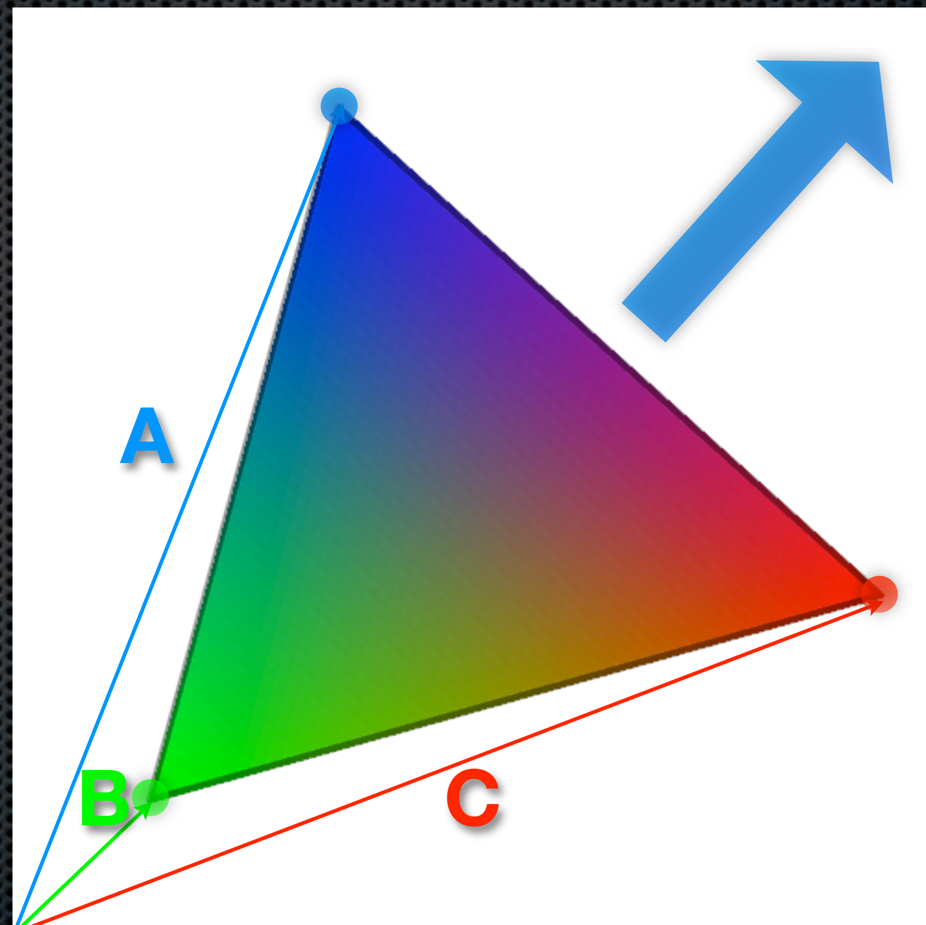
Introduction to Computer Graphics

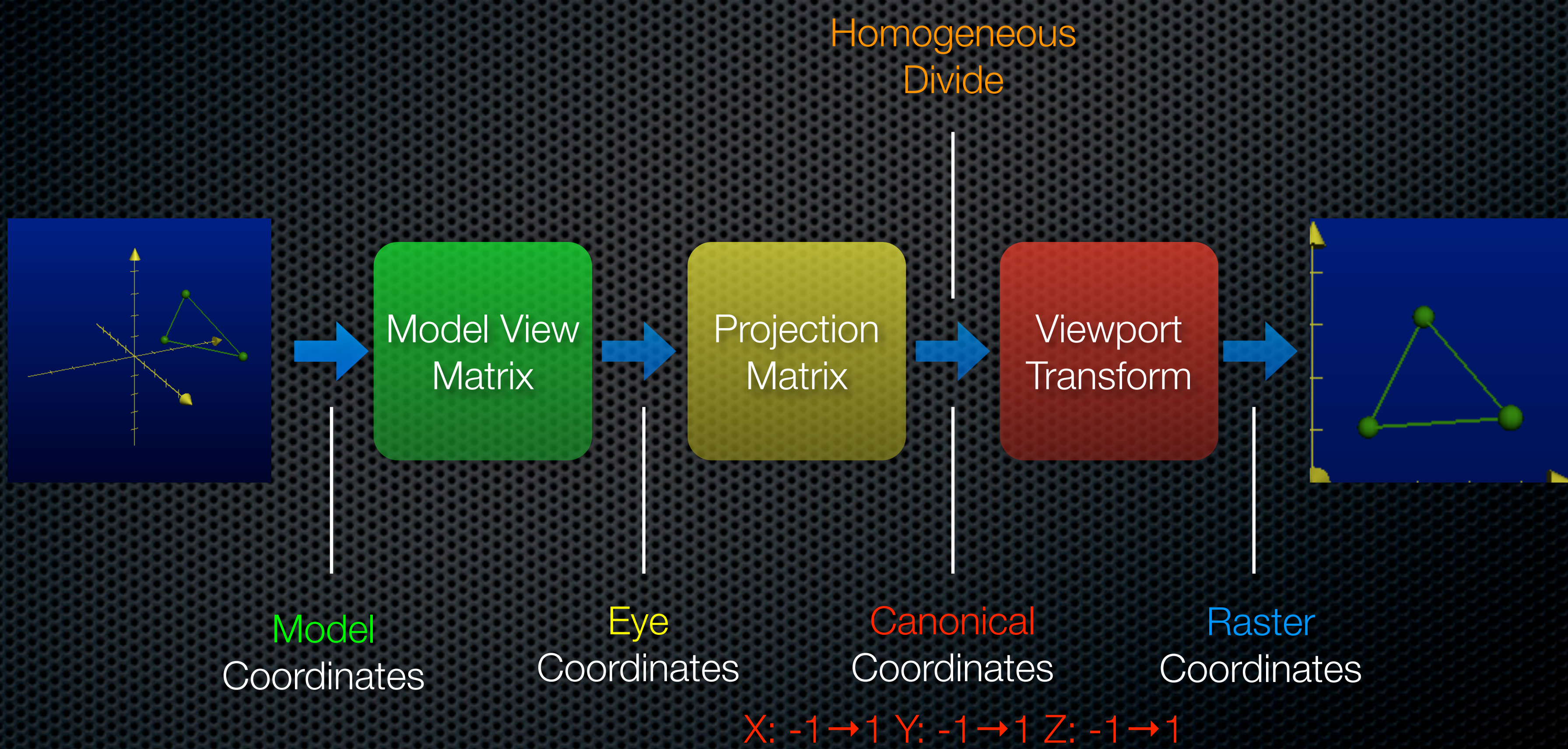
Cameras

What Can We Do So Far?

