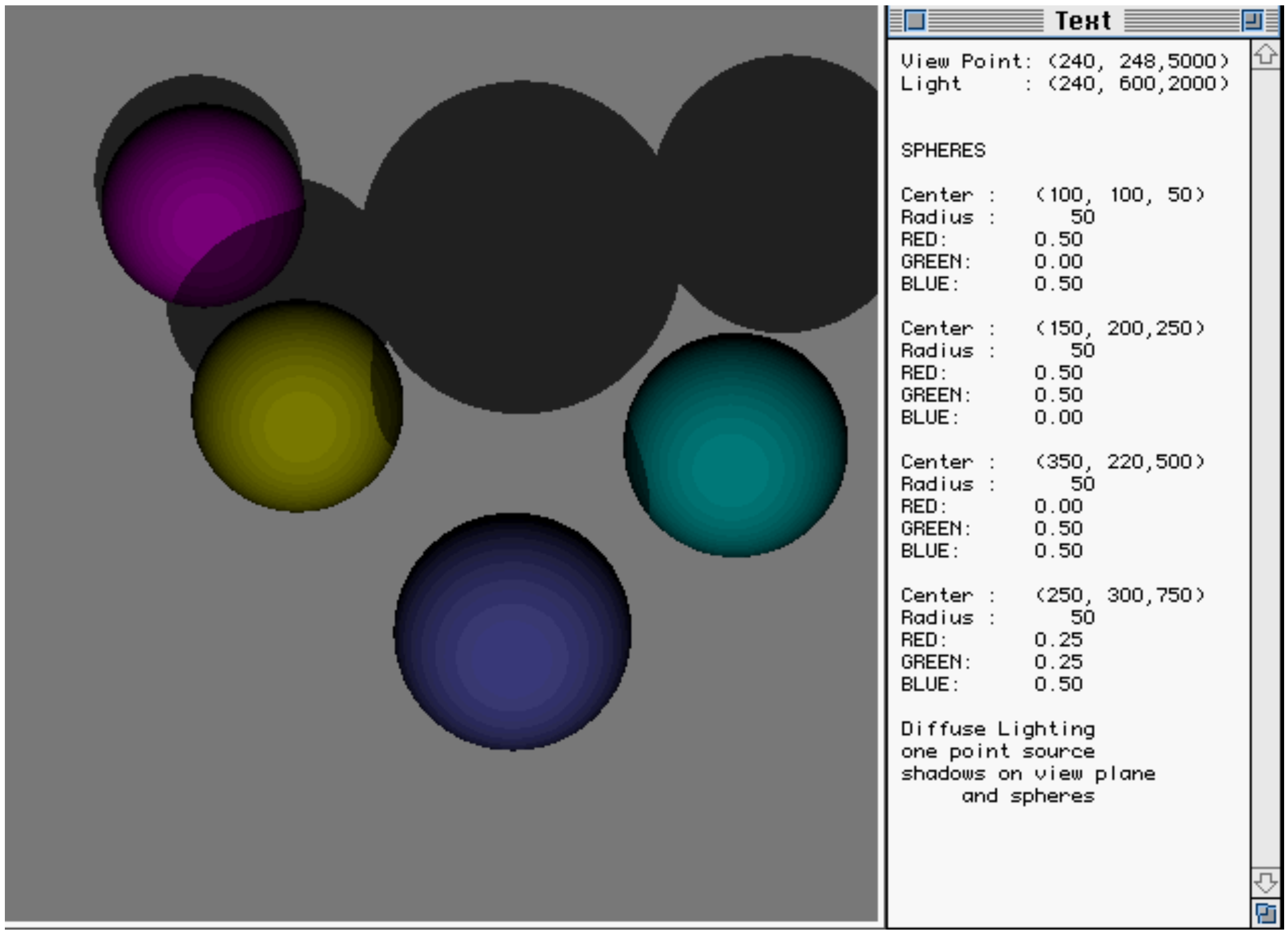
Diffuse Light 2



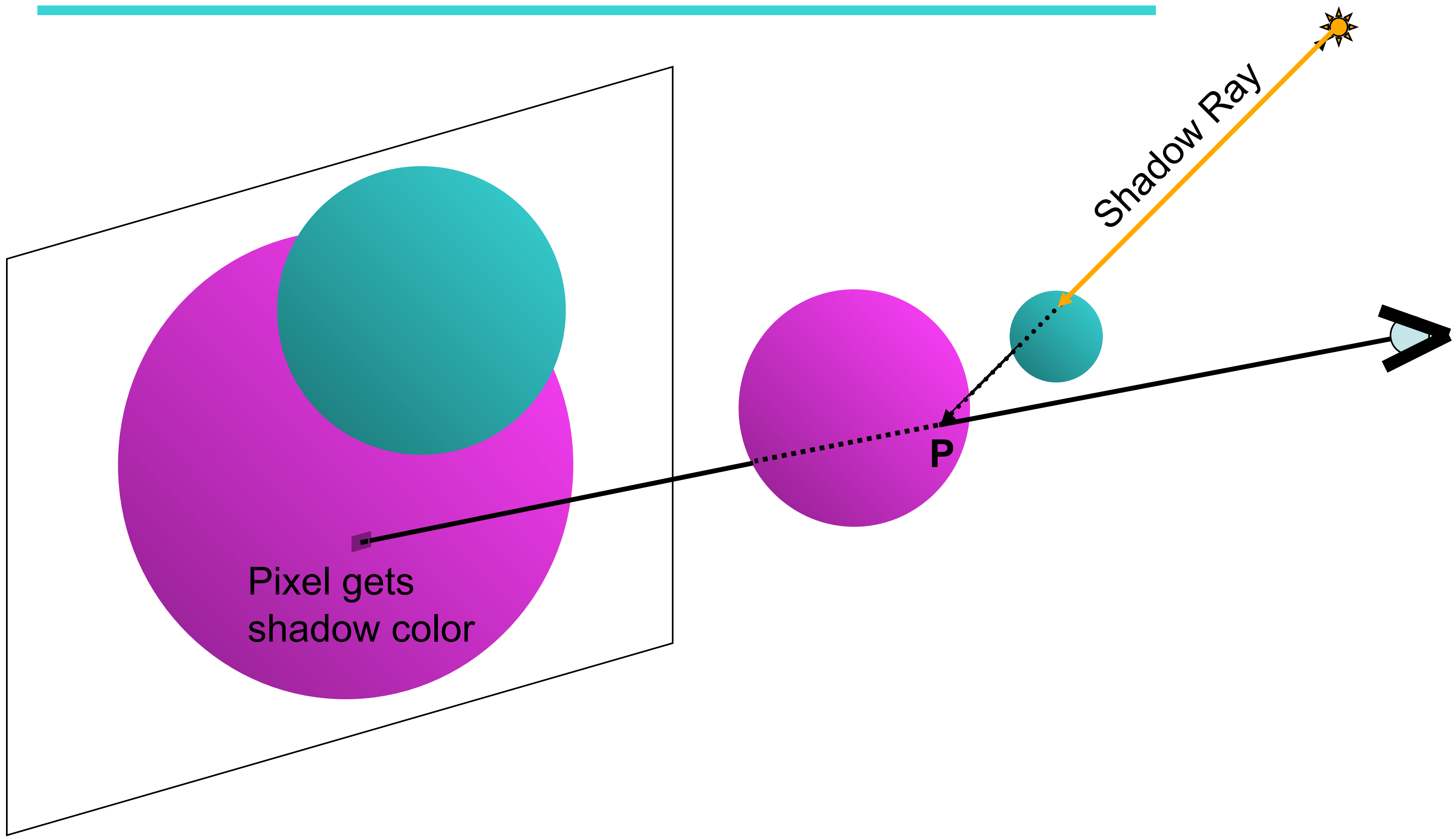
Shadows on Spheres



More Shadows



Finding Shadows



Shadow Color

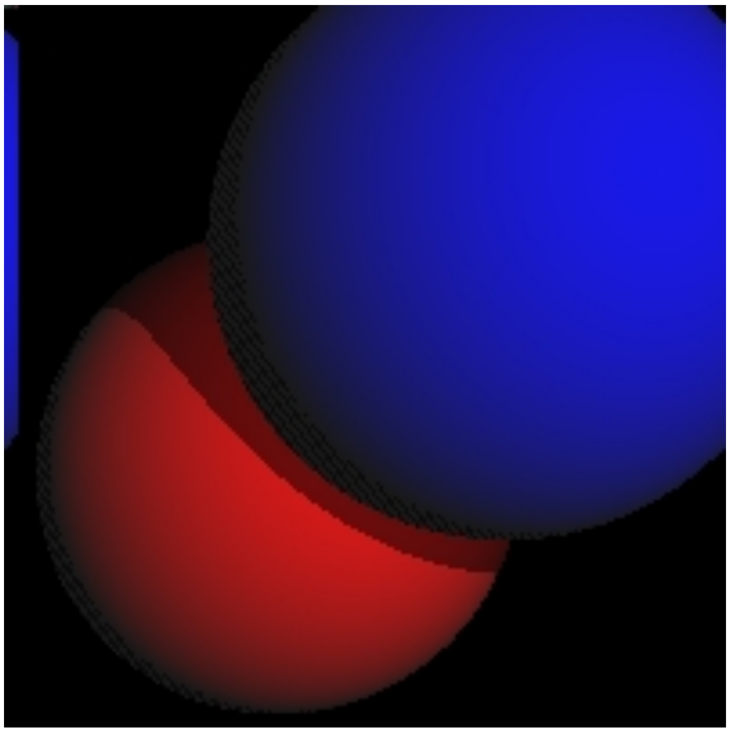
- Given

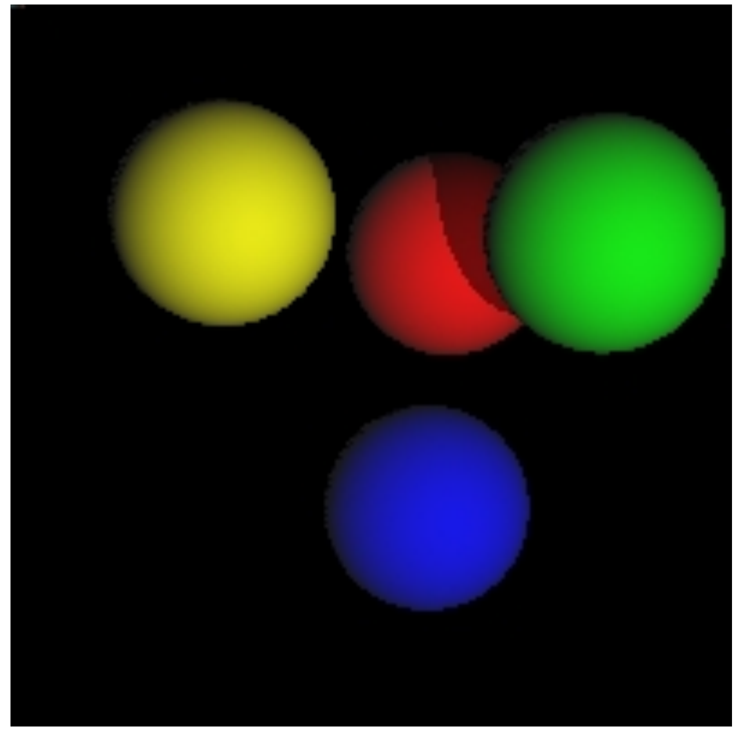
  Ray from P (point on sphere S) to L (light)

  $P = P_0 = (x_0, y_0, z_0)$ and $L = P_1 = (x_1, y_1, z_1)$

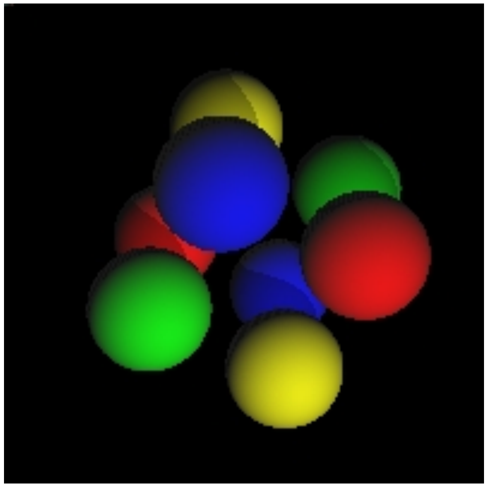
- Find out whether the ray intersects any other object (sphere).
  - If it does, P is in shadow.
  - Use only ambient light for pixel.

