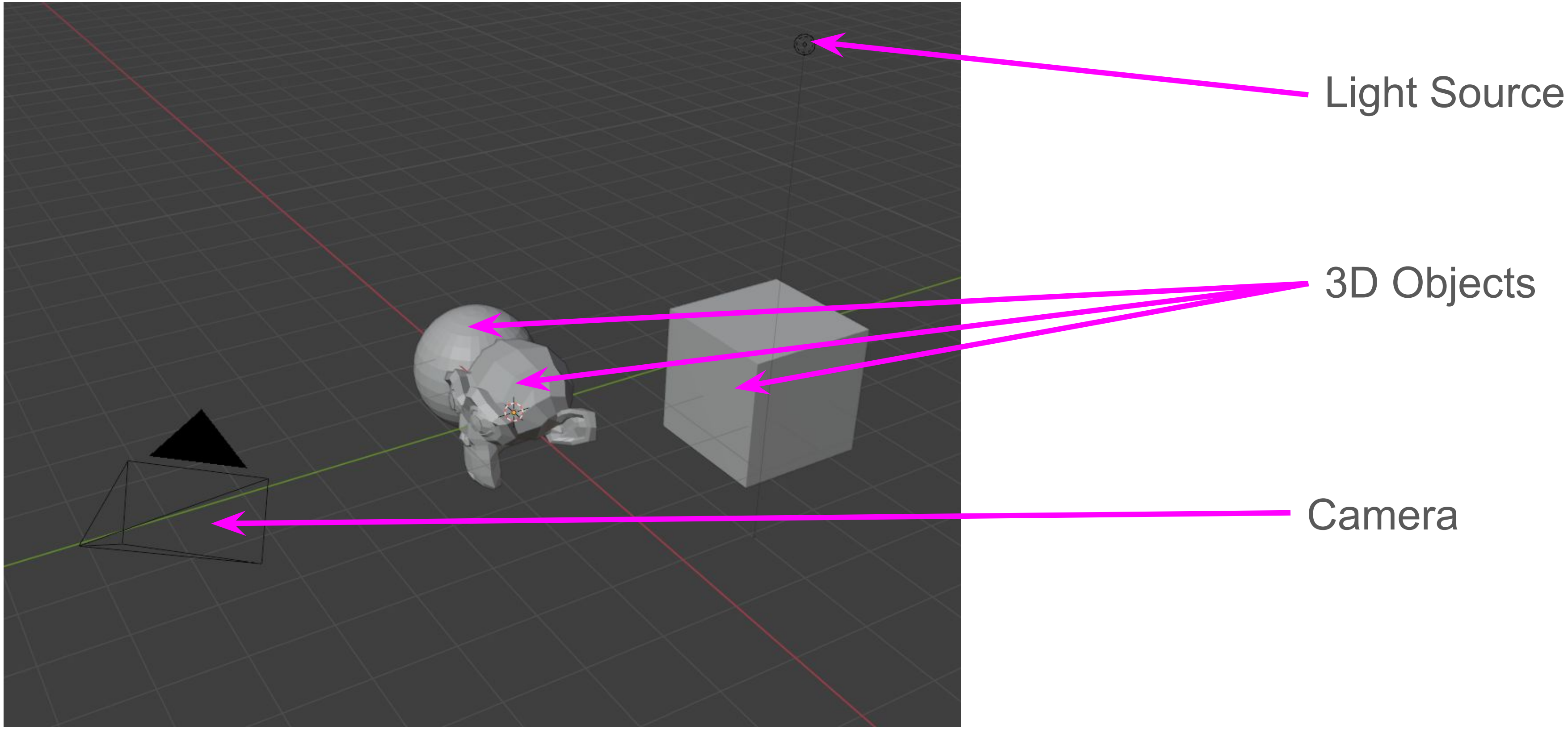
3D Graphics

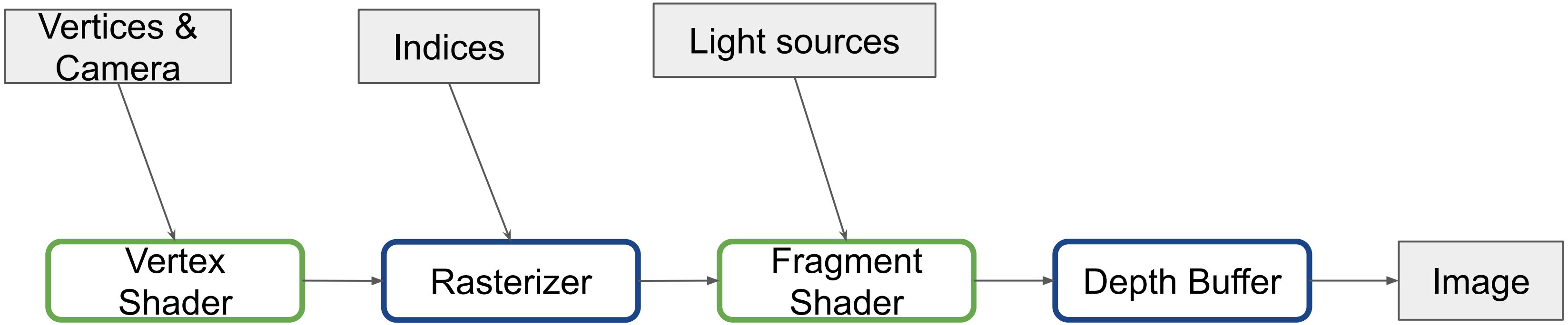
The rendering pipeline

Reminder : Scene



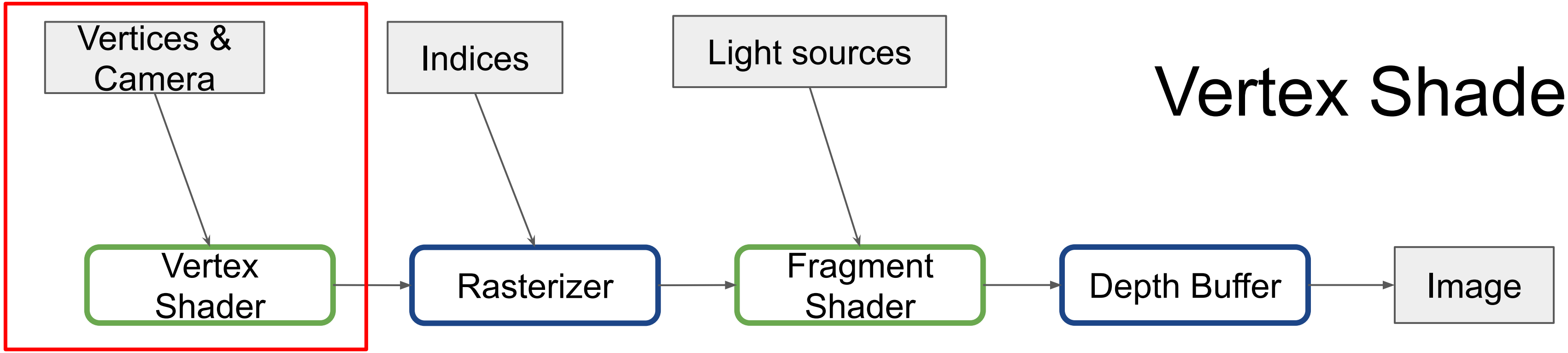
The rendering pipeline

Helps us go from a 3D scene to a 2D image