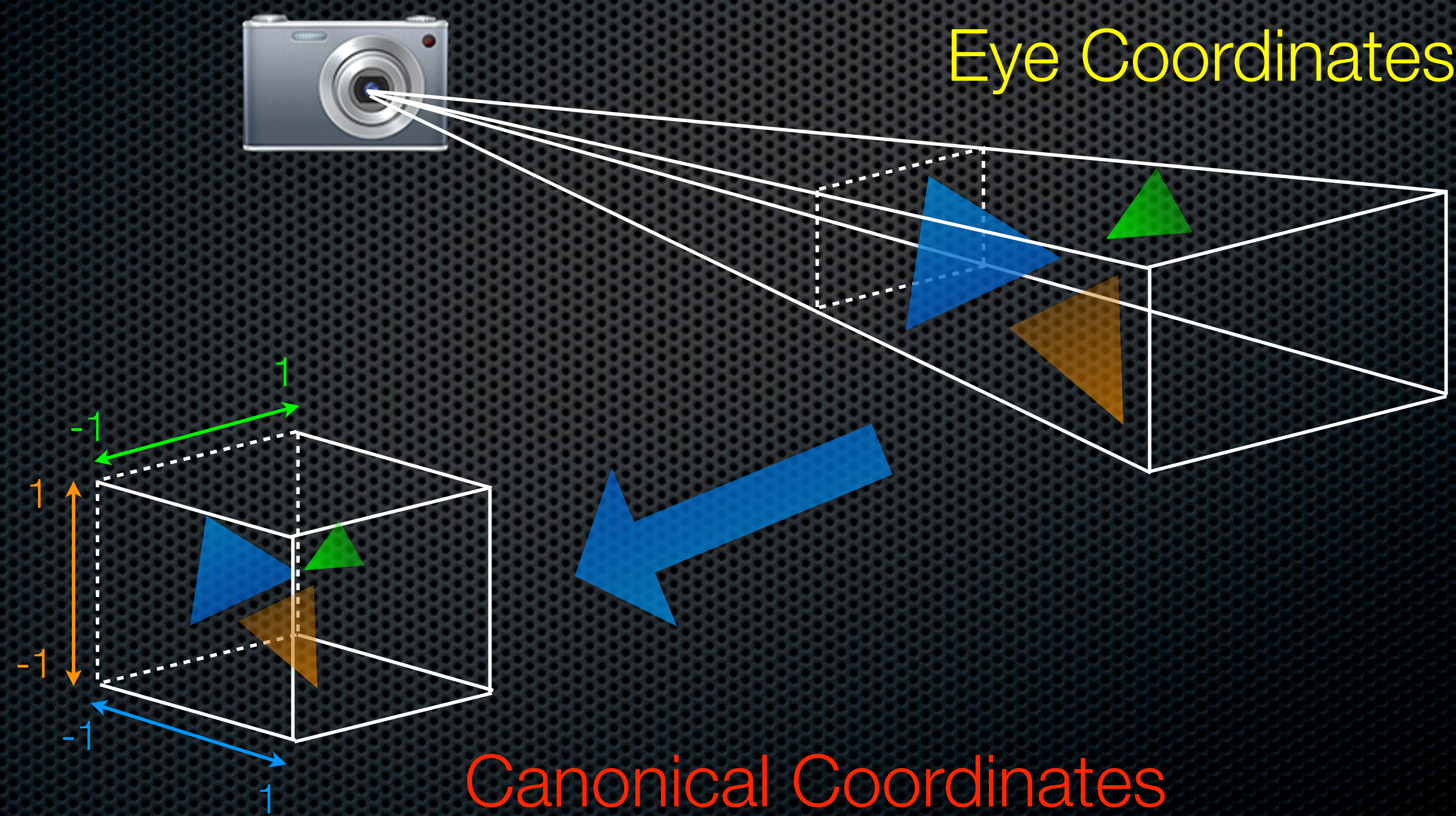
- Create and save rasters
- Draw triangles using interpolated colors
- Transform 3D input geometry
- Perspective depth with Z-buffering



Typical Matrices



Projection Matrix

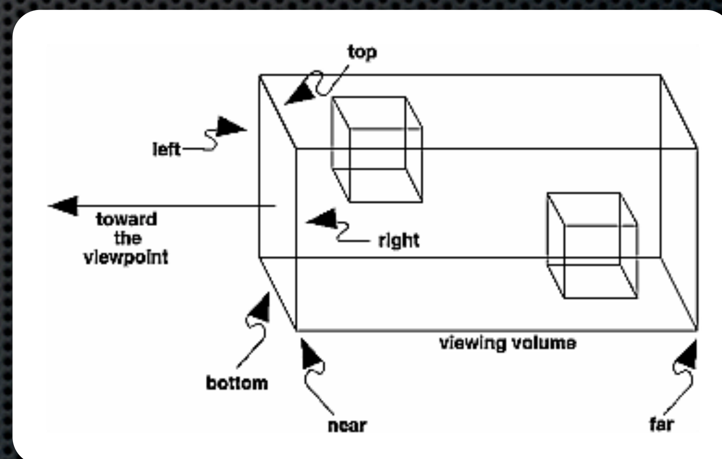


Projection - Orthographic

Combination Scale & Translation

ortho(l,r,b,t,n,f)

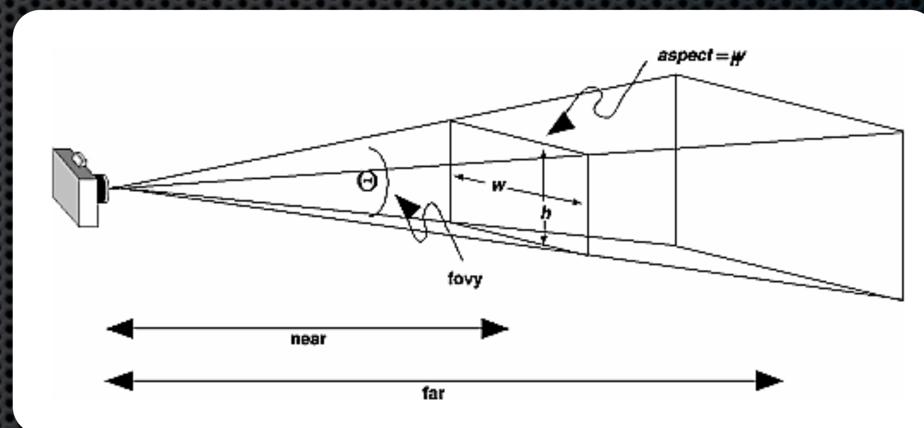
$$R = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } R^{-1} = \begin{bmatrix} \frac{r-l}{2} & 0 & 0 & \frac{r+l}{2} \\ 0 & \frac{t-b}{2} & 0 & \frac{t+b}{2} \\ 0 & 0 & \frac{f-n}{-2} & \frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Projection - Perspective

frustum(l,r,b,t,n,f)