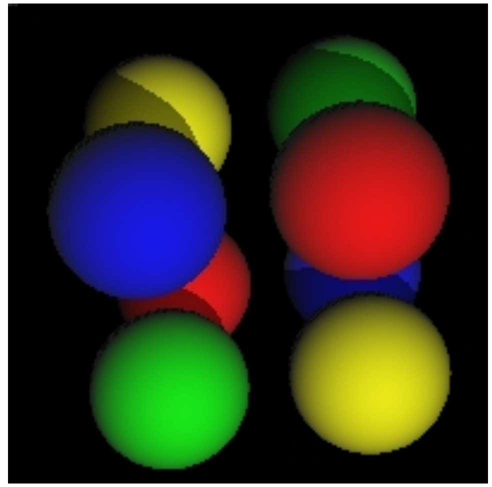
Shape of Shadows

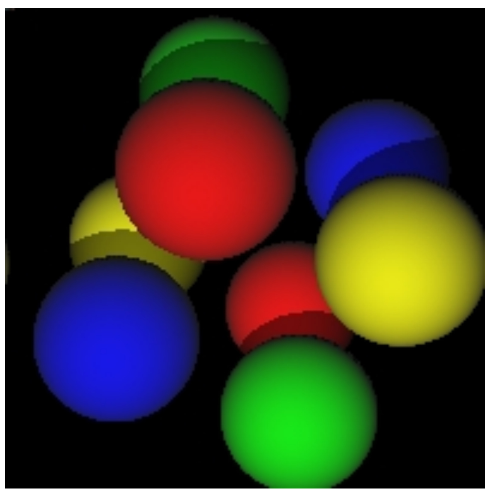


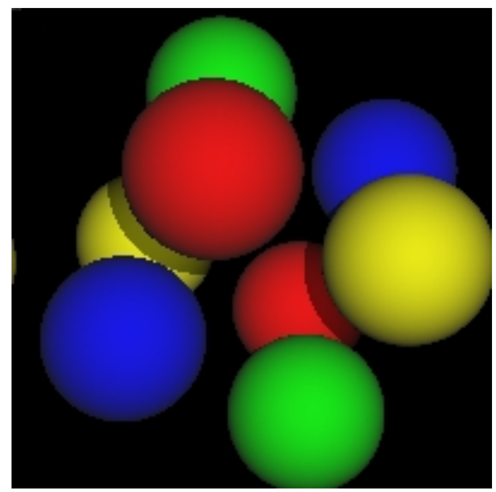


Different Views

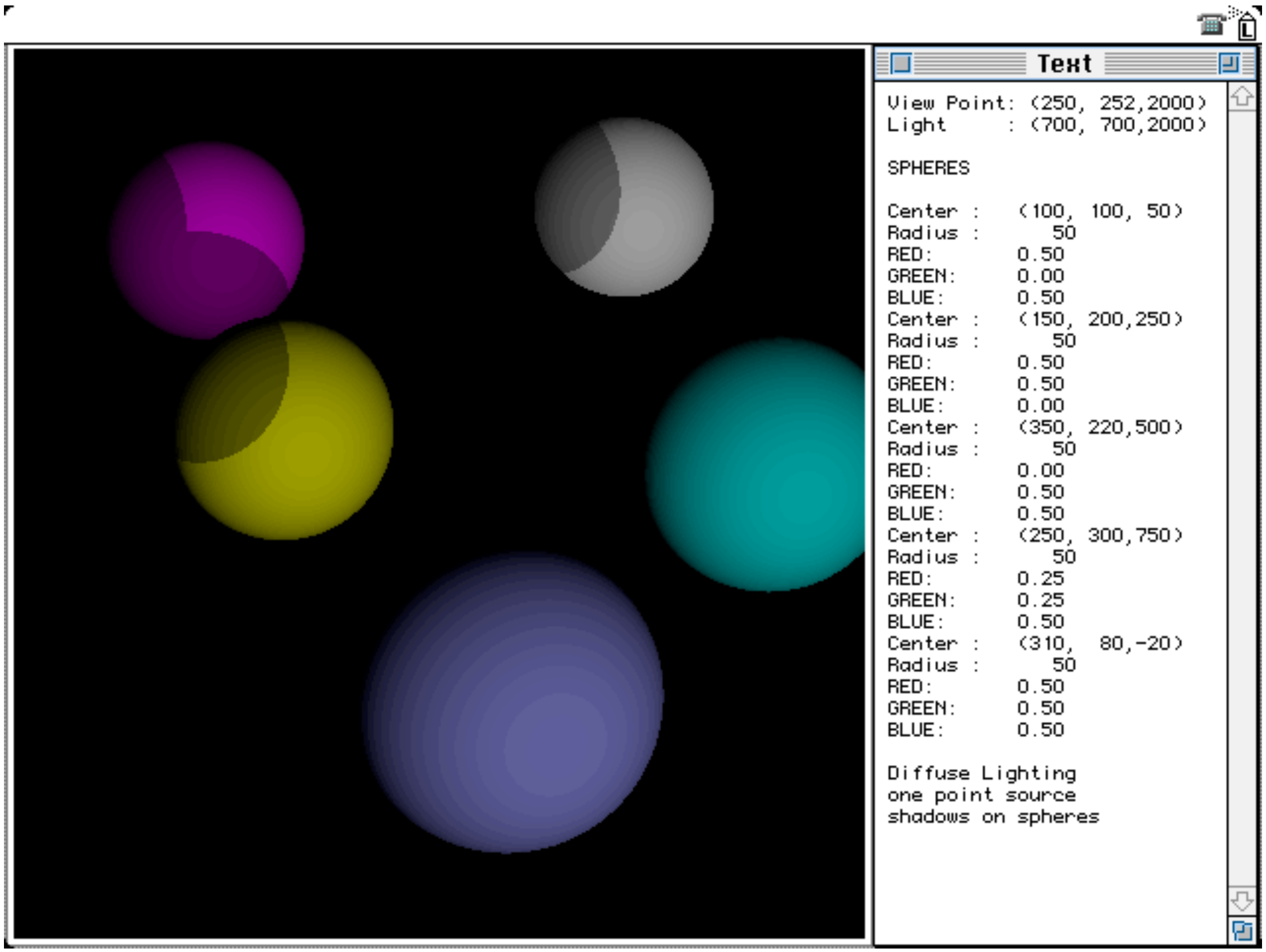




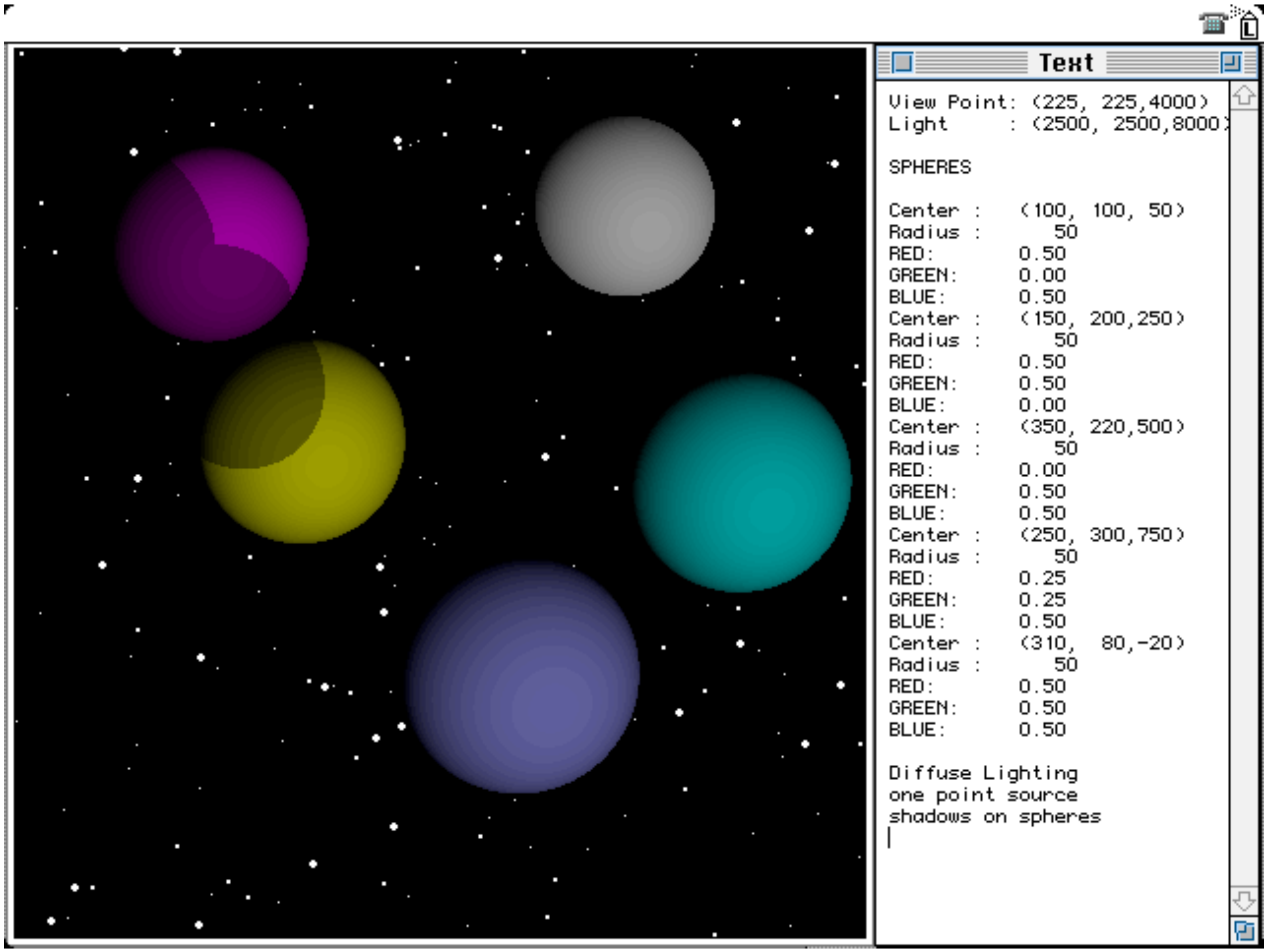




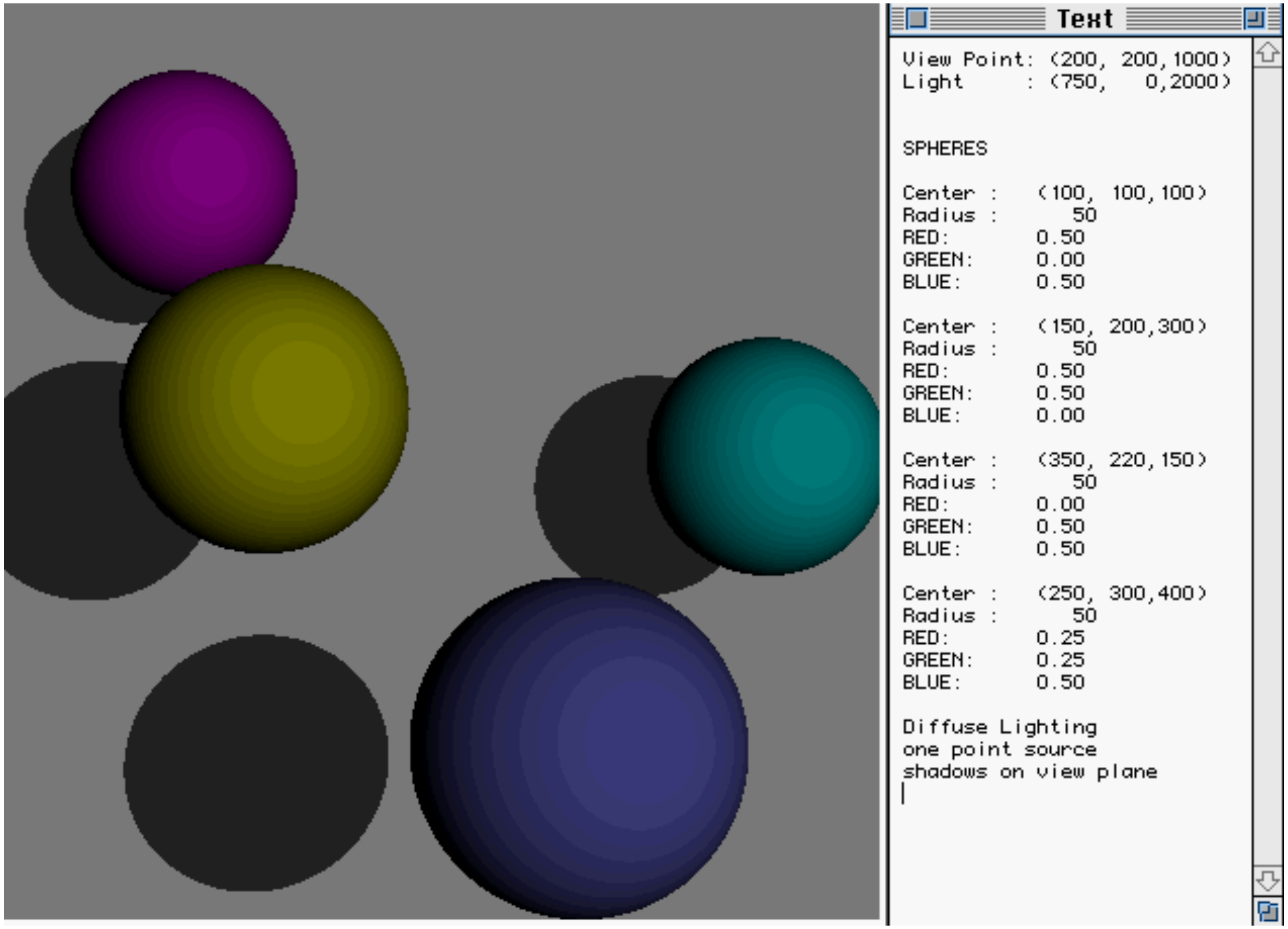
Planets


Text
View Point: (250, 252,2000)
Light : (700, 700,2000)
SPHERES
Center : (100, 100, 50)
Radius : 50
RED: 0.50
GREEN: 0.00
BLUE: 0.50
Center : (150, 200,250)
Radius : 50
RED: 0.50
GREEN: 0.50
BLUE: 0.00
Center : (350, 220,500)
Radius : 50
RED: 0.00
GREEN: 0.50
BLUE: 0.50
Center : (250, 300,750)
Radius : 50
RED: 0.25
GREEN: 0.25
BLUE: 0.50
Center : (310, 80,-20)
Radius : 50
RED: 0.50
GREEN: 0.50
BLUE: 0.50
Diffuse Lighting
one point source
shadows on spheres