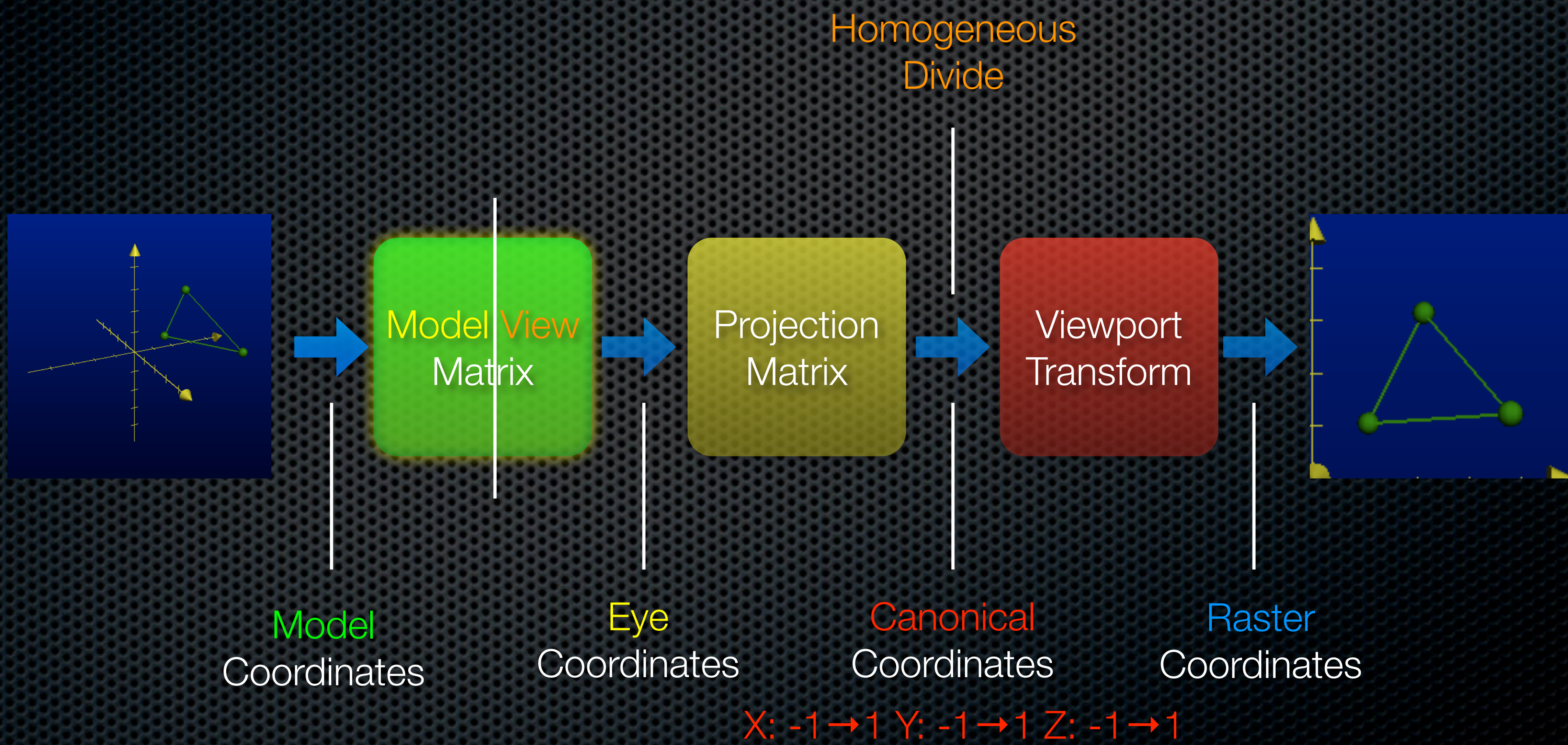
$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \text{ and } R^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & \frac{r+l}{2n} \\ 0 & \frac{t-b}{2n} & 0 & \frac{t+b}{2n} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{-(f-n)}{2fn} & \frac{f+n}{2fn} \end{bmatrix}$$



v = fovY
a = aspect
n = near
f = far

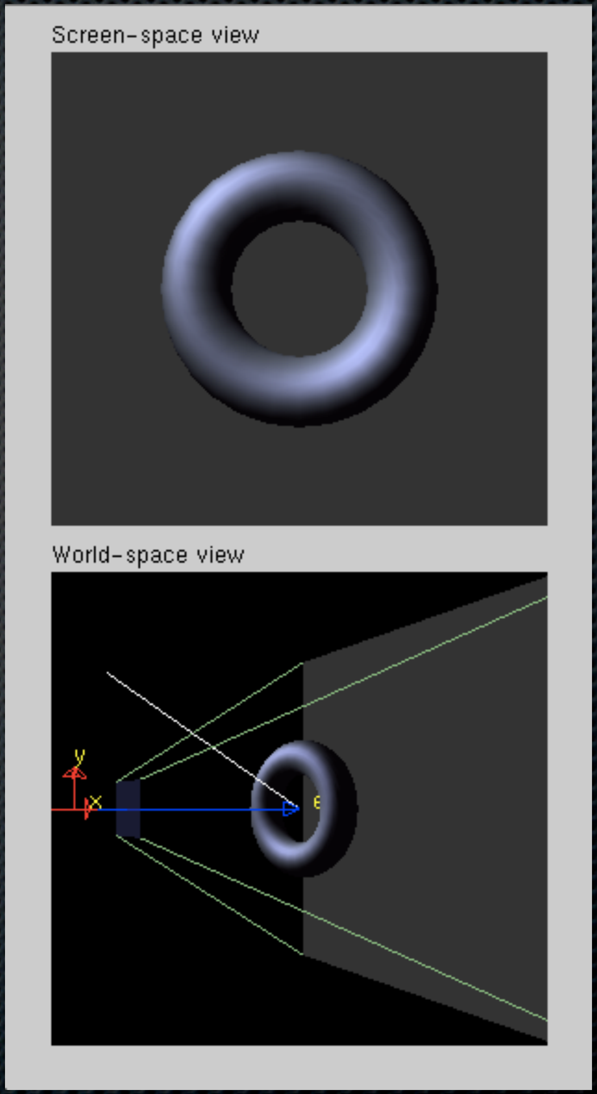
$$\begin{bmatrix} a/\tan(v/2) & 0 & 0 & 0 \\ 0 & 1/\tan(v/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -2nf/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Typical Matrices



Eye Coordinates

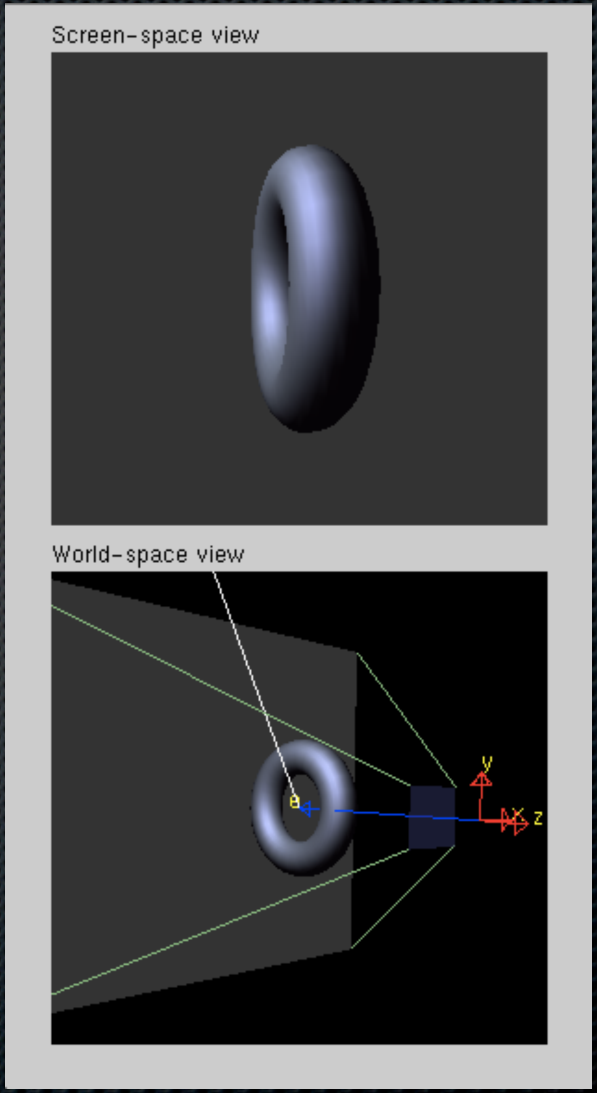
- User positioned at (0,0,0)
- **+**X axis to the user's right
- **+**Y axis points up
- User looks down **-**Z axis by the right hand rule



Eye Coordinates

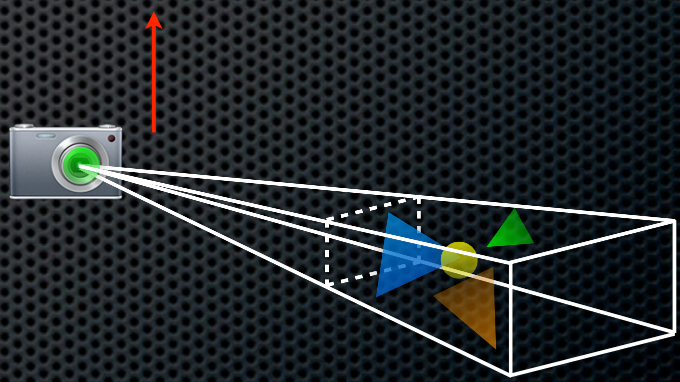
- User positioned at (0,0,0)
- **+**X axis to the user's right
- **+**Y axis points up
- User looks down **-**Z axis by the right hand rule

The world literally must revolve around you!



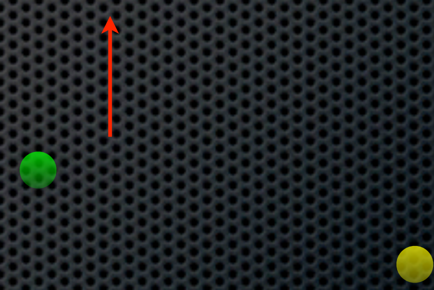
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be “up”
- User looks down vector between eye point and spot point



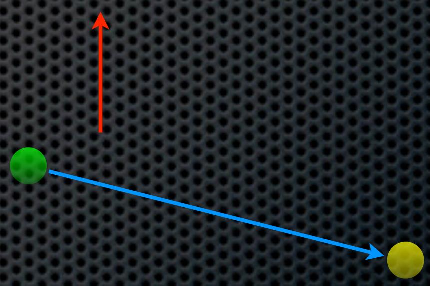
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be “up”
- User looks down vector between eye point and spot point



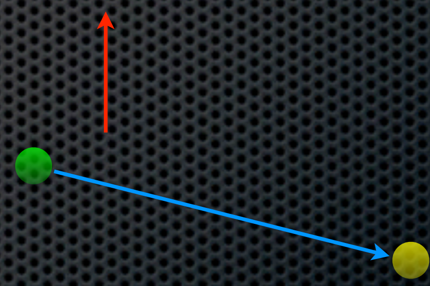
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down vector between eye point and spot point



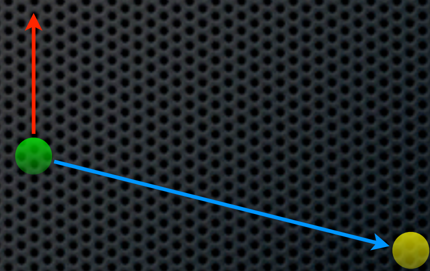
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be “up”
- User looks down look vector



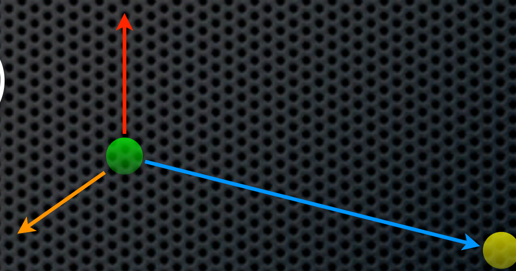
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector



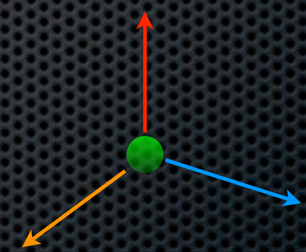
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"



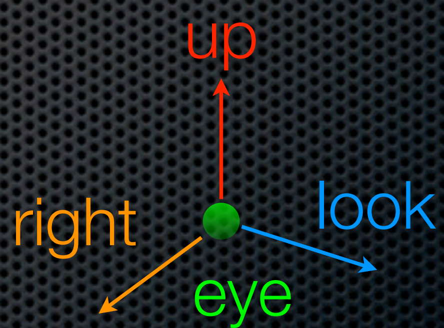
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be “up”
- User looks down look vector
- There exists a vector describing the direction “right”



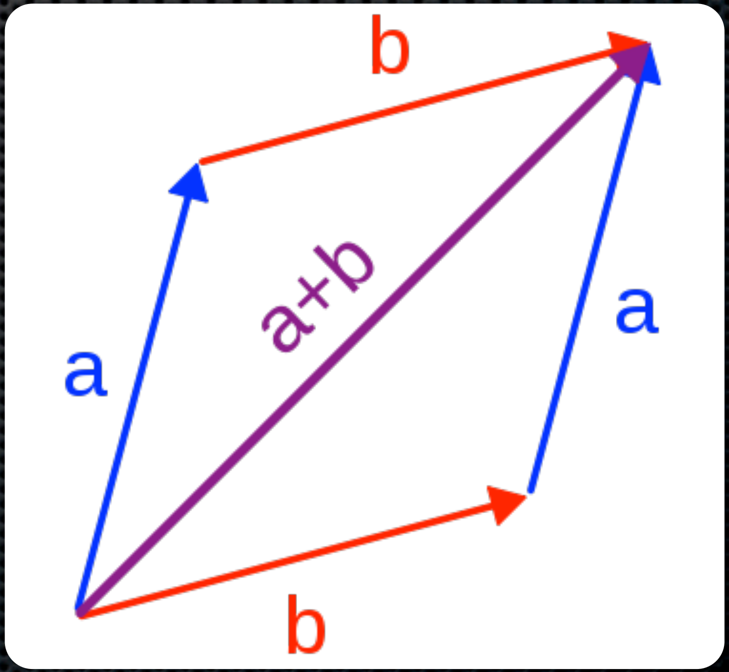
Better Definition

- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be “up”
- User looks down look vector
- There exists a vector describing the direction “right”