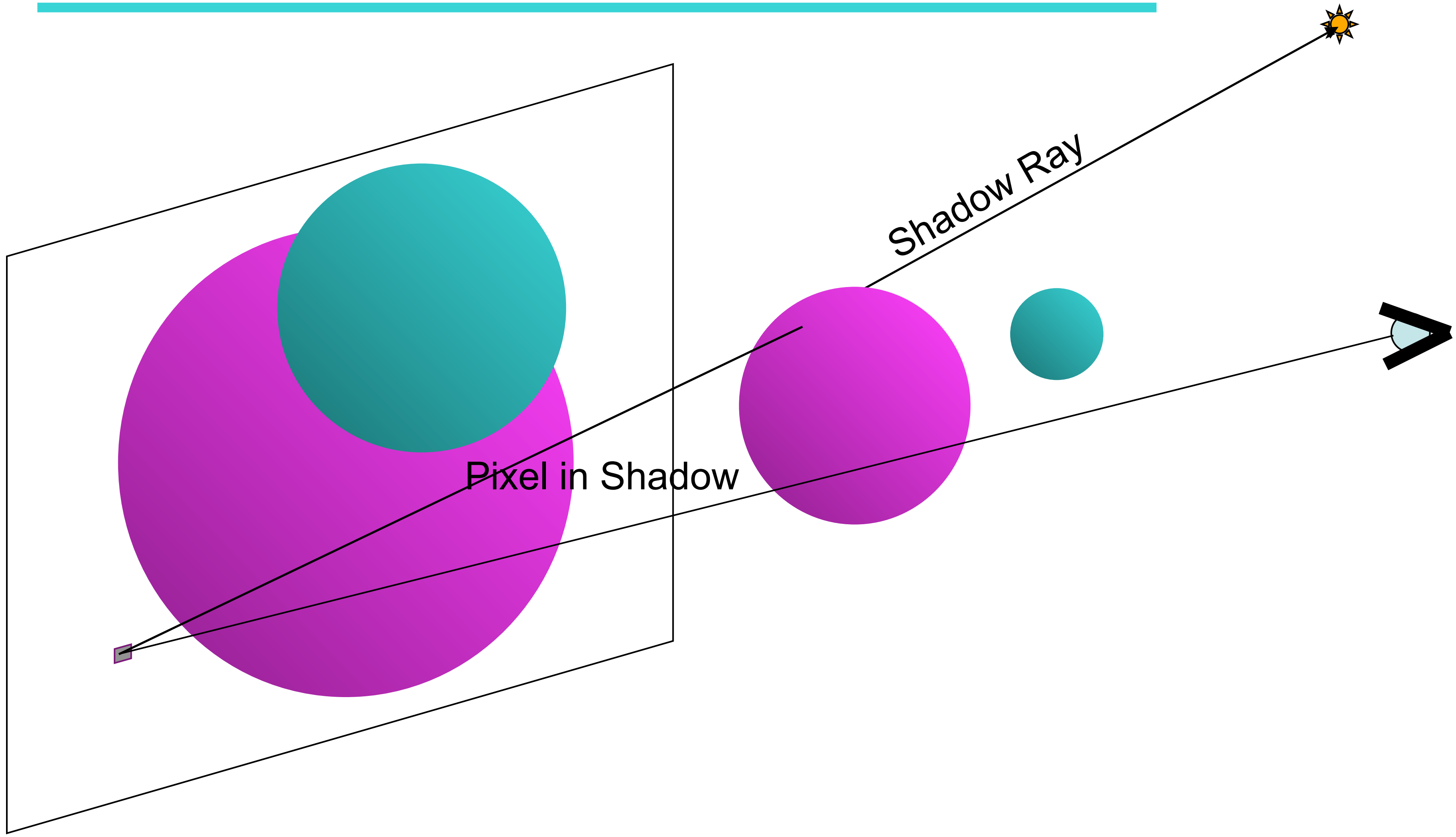
Starry Skies



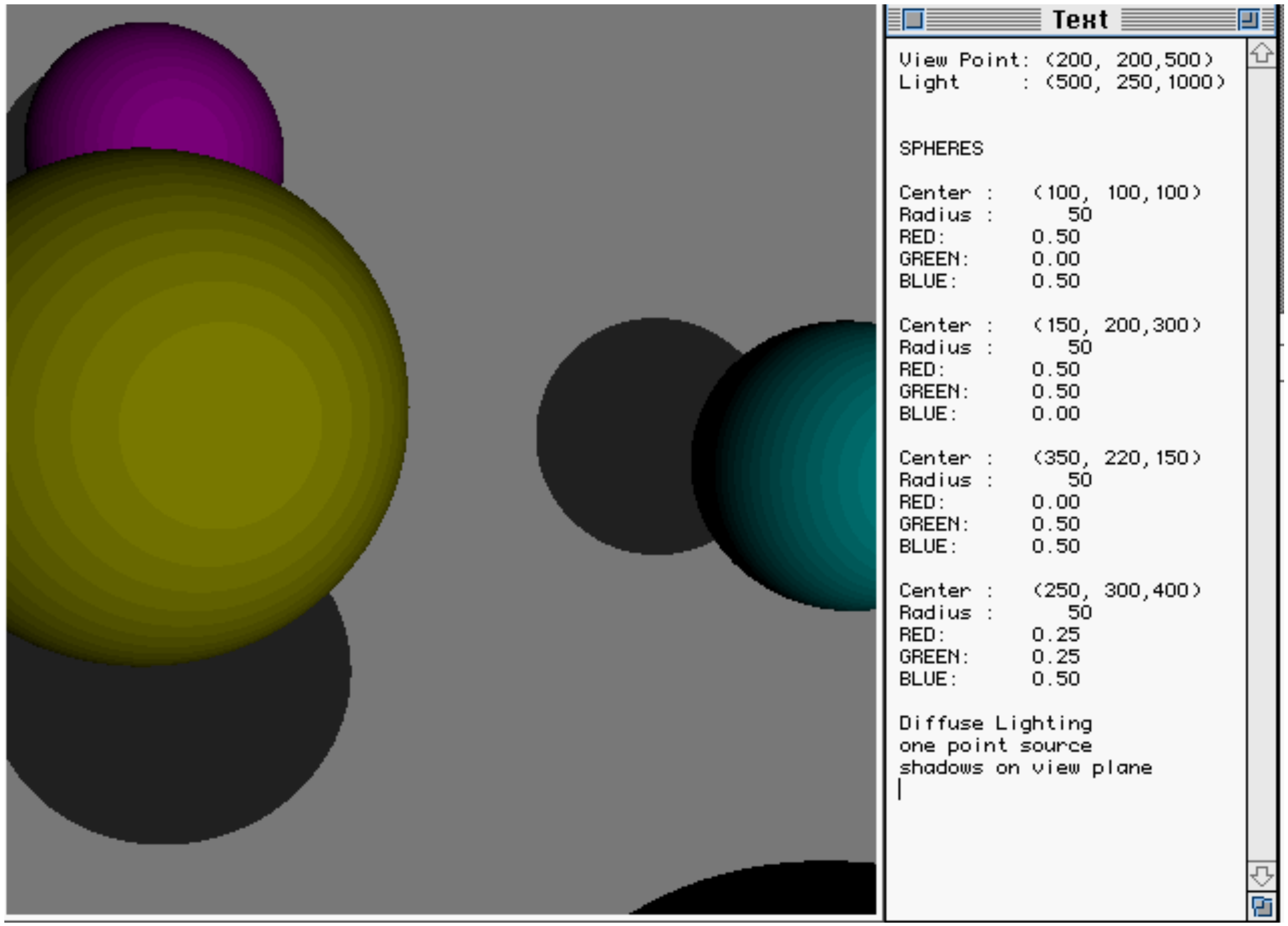
Shadows on the Plane



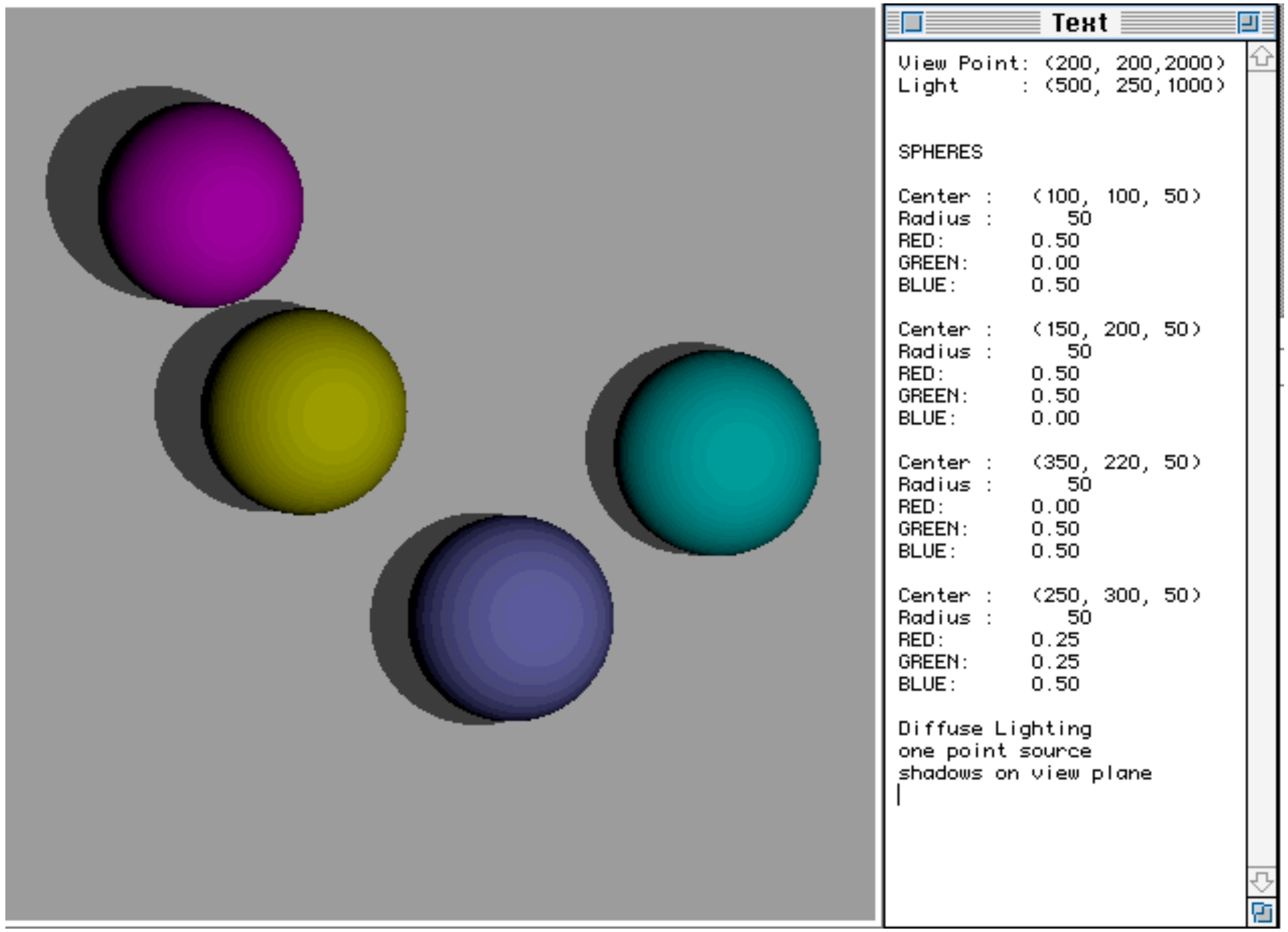
Finding Shadows on the Back Plane



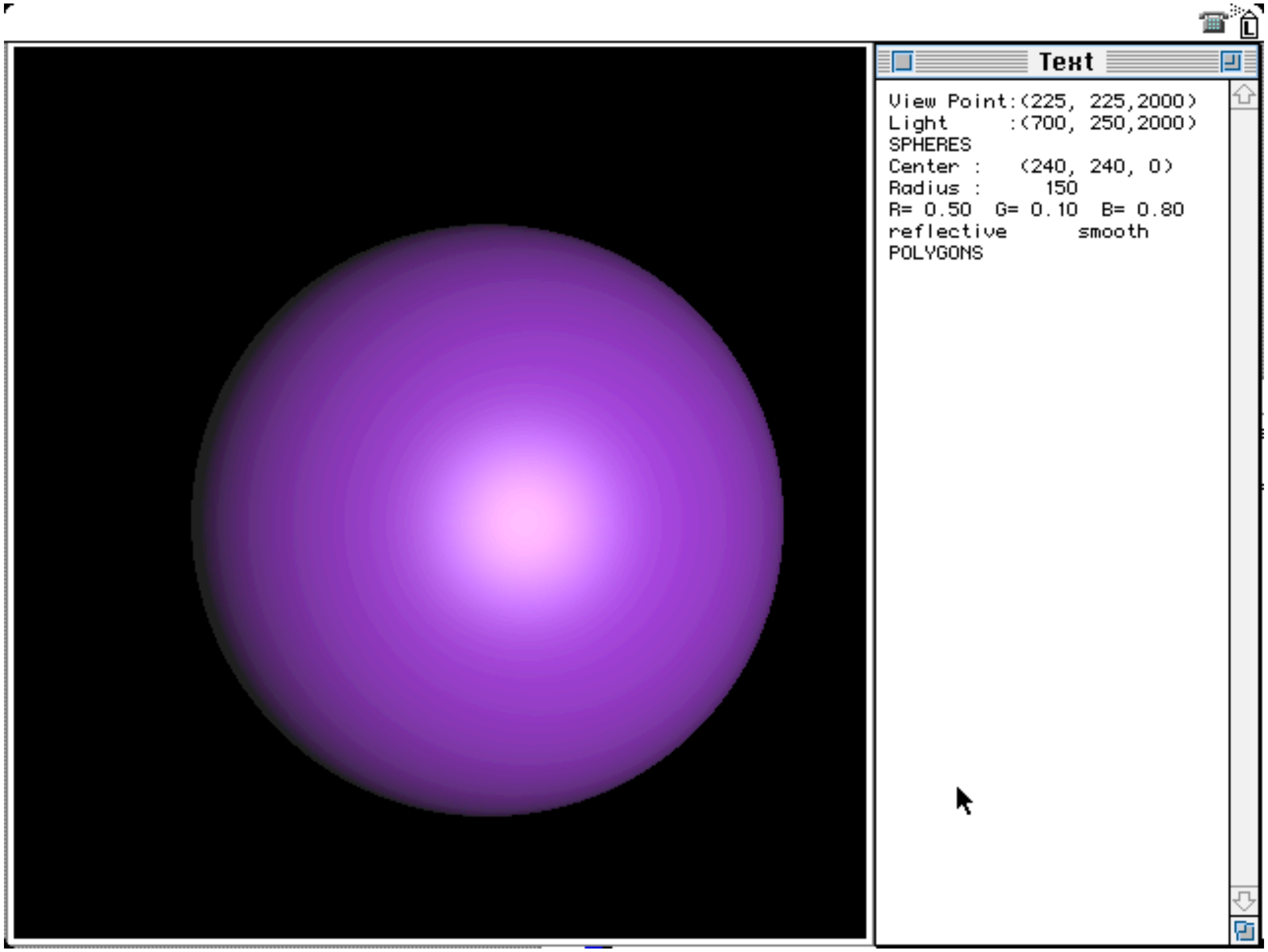
Close up

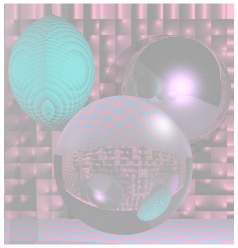

Text
View Point: (200, 200,500)
Light : (500, 250,1000)
SPHERES
Center : (100, 100,100)
Radius : 50
RED: 0.50
GREEN: 0.00
BLUE: 0.50
Center : (150, 200,300)
Radius : 50
RED: 0.50
GREEN: 0.50
BLUE: 0.00
Center : (350, 220,150)
Radius : 50
RED: 0.00
GREEN: 0.50
BLUE: 0.50
Center : (250, 300,400)
Radius : 50
RED: 0.25
GREEN: 0.25
BLUE: 0.50
Diffuse Lighting
one point source
shadows on view plane

On the Table


Text
View Point: (200, 200,2000)
Light : (500, 250,1000)
SPHERES
Center : (100, 100, 50)
Radius : 50
RED: 0.50
GREEN: 0.00
BLUE: 0.50
Center : (150, 200, 50)
Radius : 50
RED: 0.50
GREEN: 0.50
BLUE: 0.00
Center : (350, 220, 50)
Radius : 50
RED: 0.00
GREEN: 0.50
BLUE: 0.50
Center : (250, 300, 50)
Radius : 50
RED: 0.25
GREEN: 0.25
BLUE: 0.50
Diffuse Lighting
one point source
shadows on view plane

Phong Highlight





Phong Lighting Model

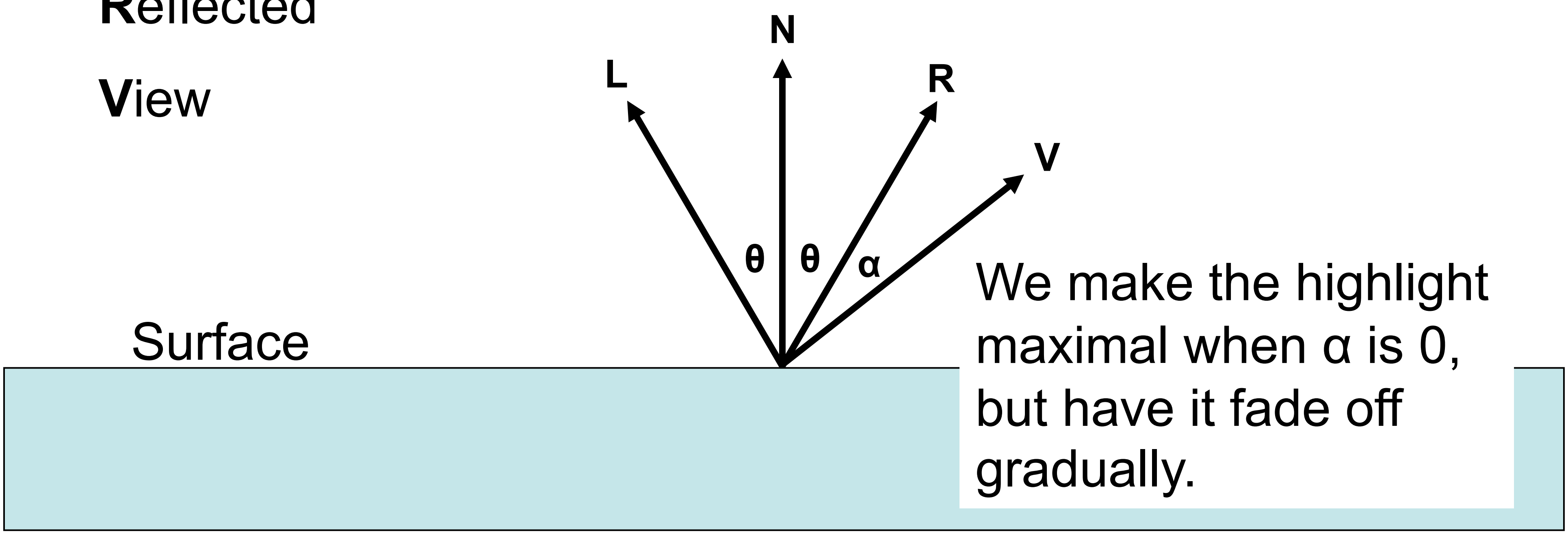
Light

Normal

Reflected

View

The viewer only sees the light when α is 0.



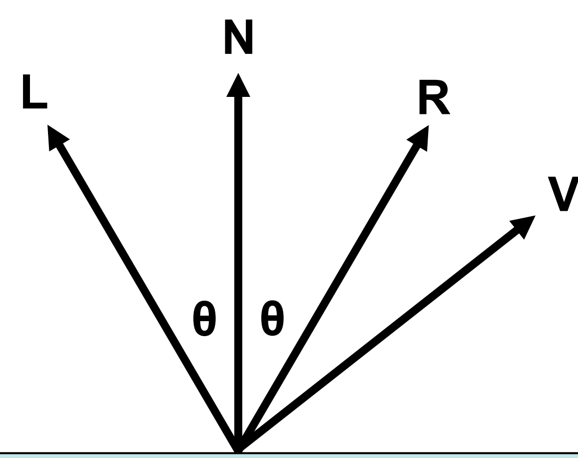
We make the highlight maximal when α is 0, but have it fade off gradually.

Phong Lighting Model

$\text{Cos}(\alpha) = \mathbf{R} \bullet \mathbf{V}$

We use $\cos^n(\alpha)$.

The higher n is, the faster the drop off.

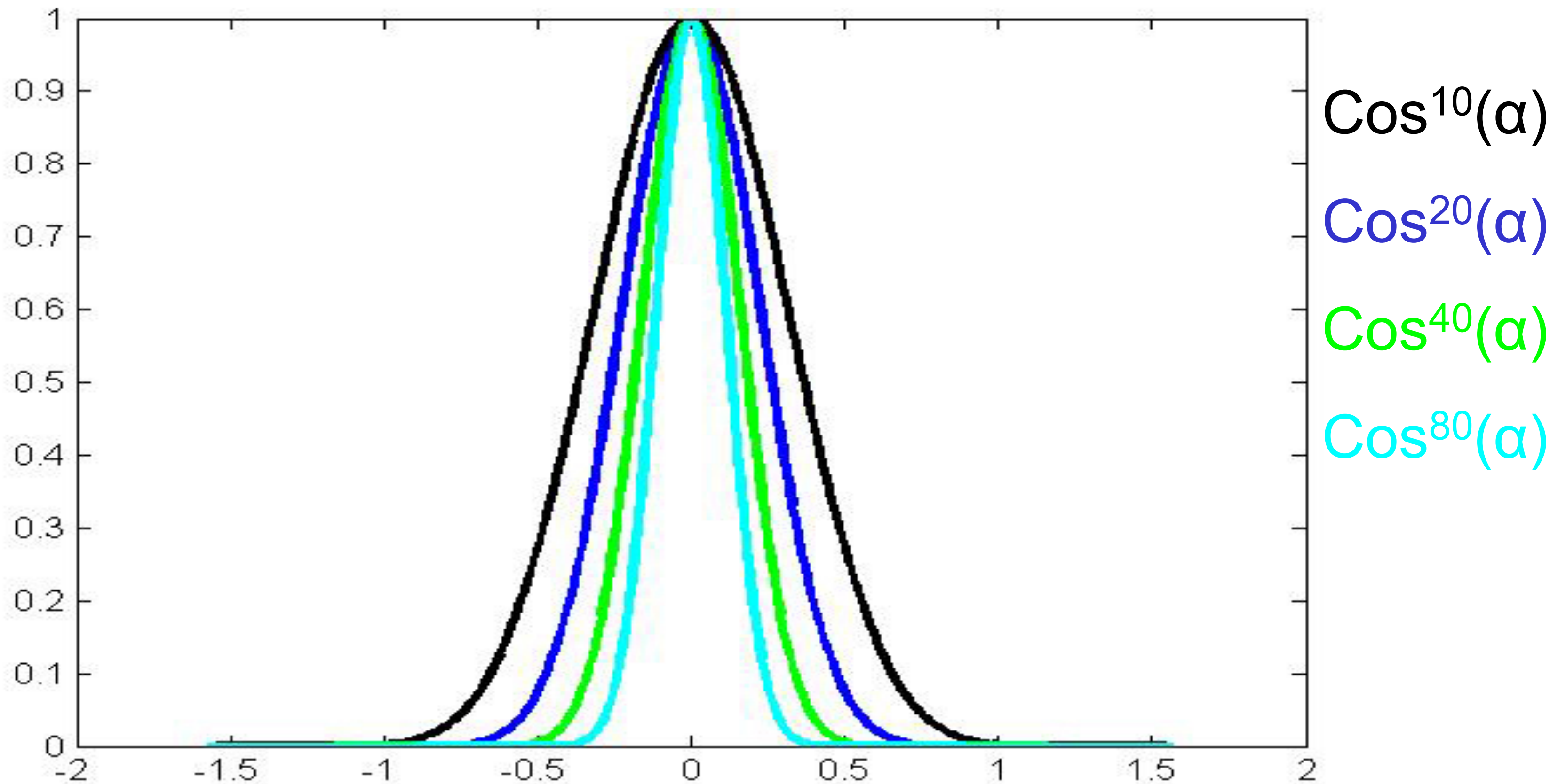


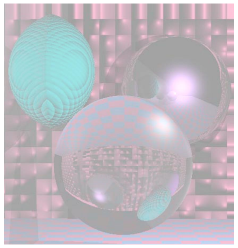
For white light

Surface

$$Cp = ka\,(SR, SG, SB) + kd\,\mathbf{N}\bullet\mathbf{L}\,(SR, SG, SB) + ks\,(\mathbf{R}\bullet\mathbf{V})^n(1, 1, 1)$$

Powers of cos(α)

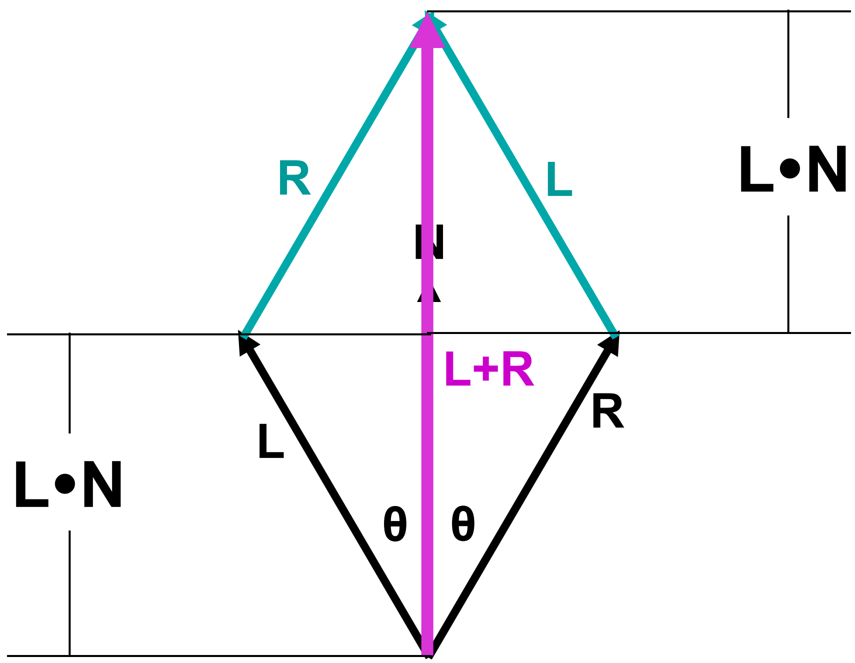


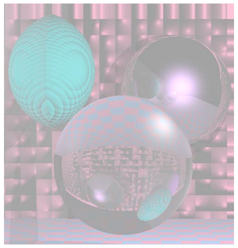


Computing **R**

L + **R** = (2 **L•N**) **N**

R = (2 **L•N**) **N** - **L**





The Halfway Vector

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$$

Use **H•N**

instead of **R•V**.

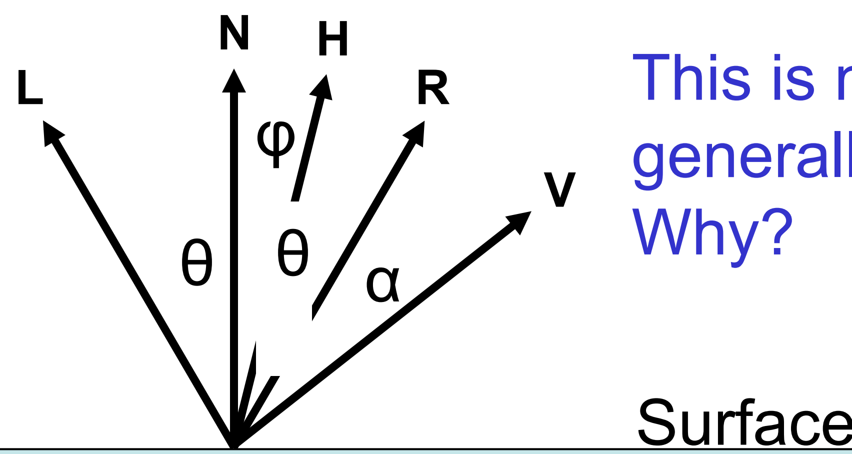
H is less expensive to compute than **R**.

From the picture

$\theta + \varphi = \theta - \varphi + \alpha$

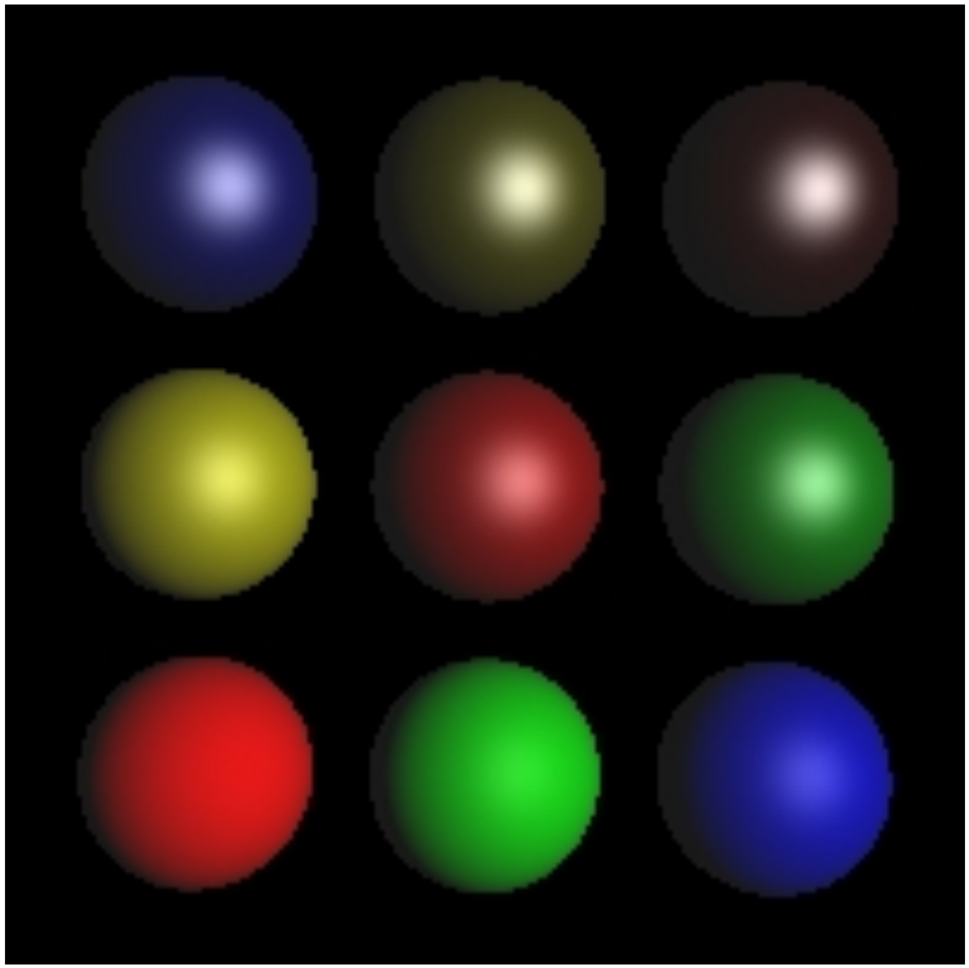
So $\varphi = \alpha/2$.

This is not generally true. Why?

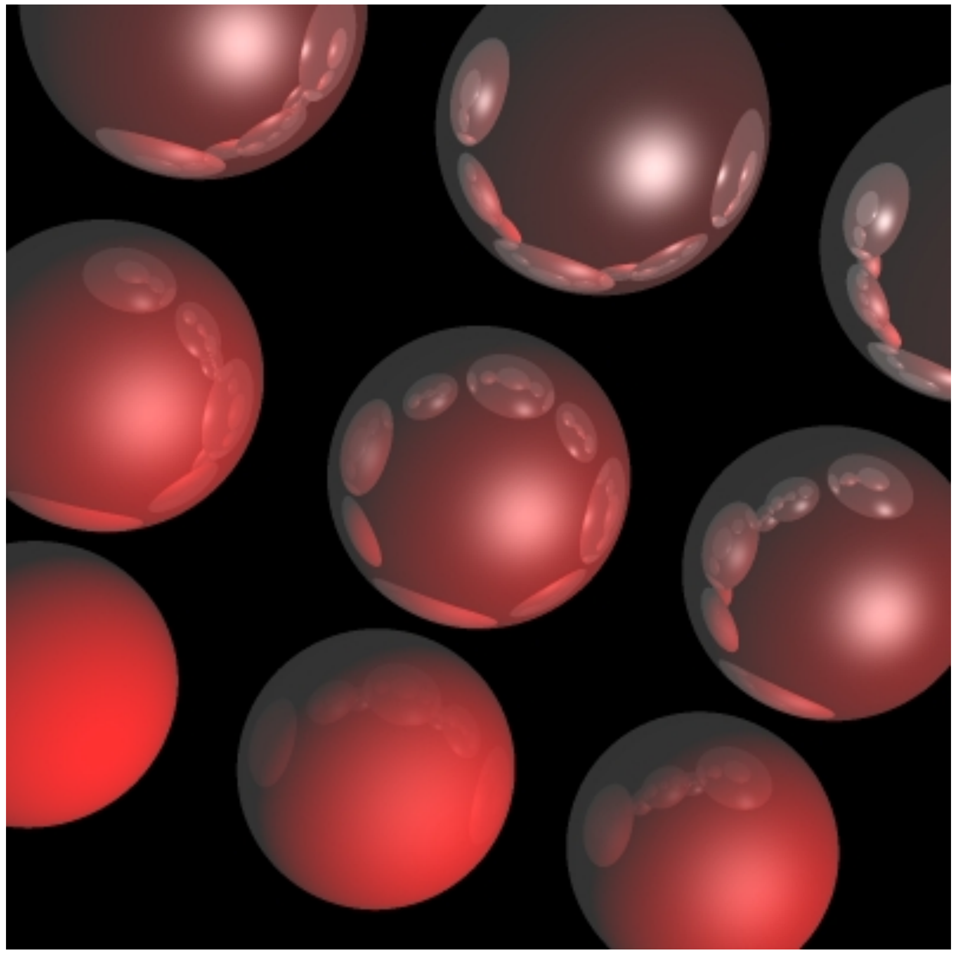


Cp= ka (SR, SG, SB) + kd **N•L** (SR, SG, SB) + ks (**H•N**)^n (1, 1, 1)

Varied Phong Highlights



Varying Reflectivity



Time for a Break

More Math

- Matrices
- Transformations
- Homogeneous Coordinates

Matrices

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \qquad C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

- We use 2x2, 3x3, and 4x4 matrices in computer graphics.
- We’ll start with a review of 2D matrices and transformations.

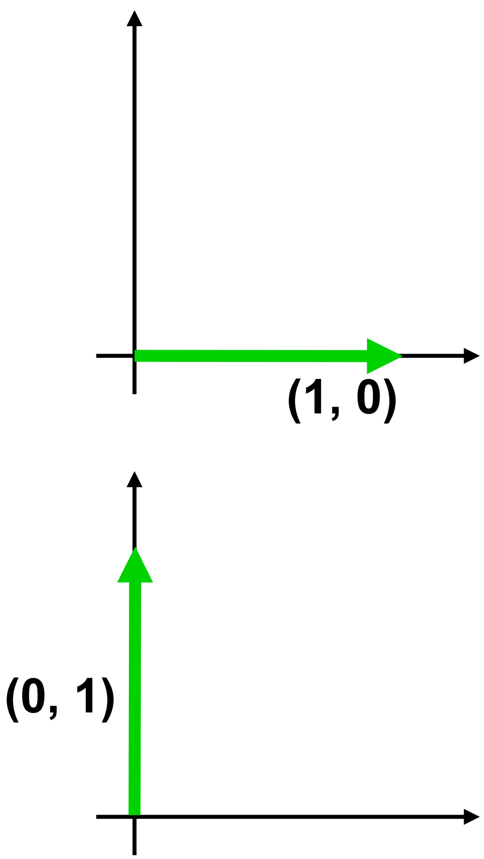
Basic 2D Linear Transforms

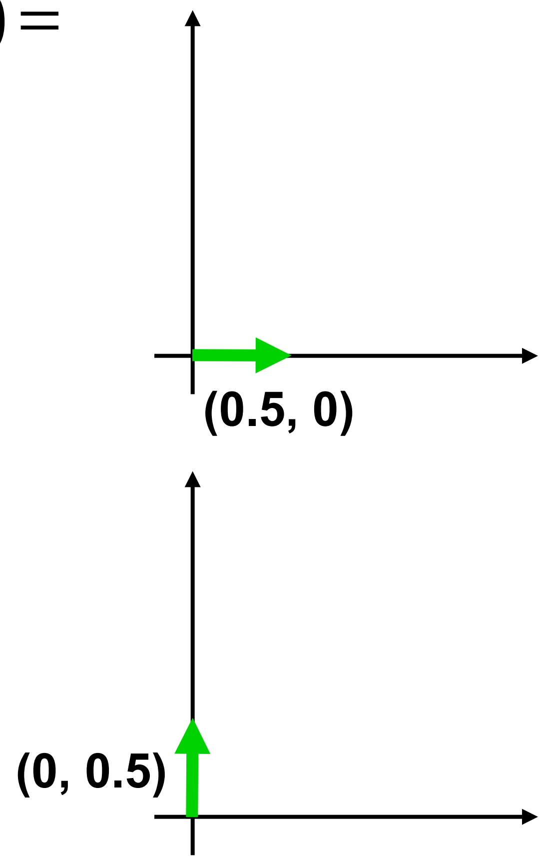
$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} \qquad \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}$$

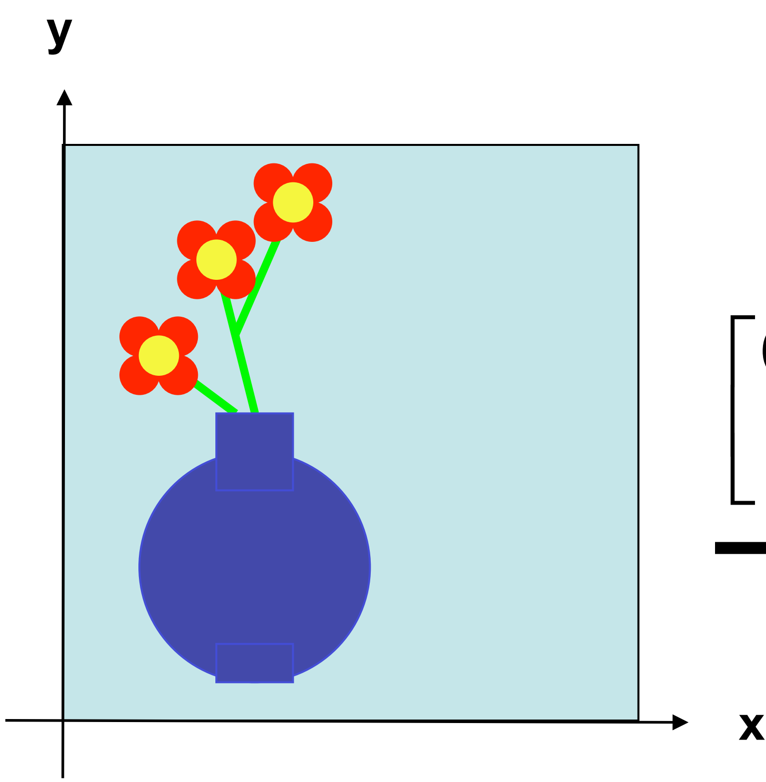
Scale by .5

$$\mathrm{scale}(.5,.5) = \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}$$