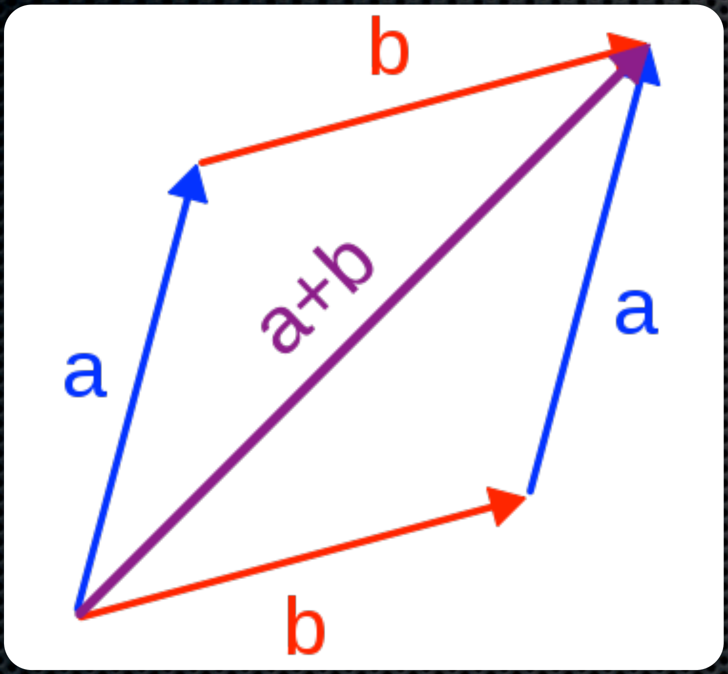


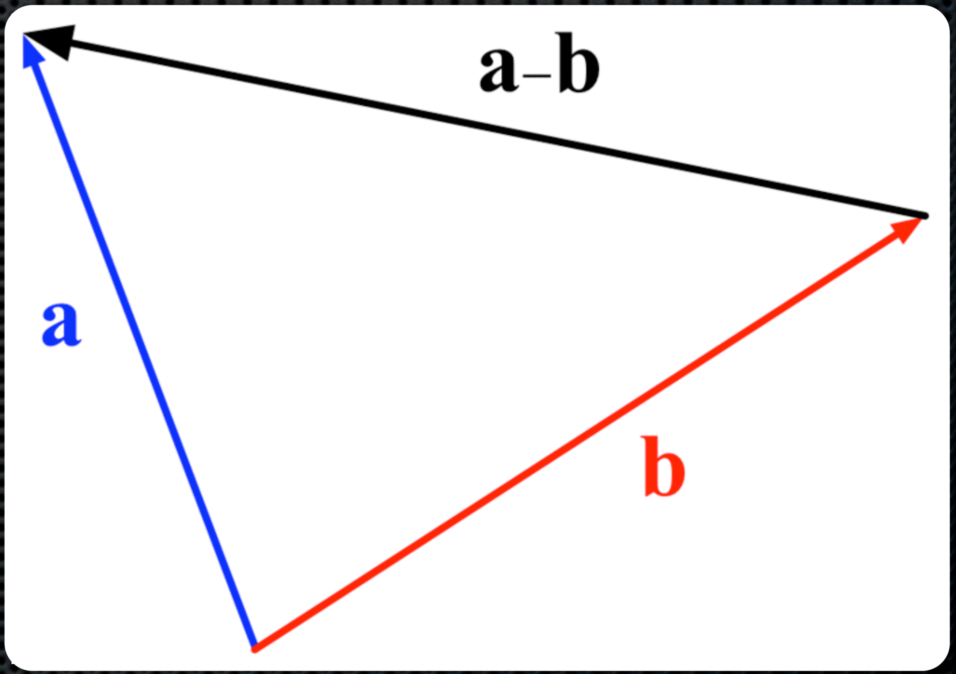
Vectors

- Vectors
  - Addition & Subtraction
  - Scalar Multiplication
  - Magnitude & Normalization
  - Dot & Cross Product