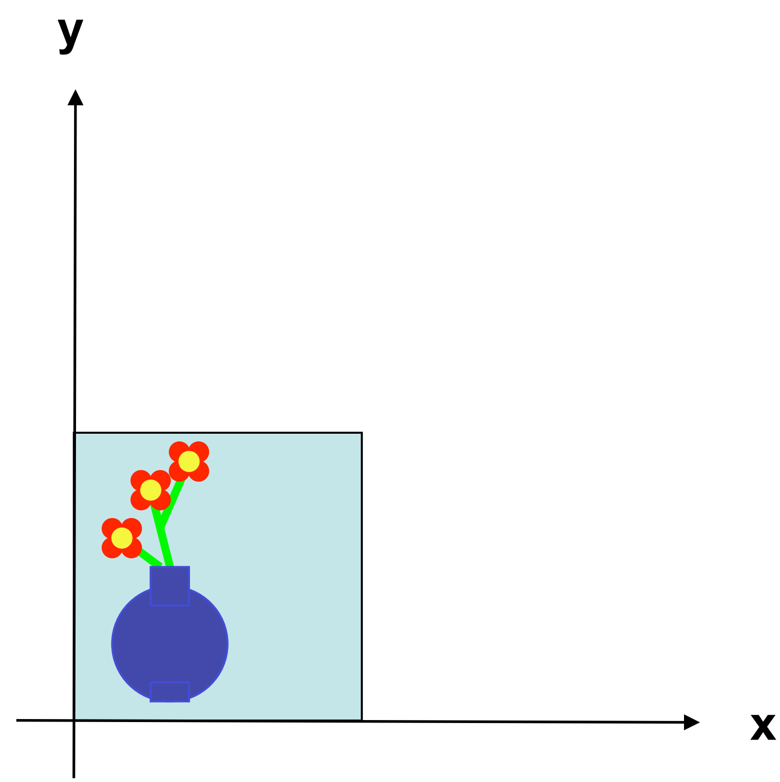




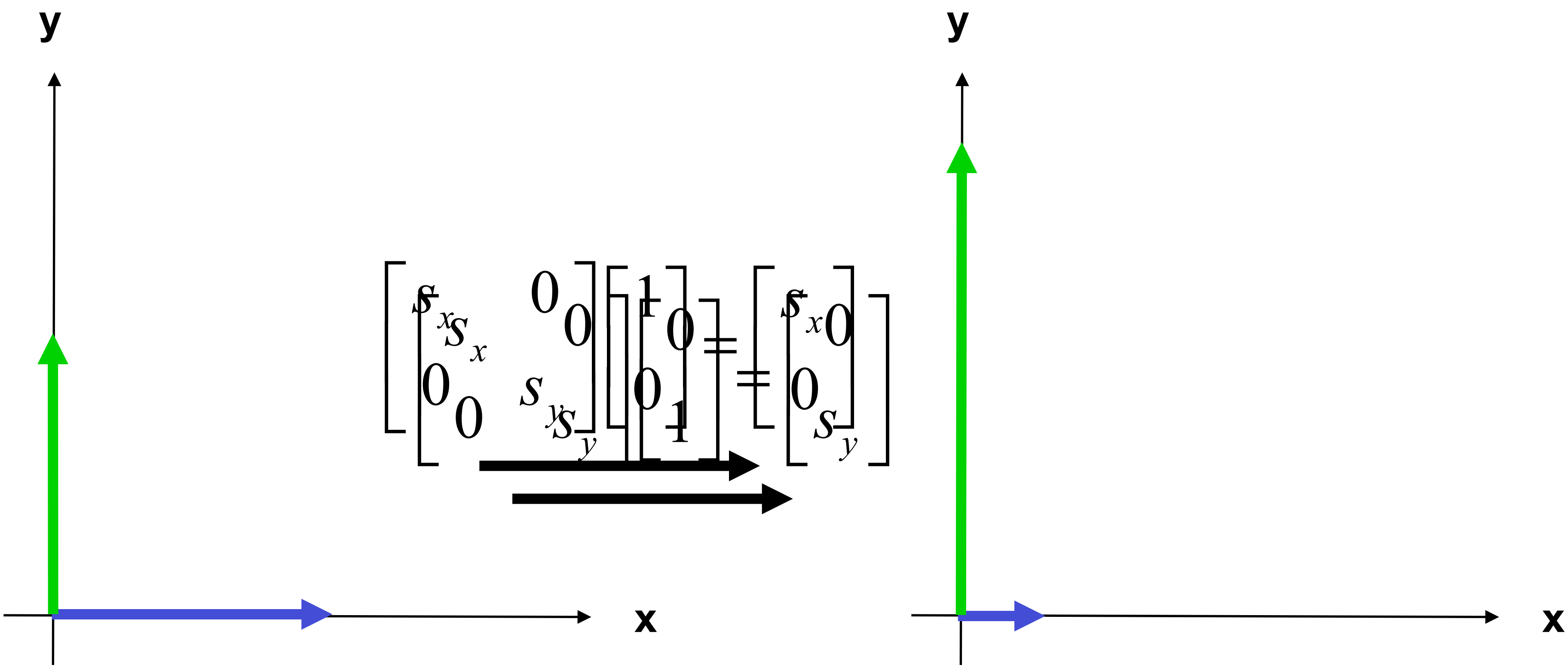
Scaling by .5



$$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$



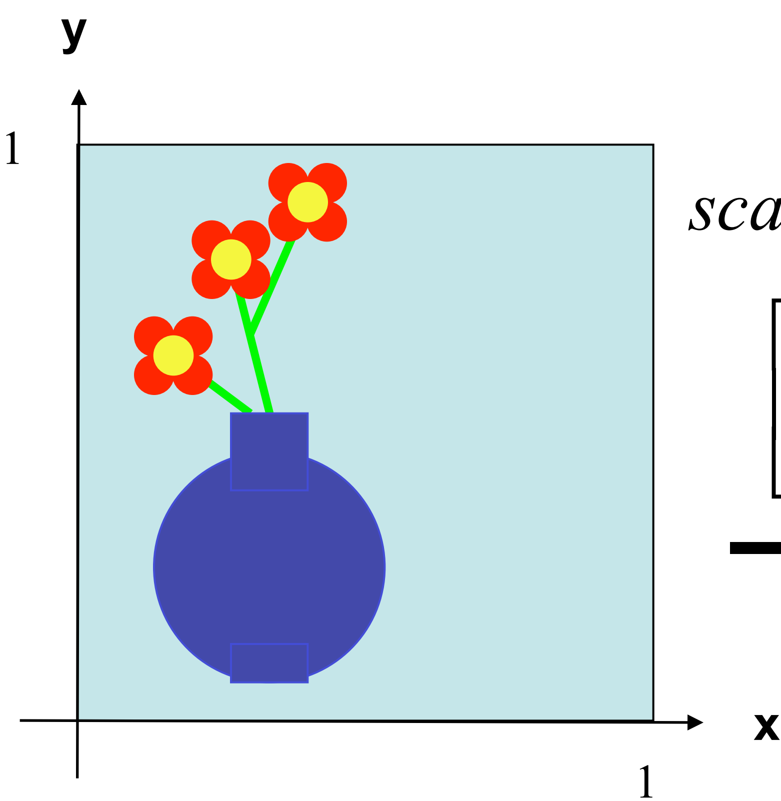
General Scaling



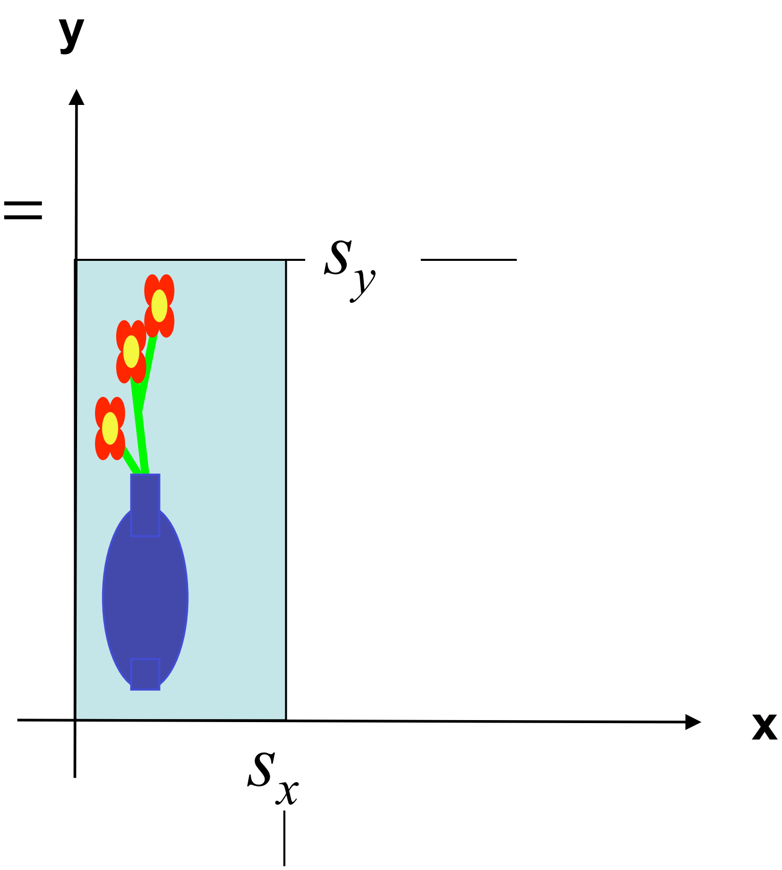
$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} s_x \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ s_y \end{bmatrix}$$

General Scaling

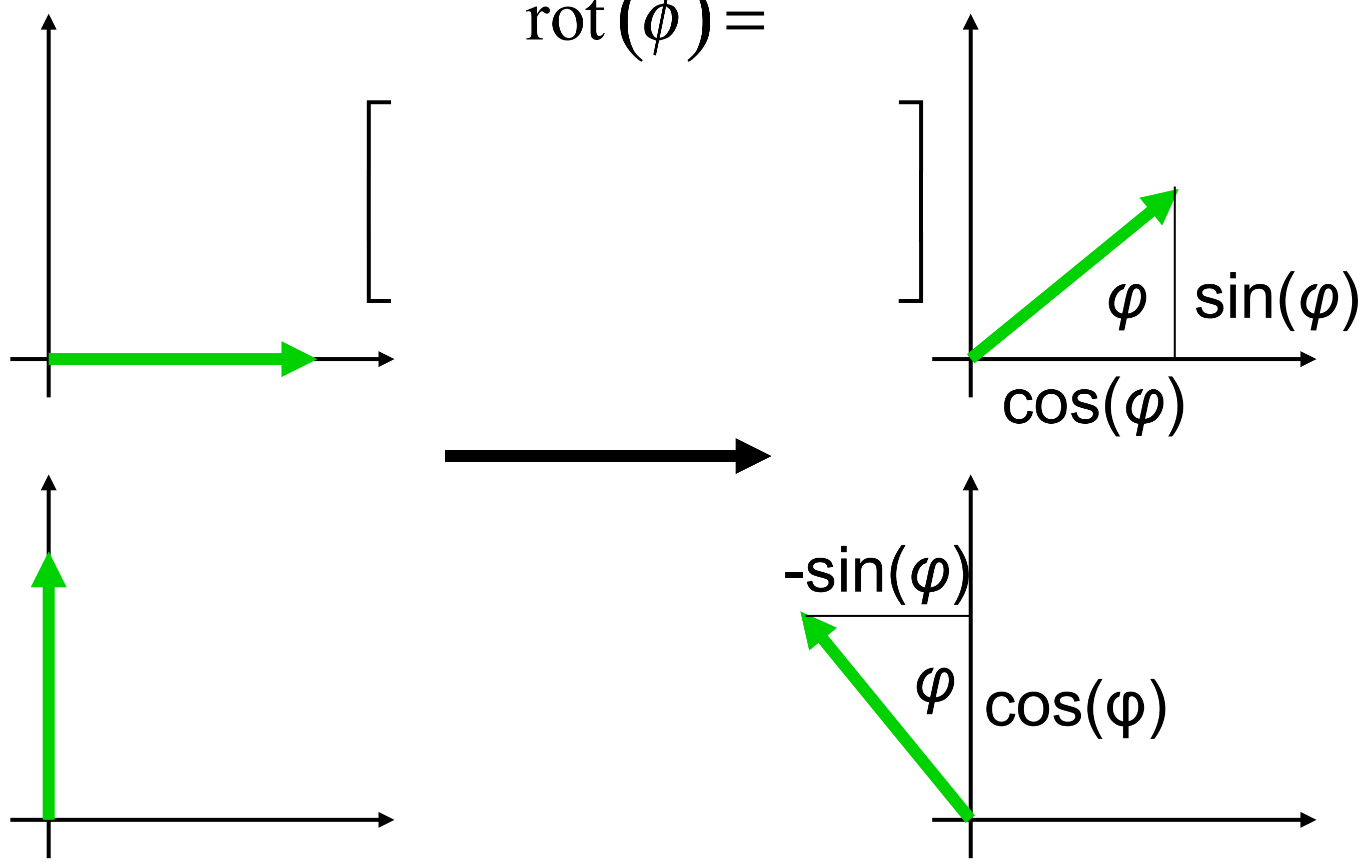


$$scale(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



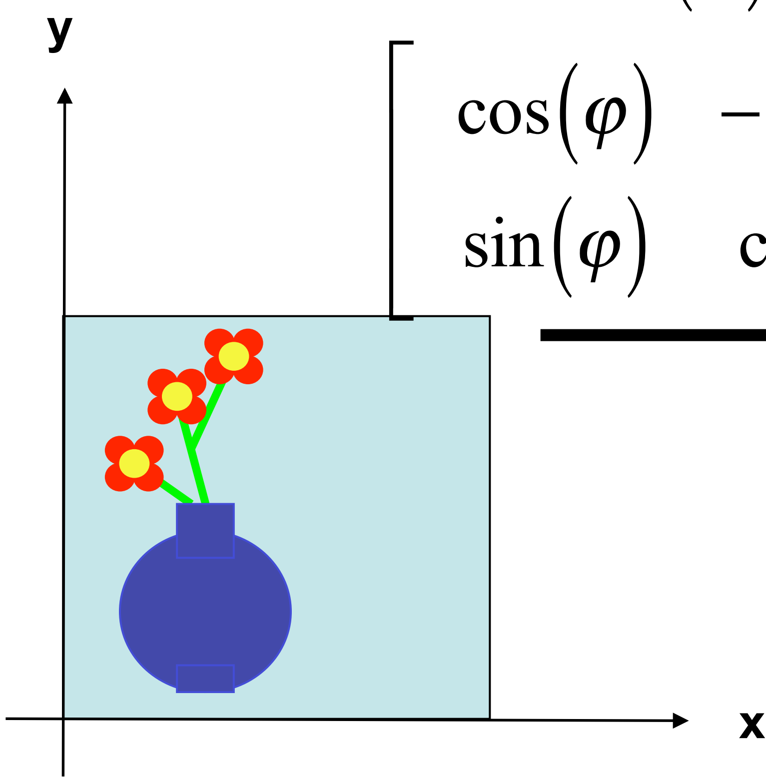
Rotation

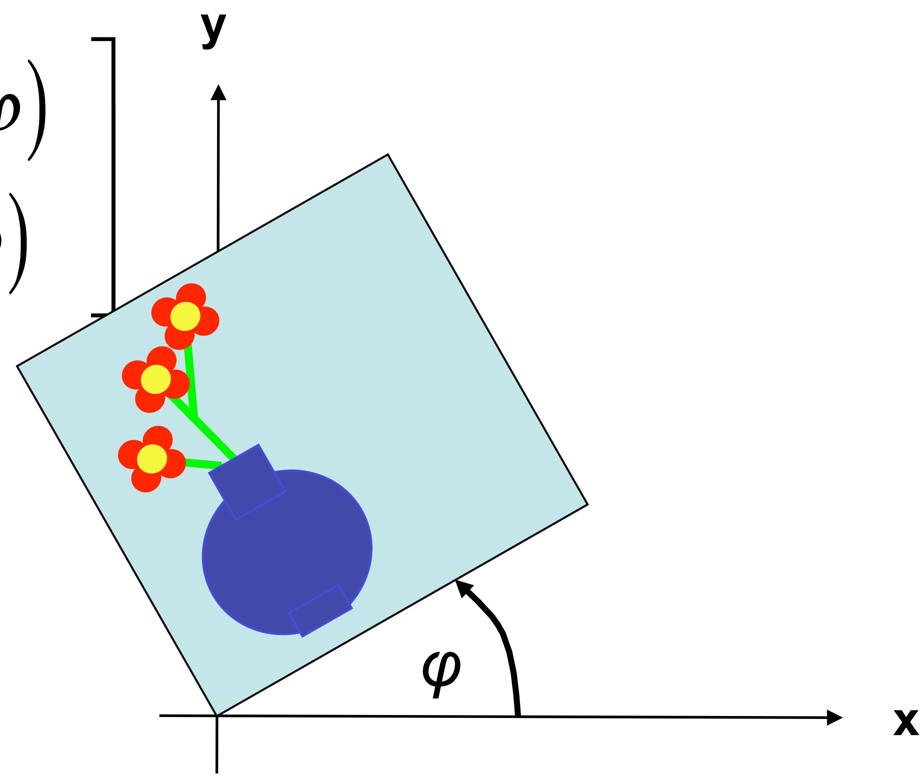
$$\mathrm{rot}(\phi) =$$



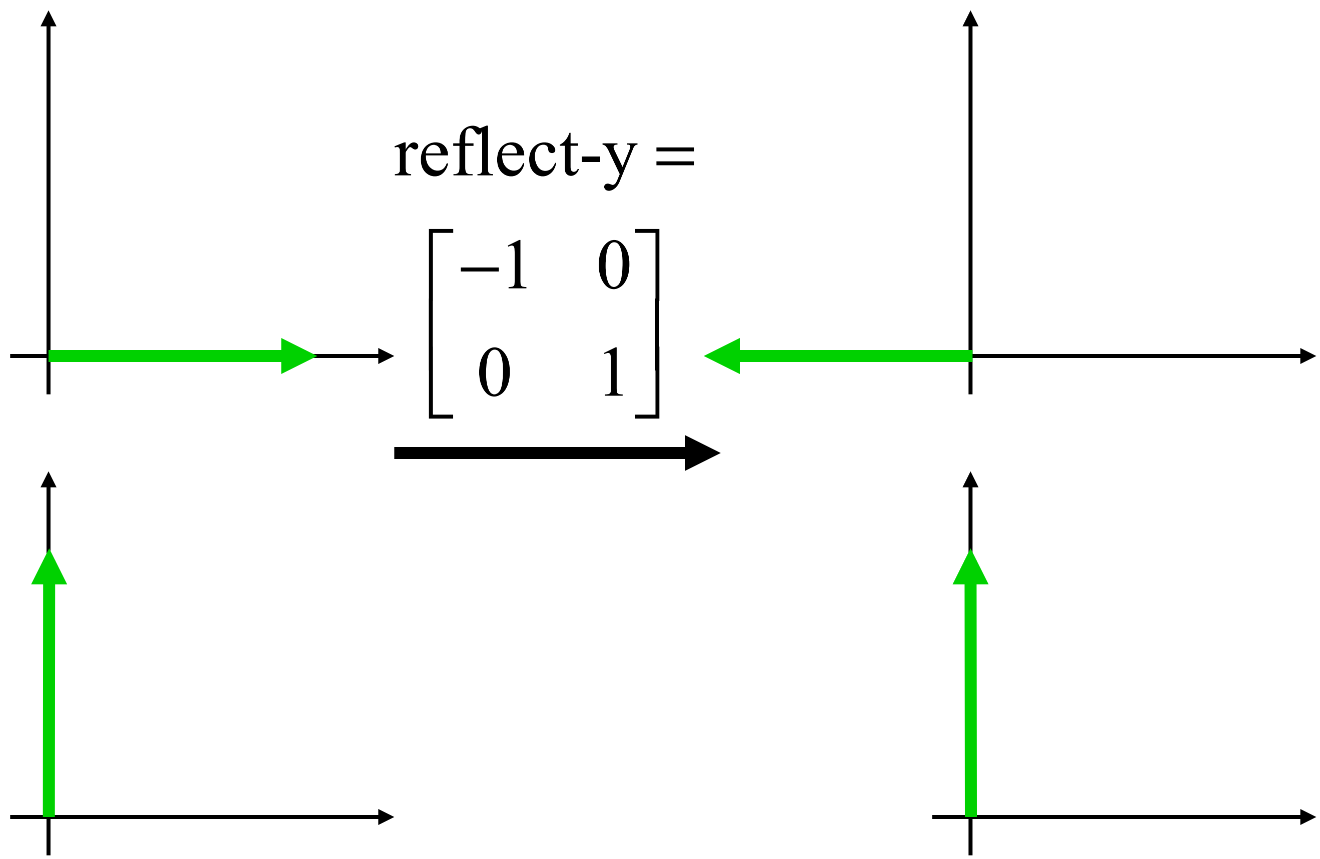
Rotation

$$\mathrm{rot}(\varphi) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}$$



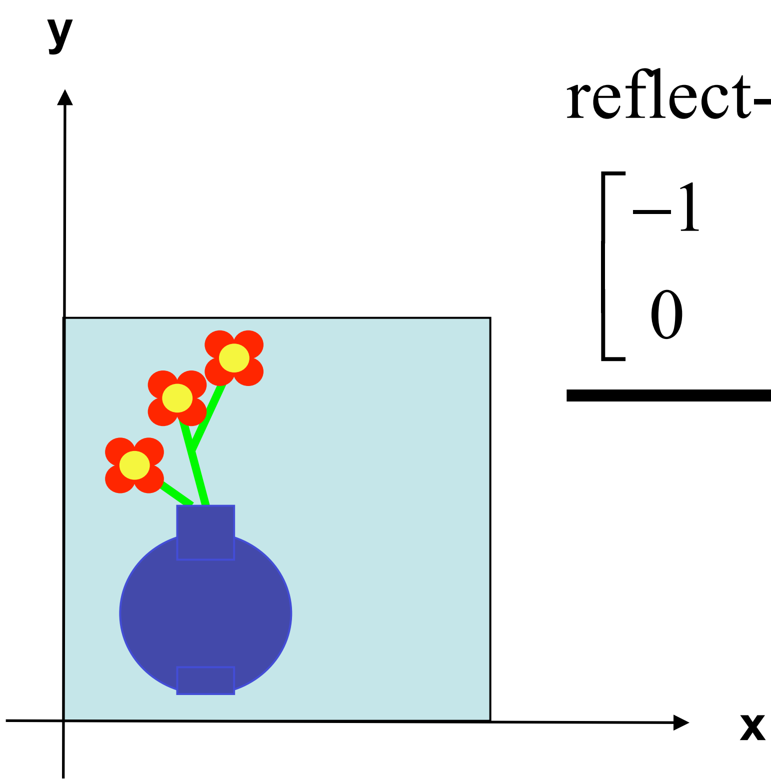


Reflection in y-axis

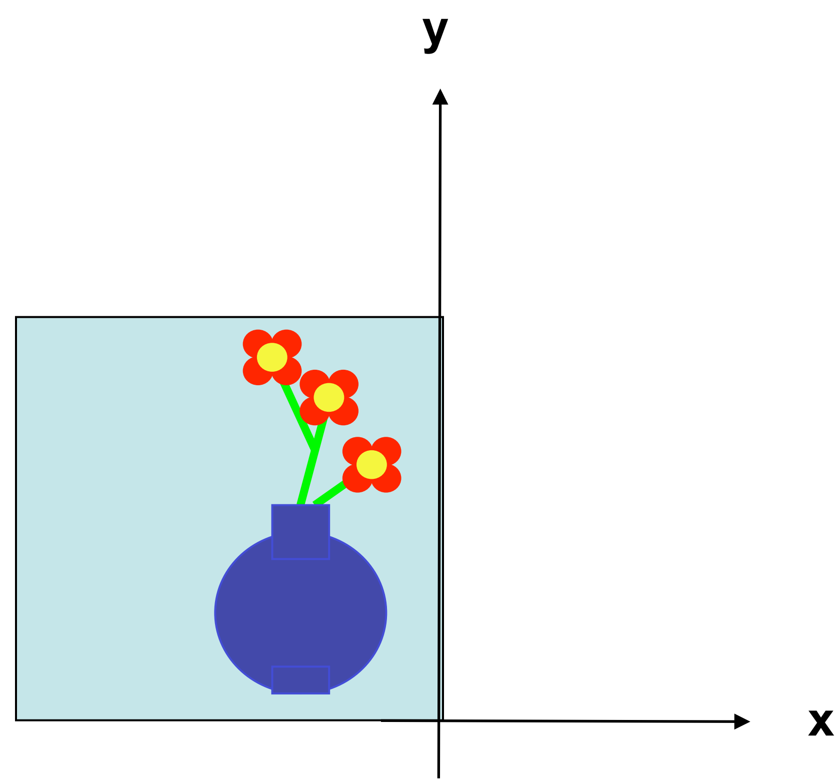


$$\text{reflect-y} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

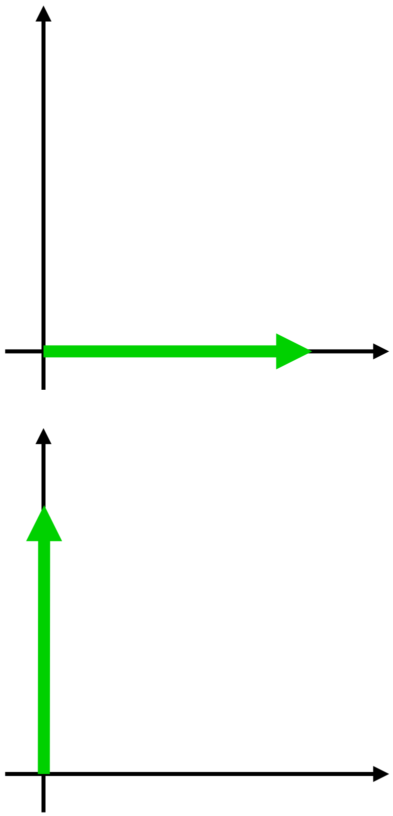
Reflection in y-axis



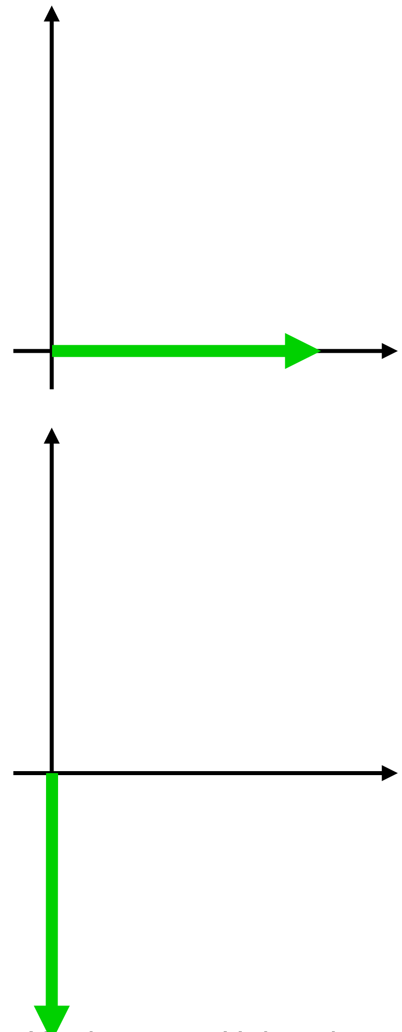
$$\text{reflect-y} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