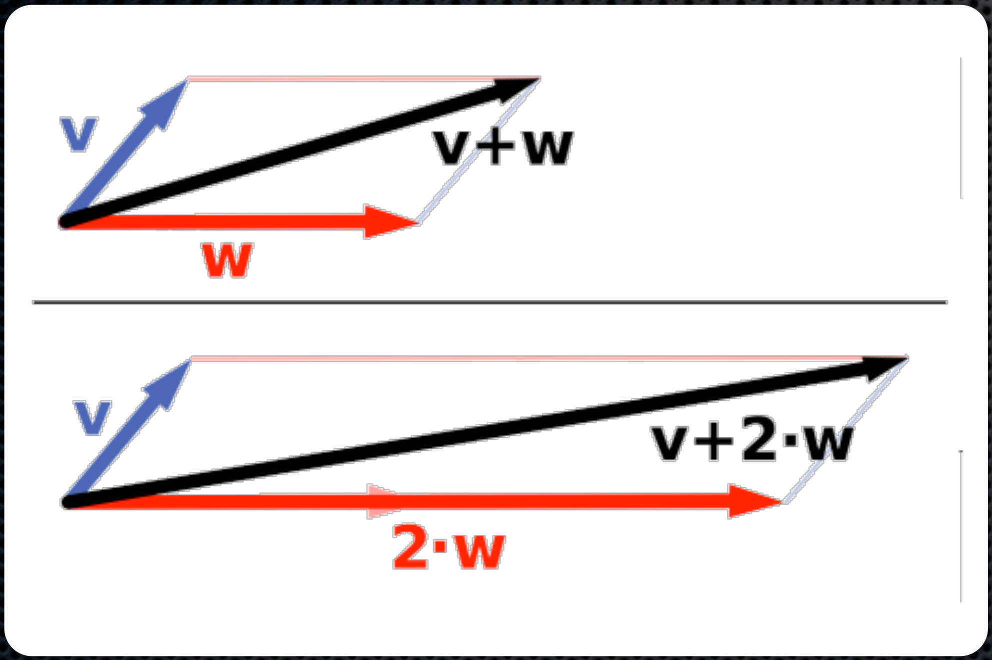


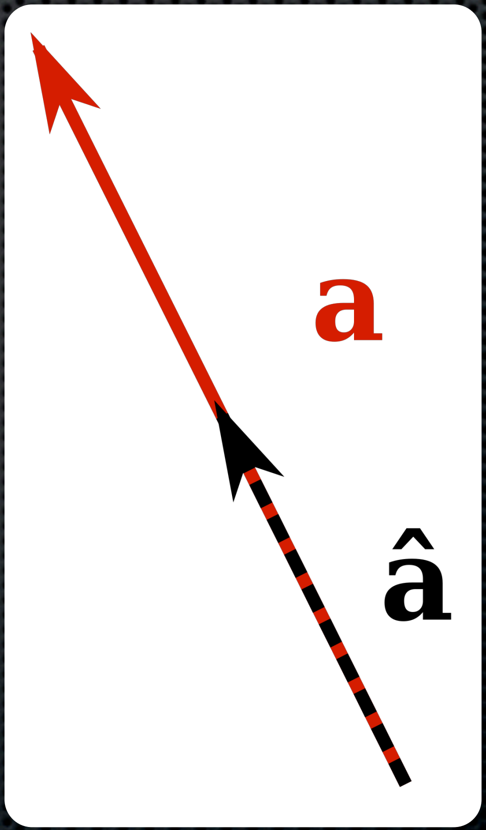
Vector Addition & Subtraction





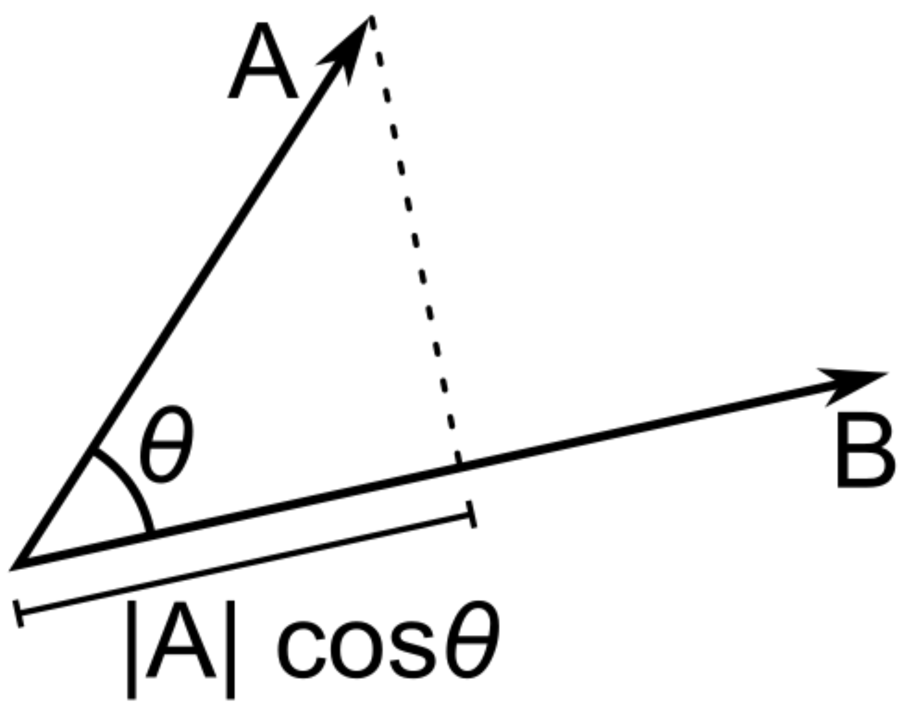
Vector-Scalar Multiplication



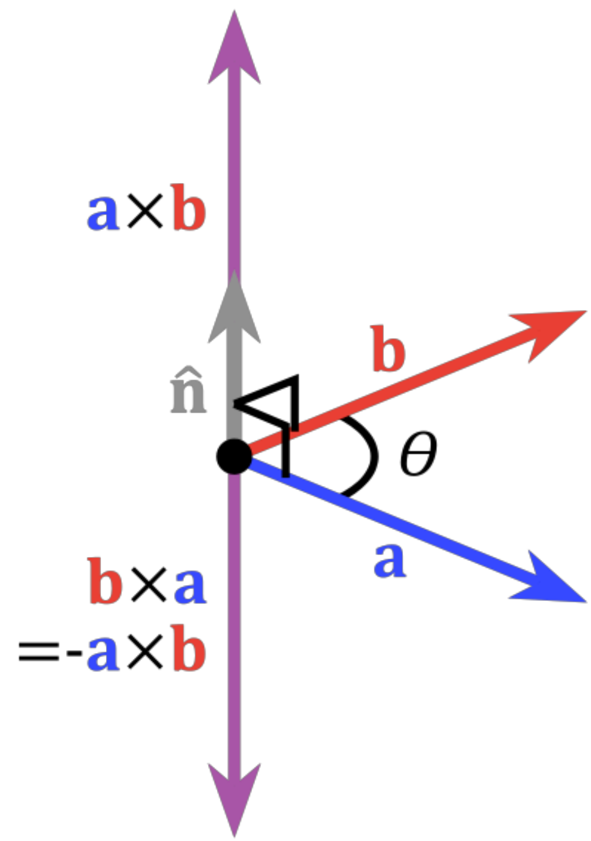


$$\hat{a} = \frac{1}{|a|}\, a$$

Vector Dot & Cross Product

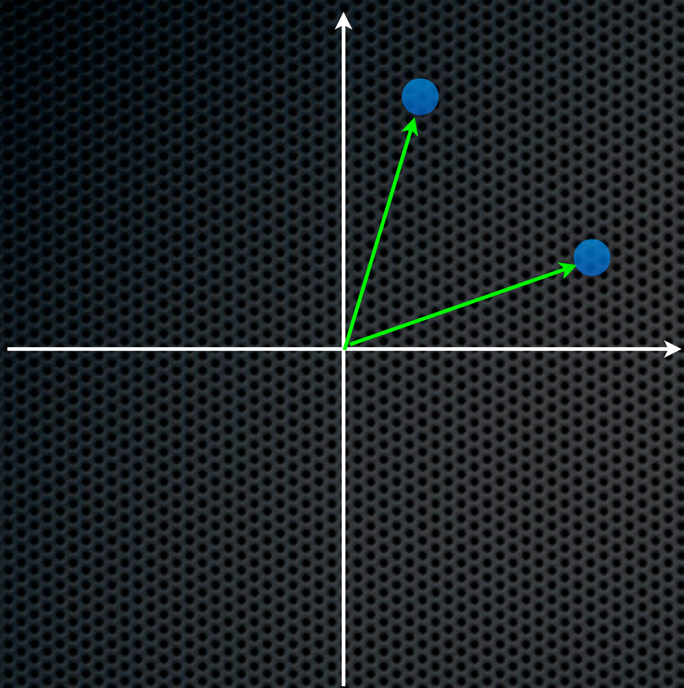


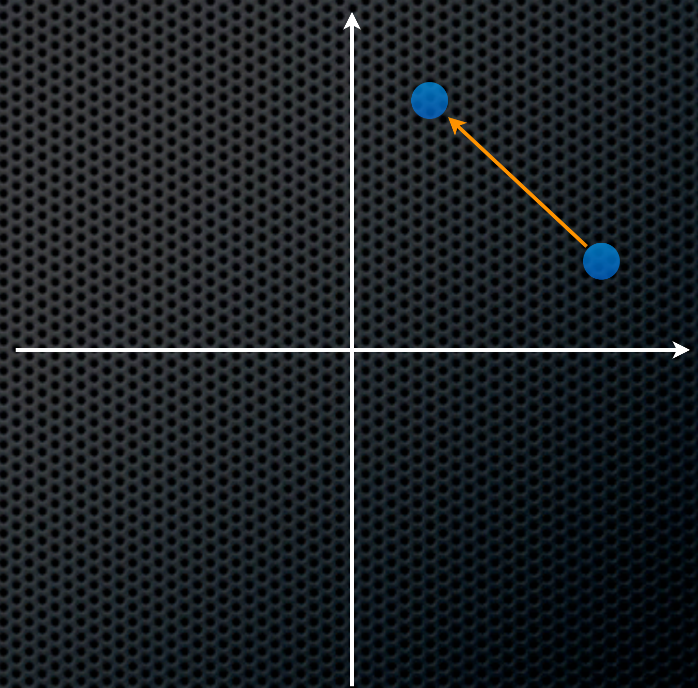
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \cos\theta$$



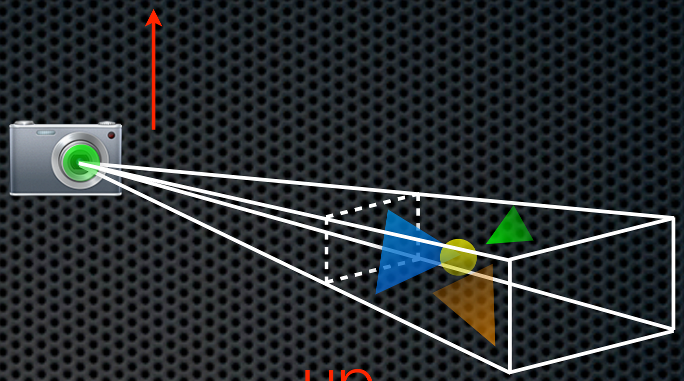
$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\| \sin(\theta)\,\mathbf{n}$$