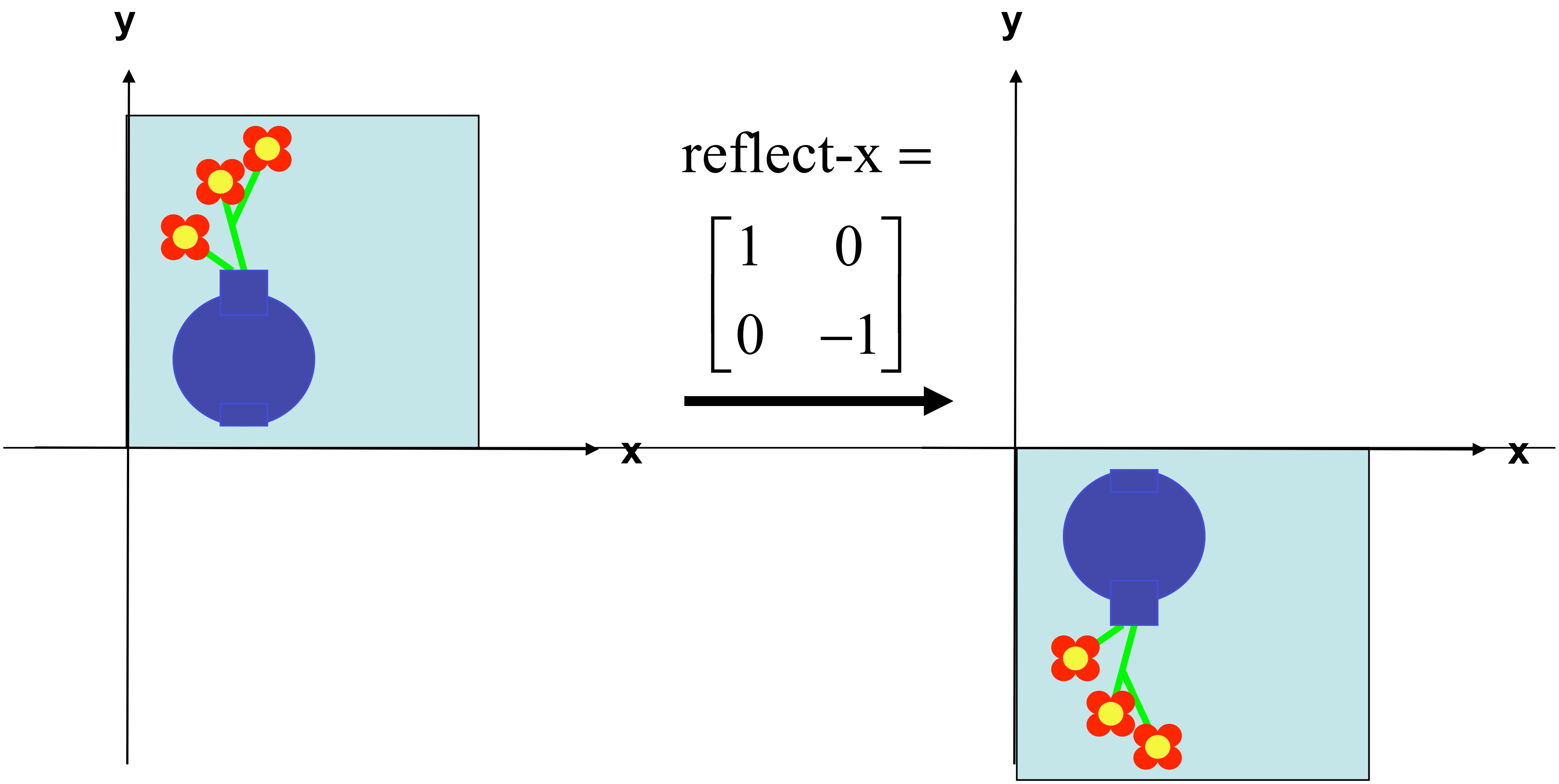
Reflection in x-axis



$$\text{reflect-x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$



Reflection in x-axis

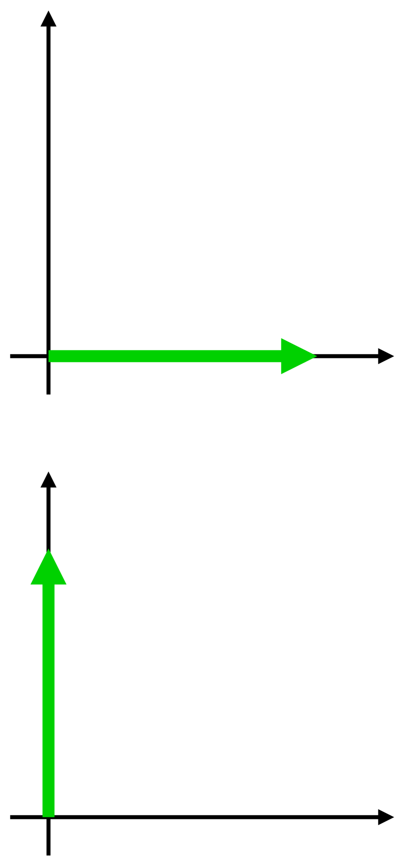


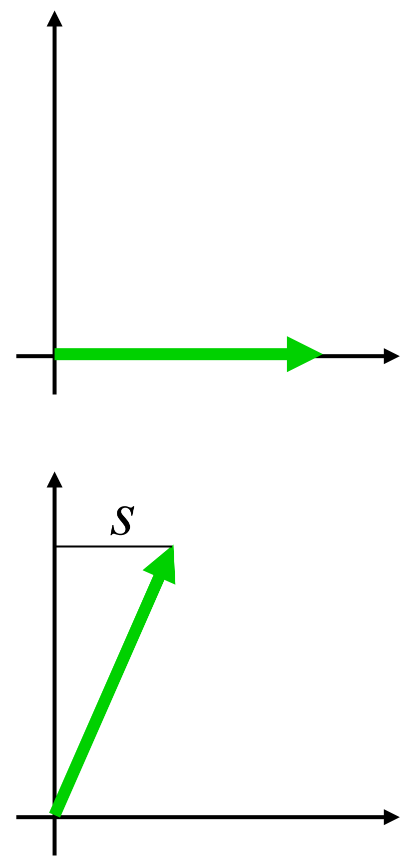
reflect-x =

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

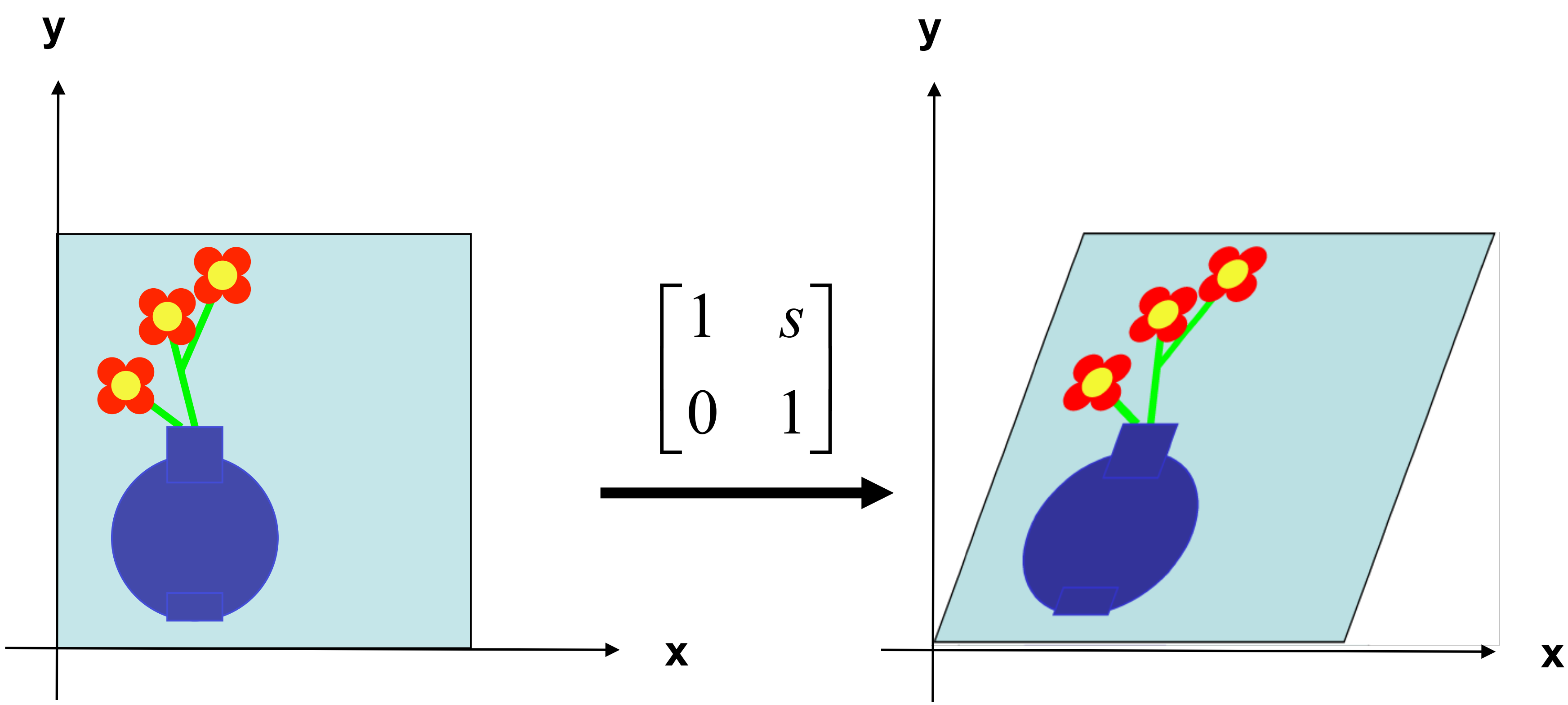
Shear-x

$$\text{shear-x}(s) = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$





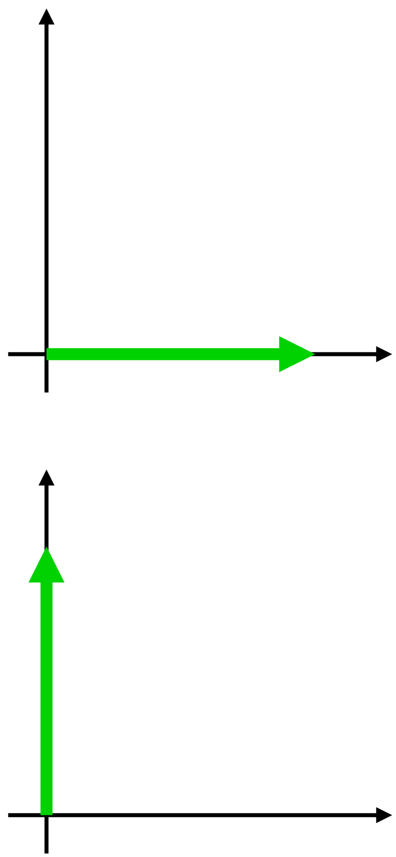
Shear x

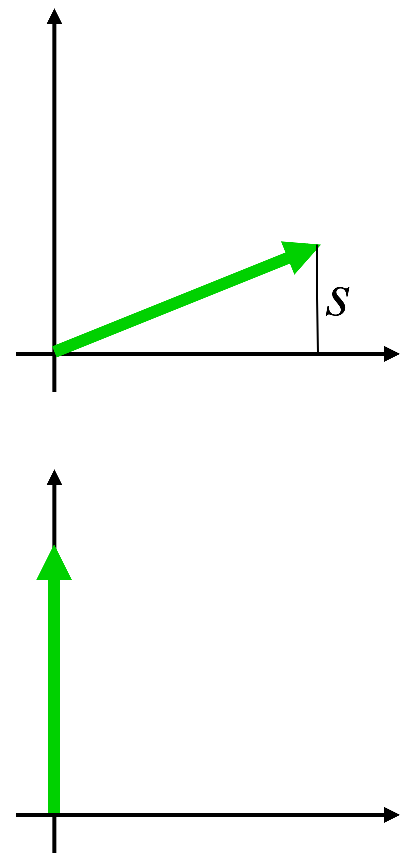


$$\begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

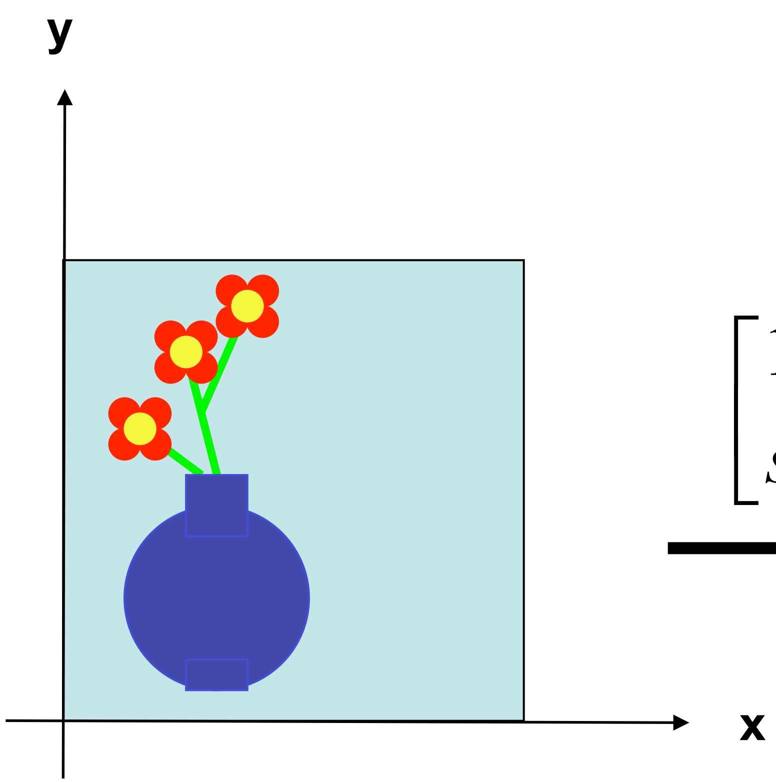
Shear-y

$$\text{shear-y}(s) = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

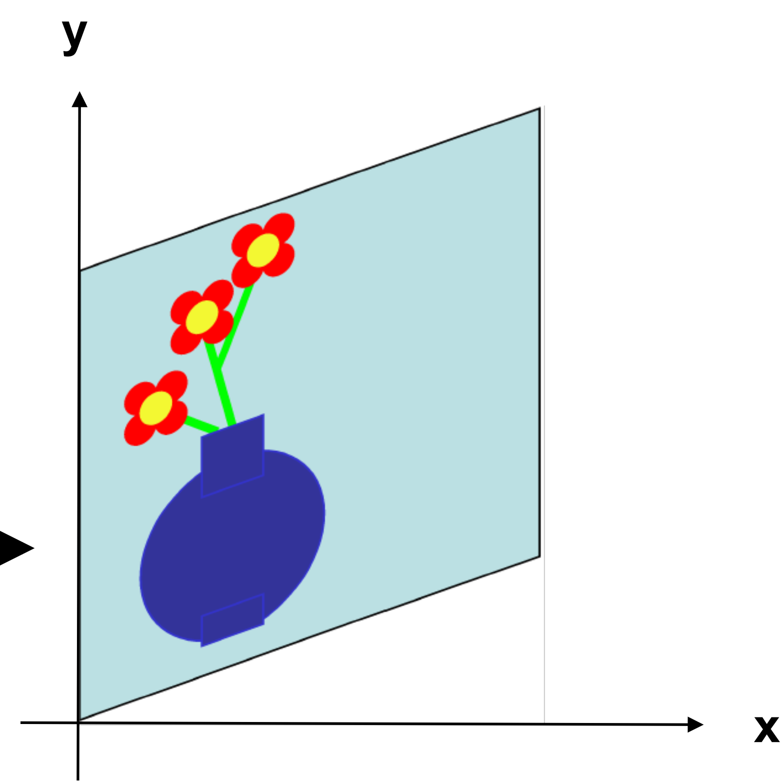




Shear y



$$\begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$



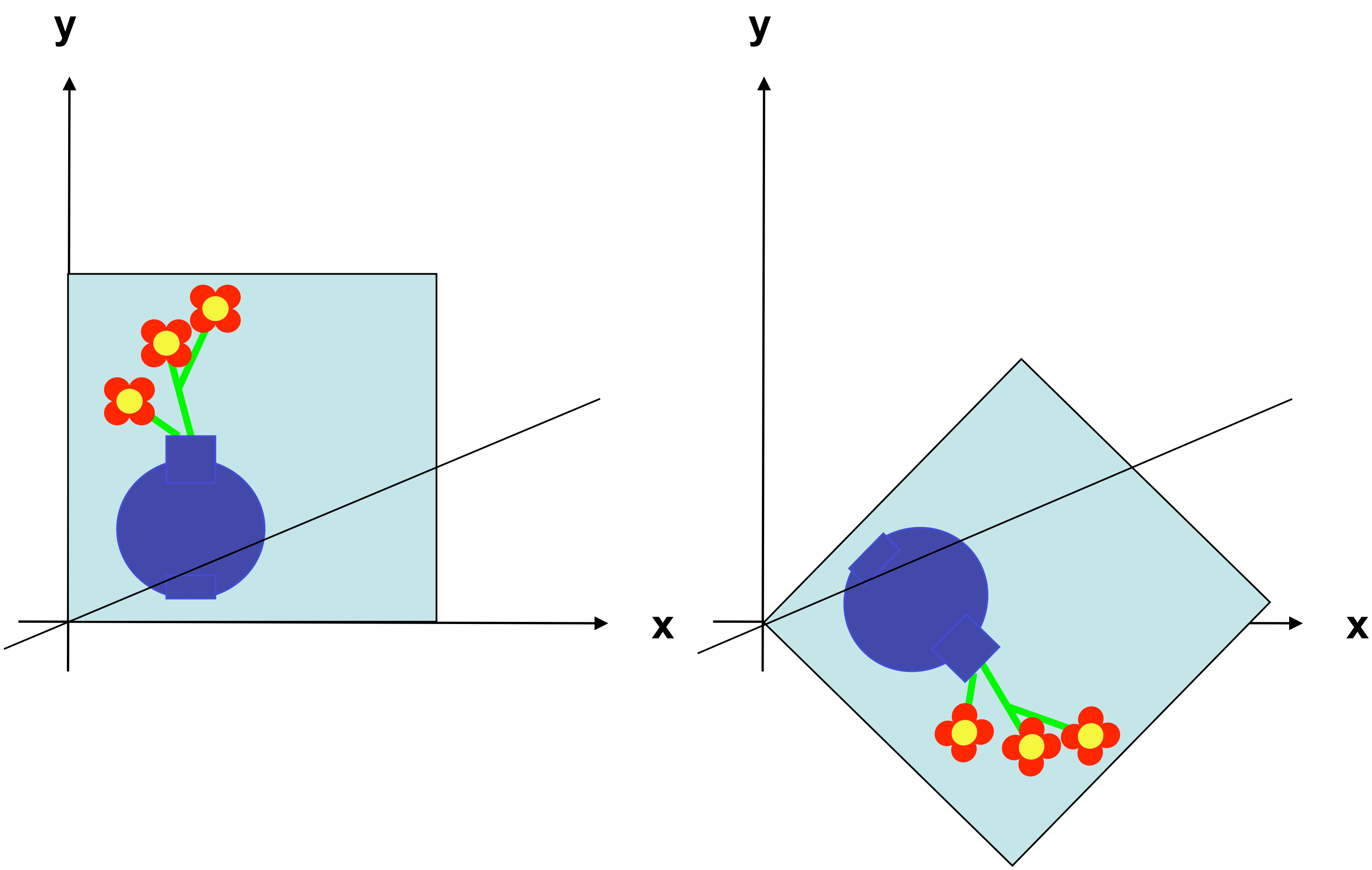
Linear Transformations

- Scale, Reflection, Rotation, and Shear are all linear transformations
- They satisfy: $T(a\mathbf{u} + b\mathbf{v}) = aT(\mathbf{u}) + bT(\mathbf{v})$
  - **u** and **v** are vectors
  - *a* and *b* are scalars
- If T is a linear transformation
  - $T((0, 0)) = (0, 0)$

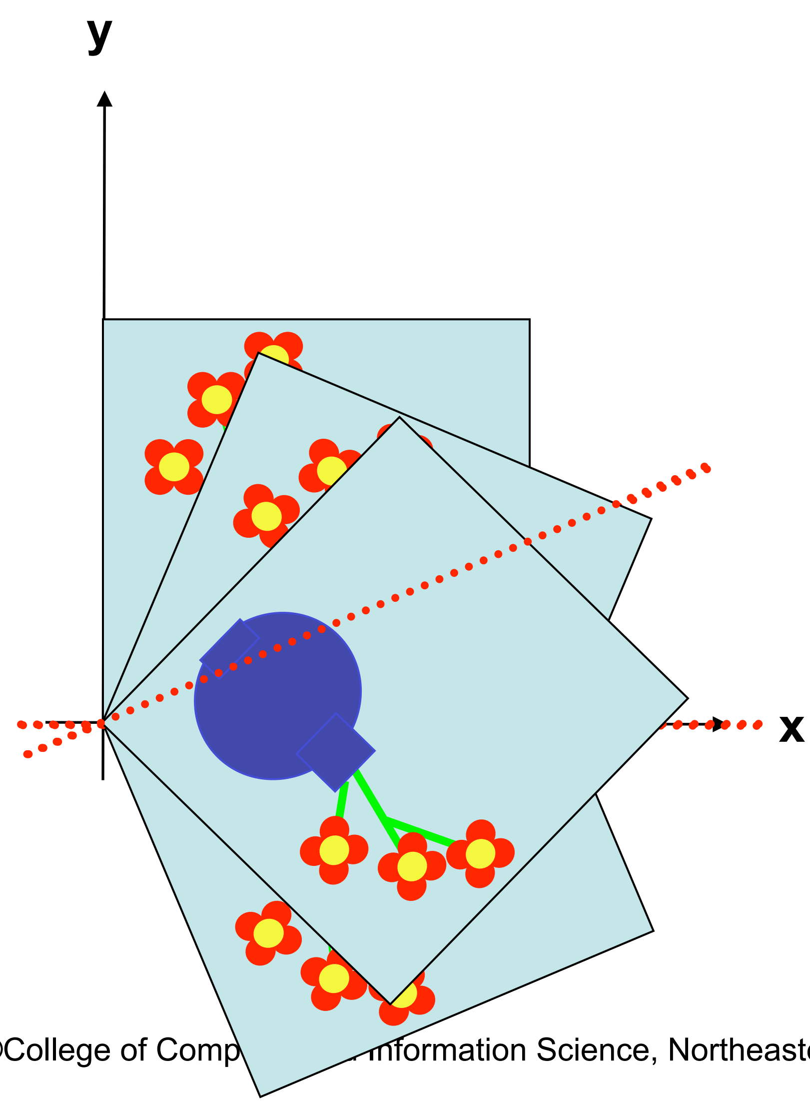
Composing Linear Transformations

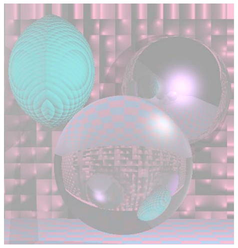
- If T_1 and T_2 are transformations
  - $T_2\,T_1(\mathbf{v}) =_{def} T_2(\,T_1(\mathbf{v}))$
- If T_1 and T_2 are linear and are represented by matrices M_1 and M_2
  - $T_2\,T_1$ is represented by $M_2\,M_1$
  - $T_2\,T_1(\mathbf{v}) = T_2(\,T_1(\mathbf{v})) = (M_2\,M_1)(\mathbf{v})$

Reflection About an Arbitrary Line (through the origin)



Reflection as a Composition

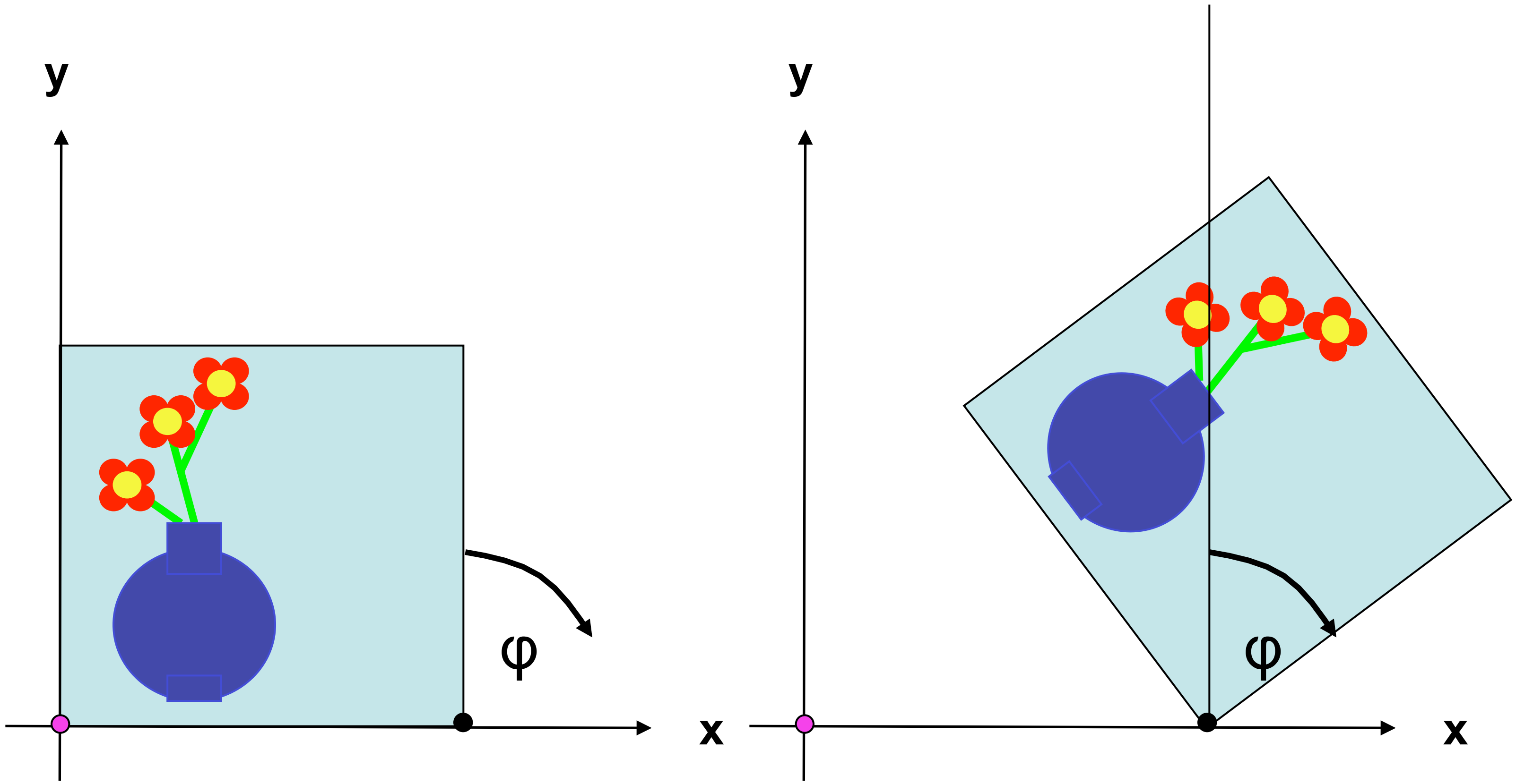




Decomposing Linear Transformations

- Any 2D Linear Transformation can be decomposed into the product of a rotation, a scale, and a rotation if the scale can have negative numbers.
- $M = R_1 S R_2$

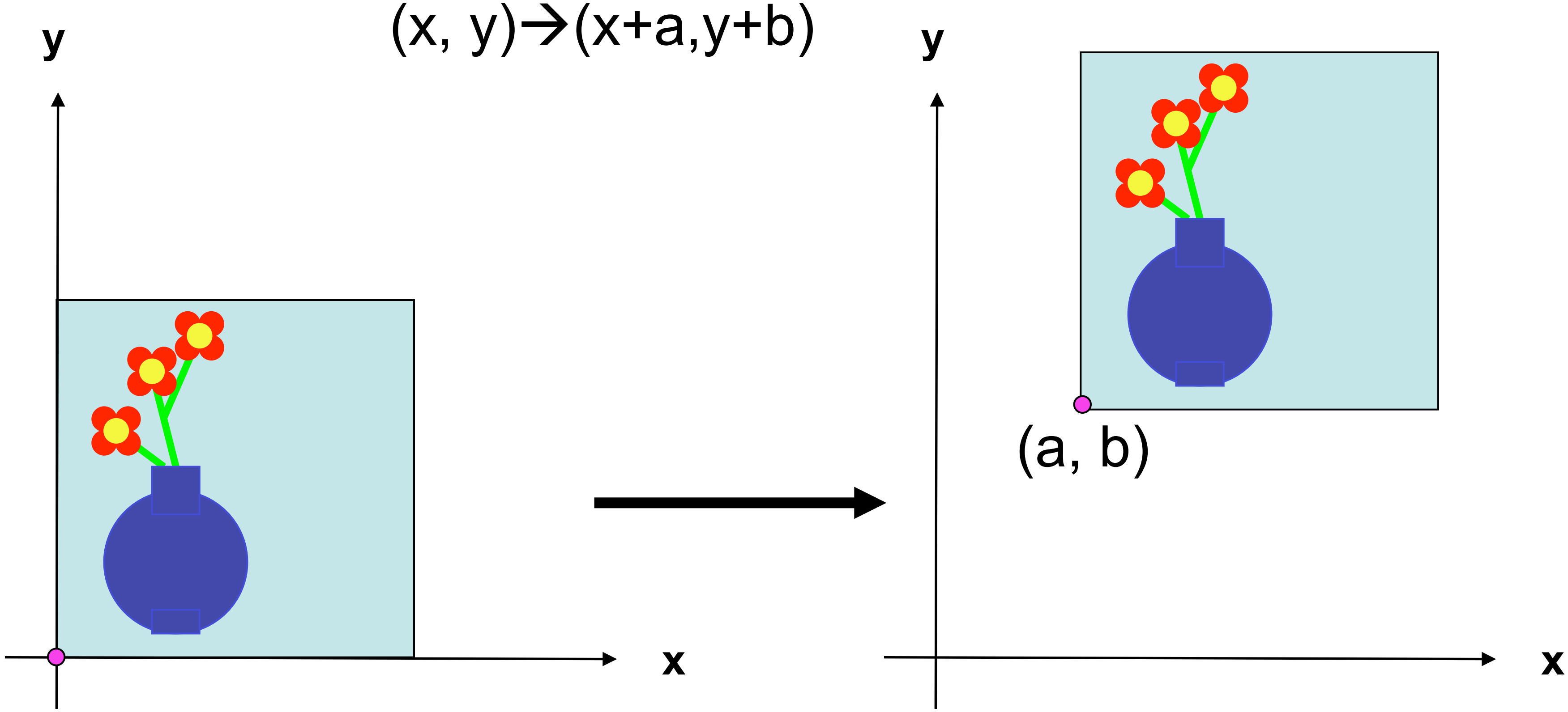
Rotation about an Arbitrary Point



This is not a linear transformation. The origin moves.

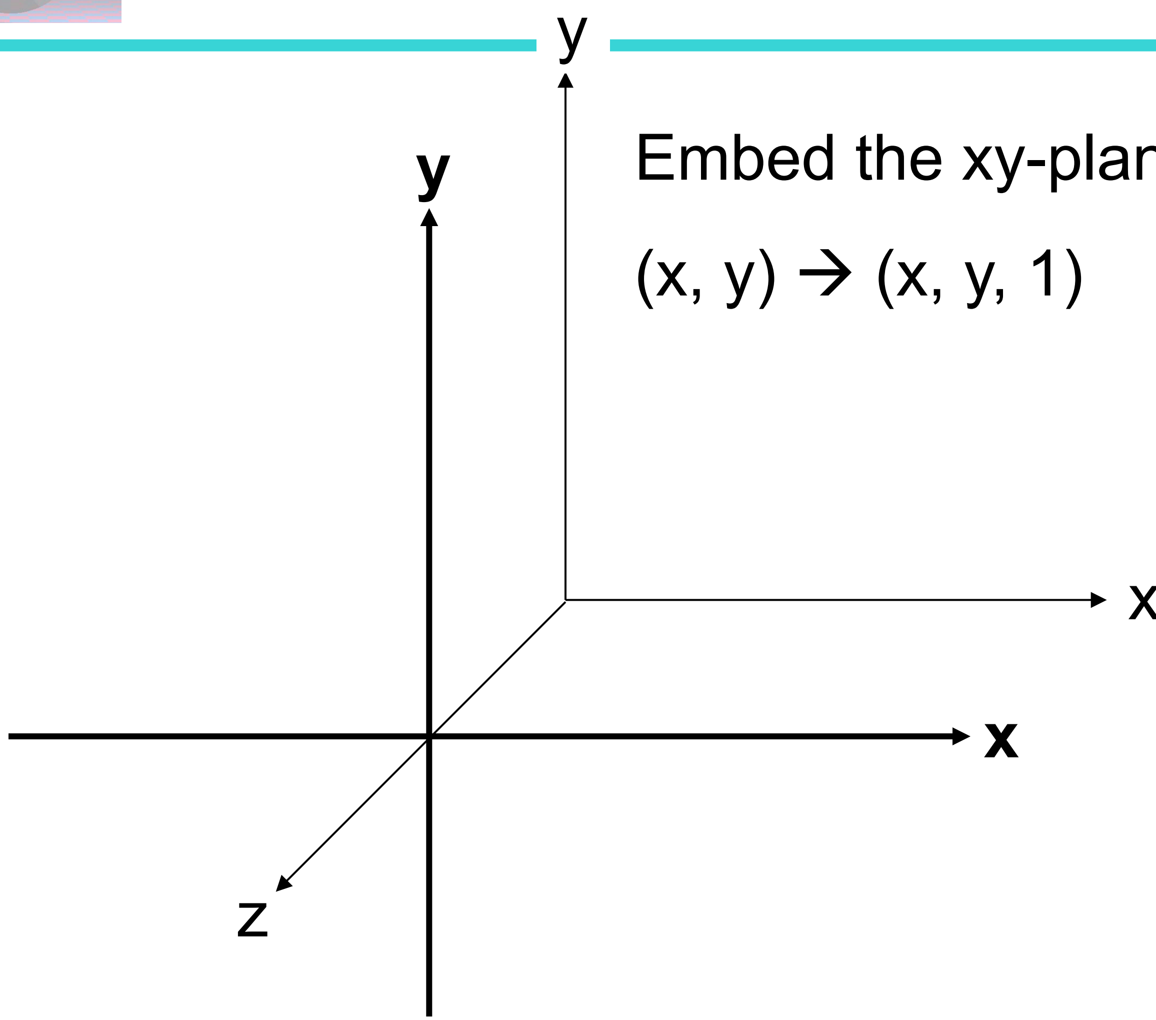
Translation

$(x, y) \rightarrow (x+a, y+b)$



This is not a linear transformation. The origin moves.

Homogeneous Coordinates



Embed the xy-plane in R^3 at $z = 1$.

$(x, y) \rightarrow (x, y, 1)$

2D Linear Transformations as 3D Matrices

Any 2D linear transformation can be represented by a 2x2 matrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

or a 3x3 matrix

$$\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \\ 1 \end{bmatrix}$$

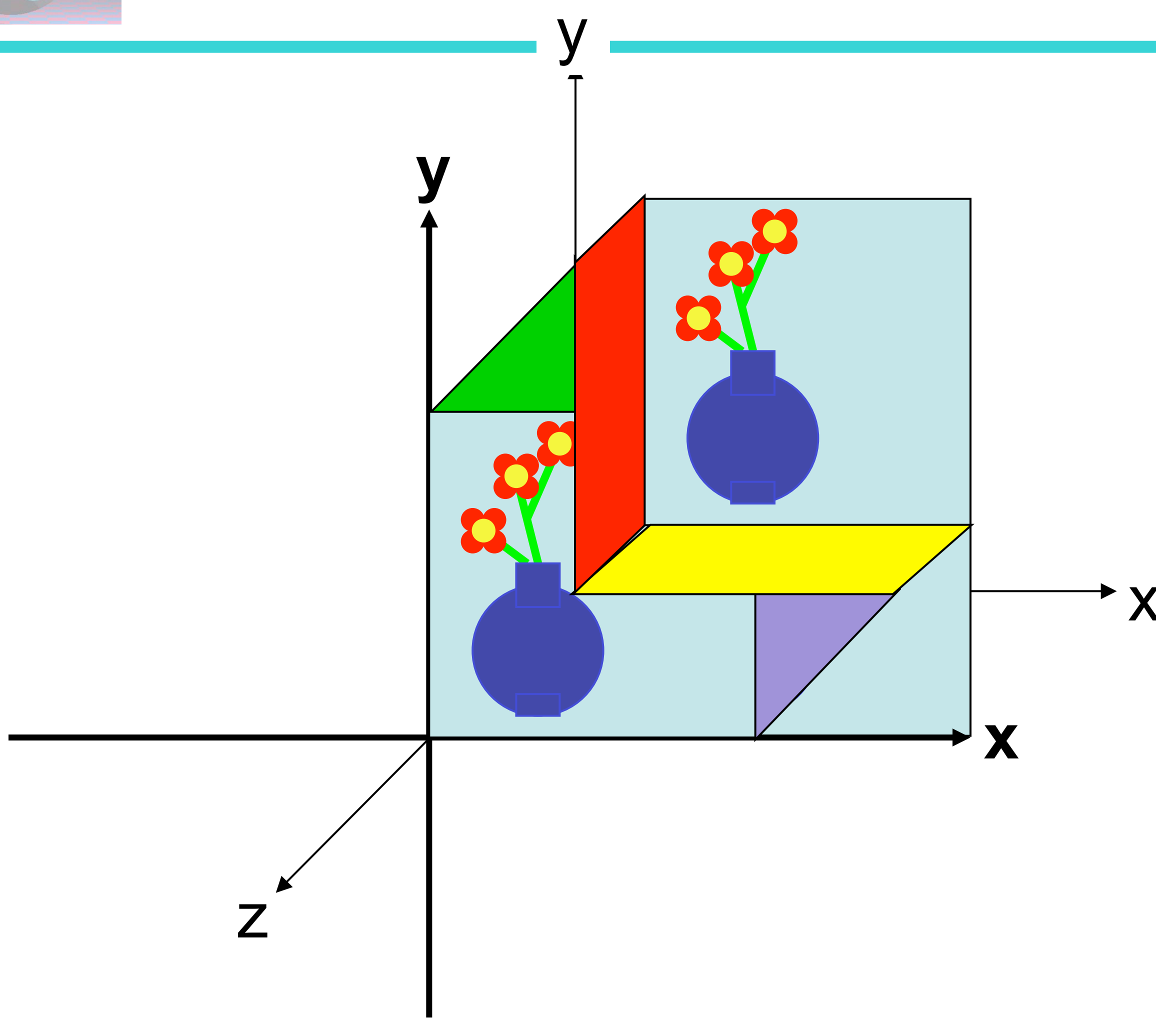
2D Linear Translations as 3D Matrices

Any 2D translation can be represented by a 3x3 matrix.

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix}$$

This is a 3D shear that acts as a translation on the plane z = 1.

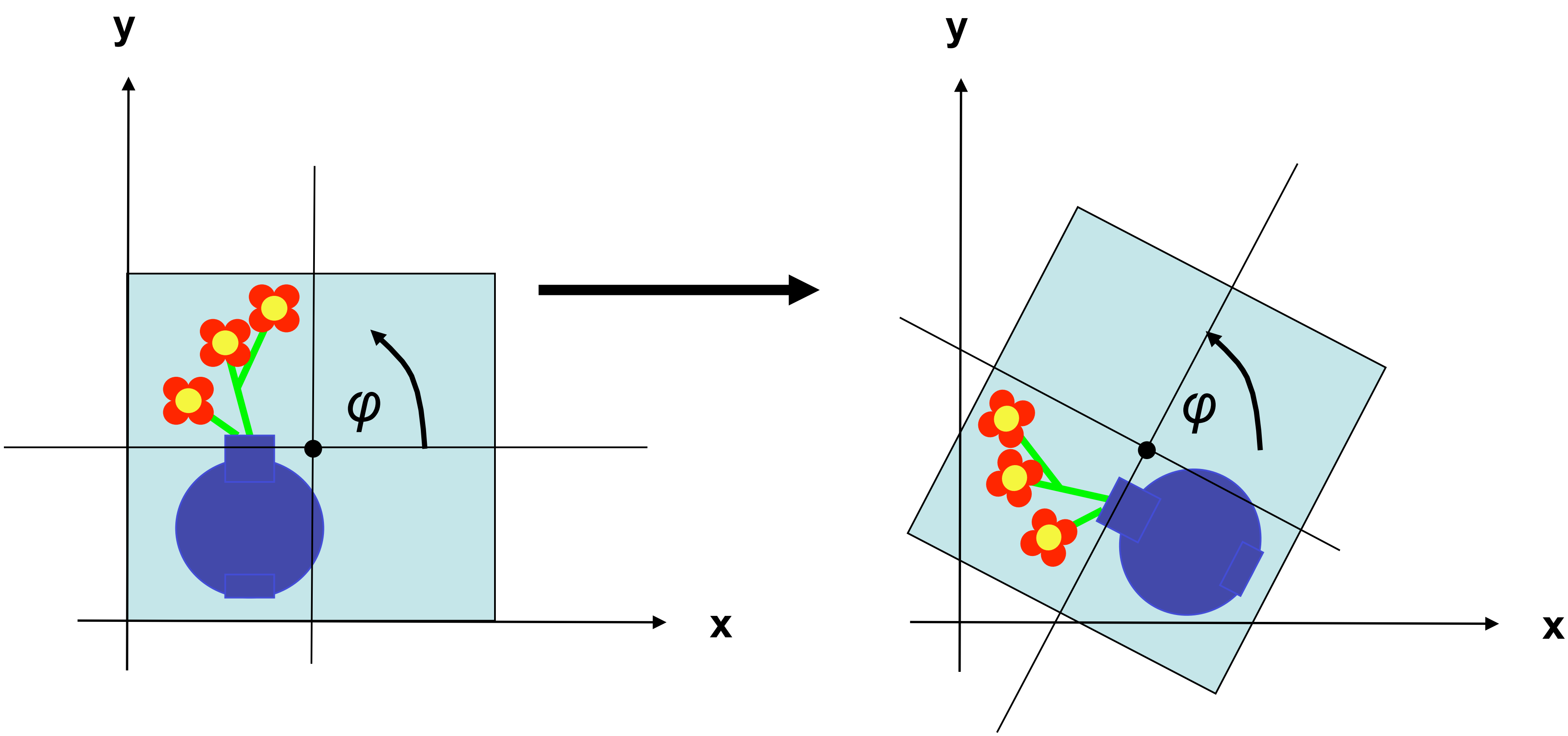
Translation as a Shear



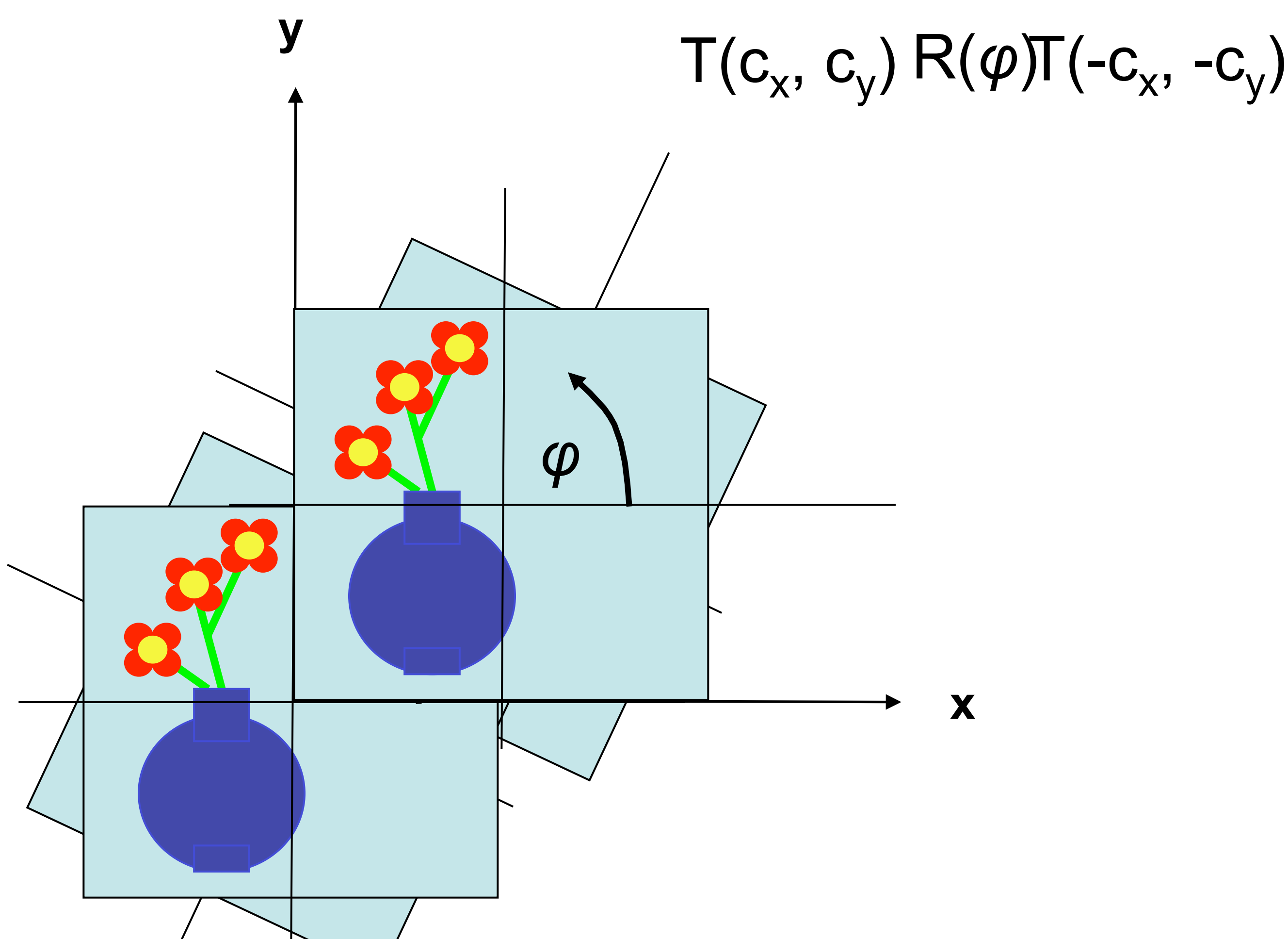
2D Affine Transformations

- An *affine transformation* is any transformation that preserves co-linearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation).
- With homogeneous coordinates, we can represent all 2D affine transformations as 3D linear transformations.
- We can then use matrix multiplication to transform objects.

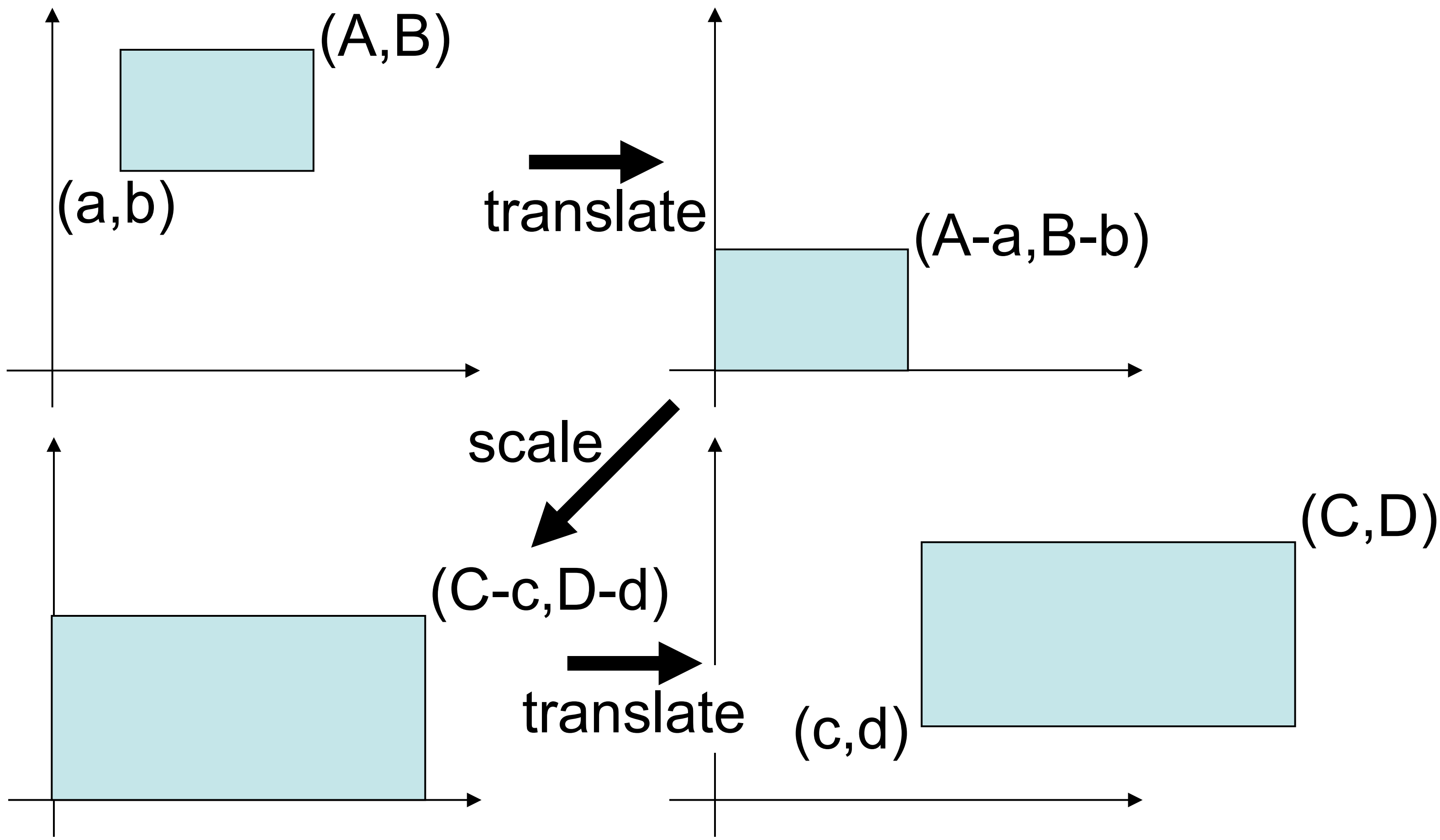
Rotation about an Arbitrary Point

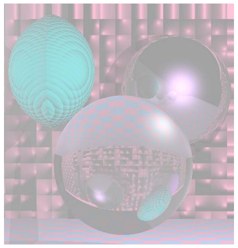


Rotation about an Arbitrary Point



Windowing Transforms





3D Transformations

Remember:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \leftrightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

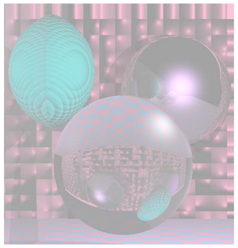
A 3D linear transformation can be represented by a 3x3 matrix.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \leftrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Affine Transformations

$$\text{scale}\left(s_x, s_y, s_z\right) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{translate}\left(t_x, t_y, t_z\right) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D Rotations

$$\mathrm{rotate_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{rotate_y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{rotate_z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$