Point and Free Vectors

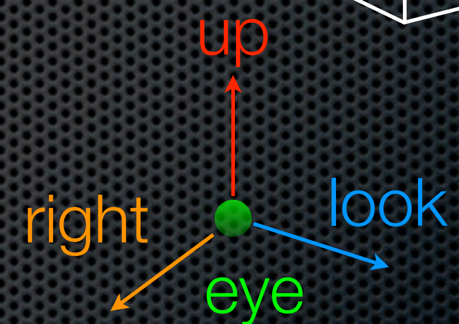




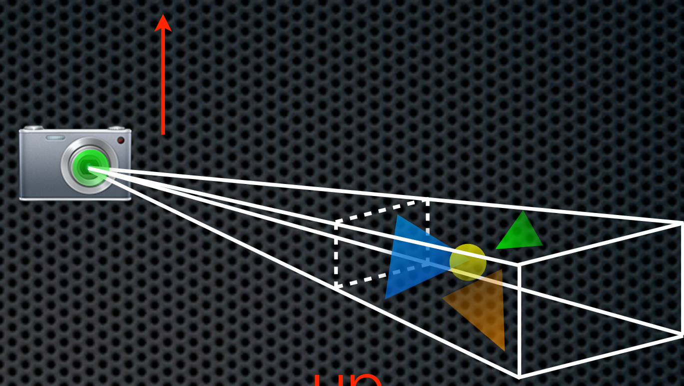
Camera Matrix



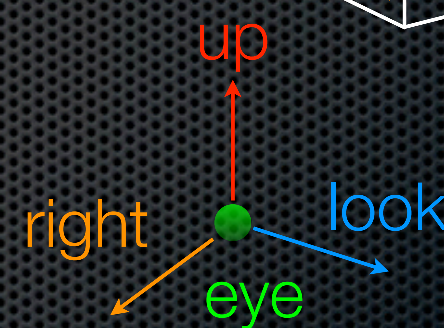
- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"



Camera Matrix

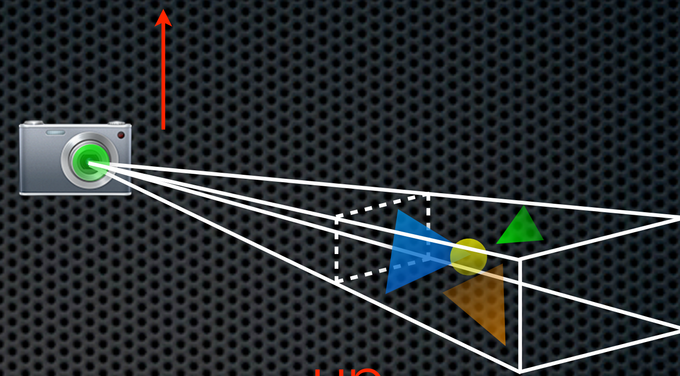


- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"

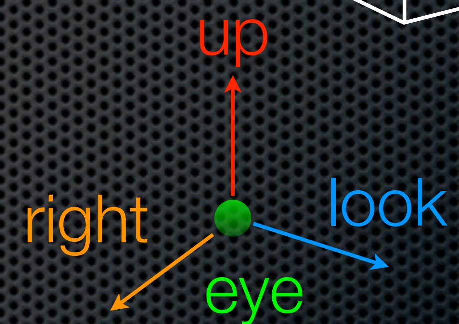


look = spot - eye

Camera Matrix



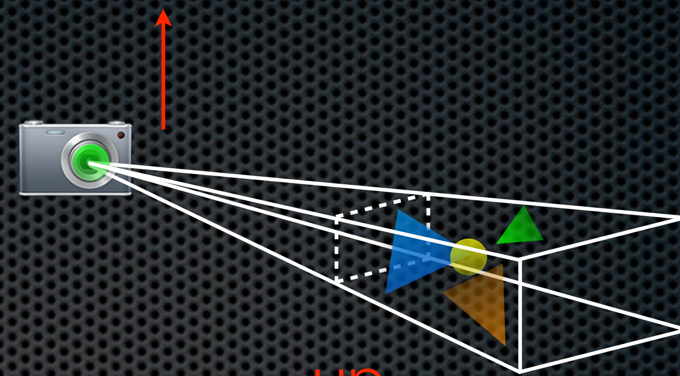
- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"



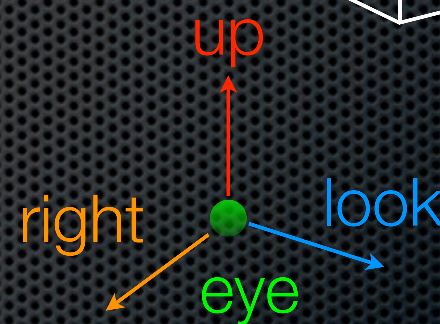
look = spot - eye

right = look x up

Camera Matrix



- User positioned at (eye_X, eye_Y, eye_Z)
- User looks at $(spot_X, spot_Y, spot_Z)$
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"

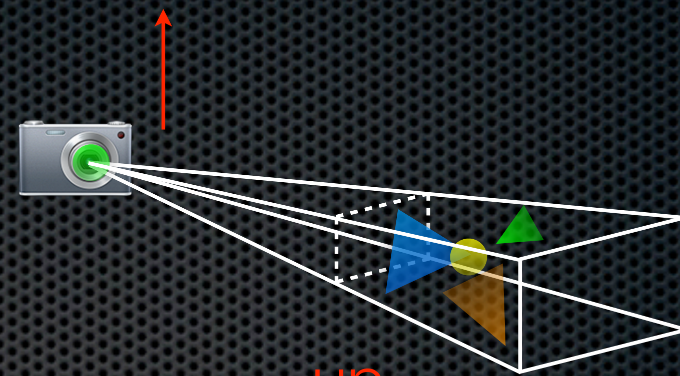


look = spot - eye

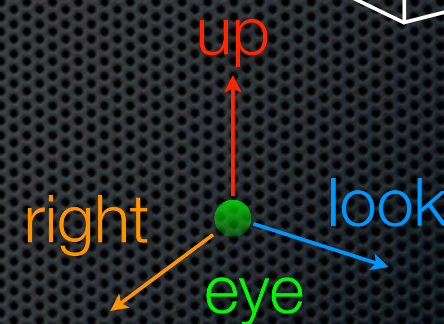
right = look x up

up = right x look

Camera Matrix



- User positioned at (eye_X,eye_Y,eye_Z)
- User looks at ($spot_X$, $spot_Y$, $spot_Z$)
- Any direction can be "up"
- User looks down look vector
- There exists a vector describing the direction "right"



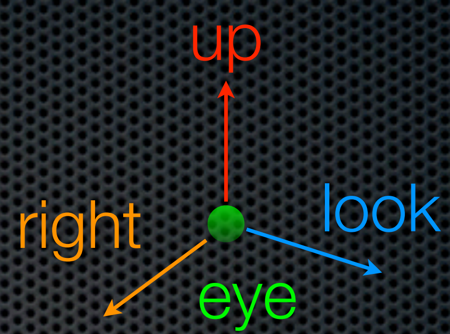
look = spot - eye

right = look x up

up = right x look

look, right, up are normalized

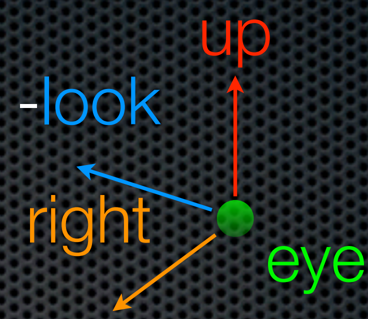
Camera Matrix



Change of Coordinate System Matrix

$$\begin{bmatrix} right_x & right_y & right_z & 0 \\ up_x & up_y & up_z & 0 \\ look_x & look_y & look_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

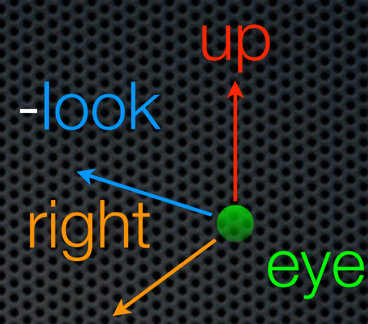
Camera Matrix



Change of Coordinate System Matrix

$$\begin{bmatrix} right_x & right_y & right_z & 0 \\ up_x & up_y & up_z & 0 \\ -look_x & -look_y & -look_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera Matrix



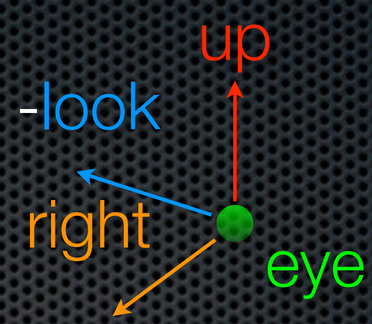
Change of Coordinate System Matrix

$$\begin{bmatrix} right_x & right_y & right_z & 0 \\ up_x & up_y & up_z & 0 \\ -look_x & -look_y & -look_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate Camera to Origin

Translate($-eye_x$, $-eye_y$, $-eye_z$)

Camera Matrix



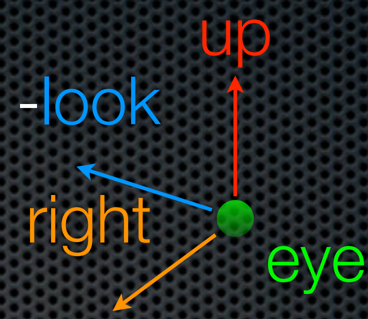
Change of Coordinate System Matrix

$$B = \begin{bmatrix} right_x & right_y & right_z & 0 \\ up_x & up_y & up_z & 0 \\ -look_x & -look_y & -look_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate Camera to Origin

$T = \text{Translate}(-eye_x, -eye_y, -eye_z)$

Camera Matrix



Change of Coordinate System Matrix

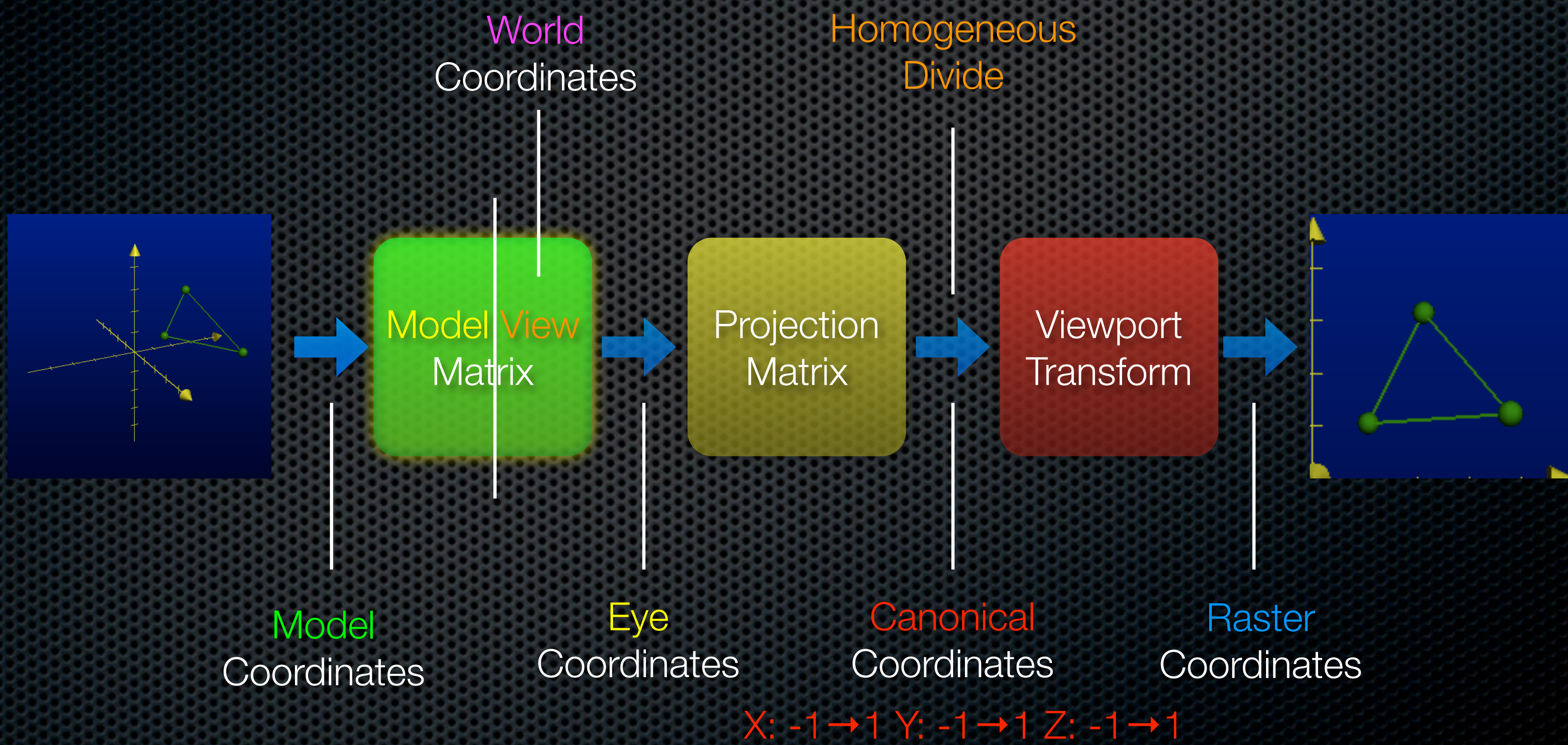
$$B = \begin{bmatrix} right_x & right_y & right_z & 0 \\ up_x & up_y & up_z & 0 \\ -look_x & -look_y & -look_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate Camera to Origin

$$T = \text{Translate}(-eye_x, -eye_y, -eye_z)$$

$$M = BT$$

Typical Matrices



Either This
10,000 Times
S
V
=
SV
R
SV
=
RSV
T
RSV
=
TRSV
Notice the ordering!
T
R
S
=
TRS
And This
10,000 Times
TRS
V
=
TRSV

Model View Matrix Parts

Camera matrix is concatenated **first**

Model View Matrix Parts

Model View Matrix | V = Camera Matrix | T | R | S | V

Camera matrix is concatenated **first**