Minimal rendering pipeline

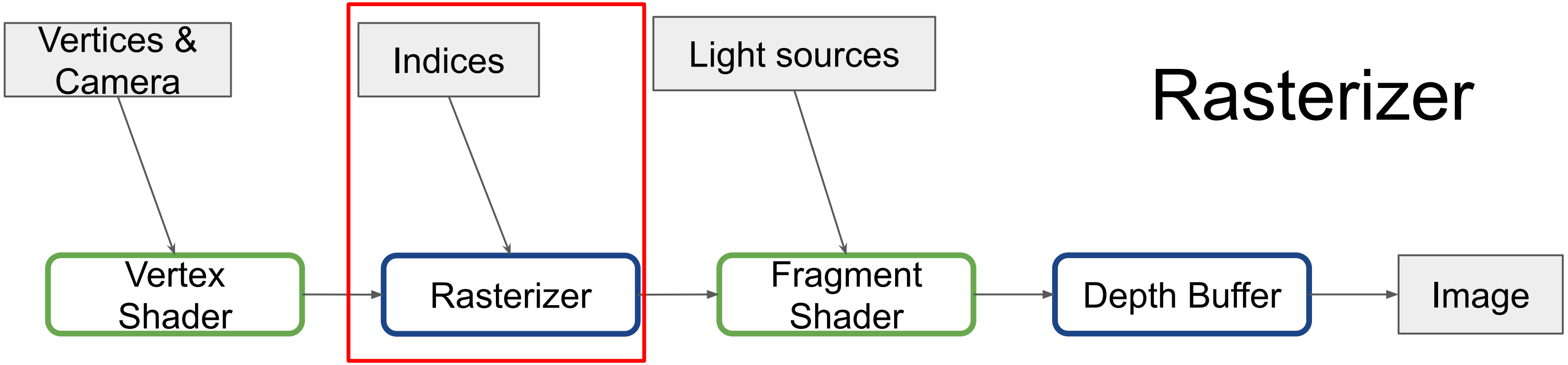
Vertex Shader



Re-express vertices in the camera coordinates system

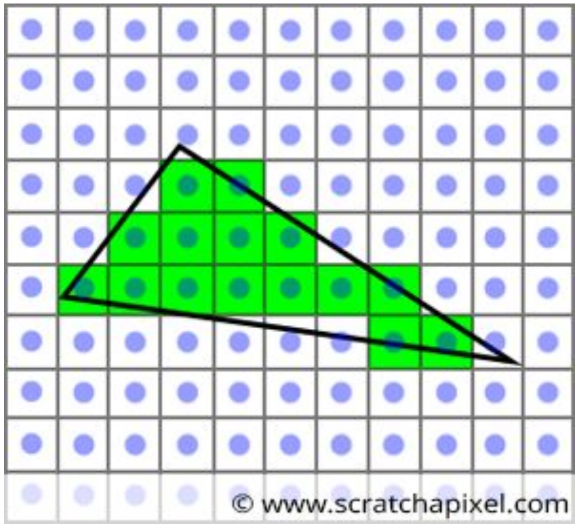
Project vertices in the frustum

Rasterizer

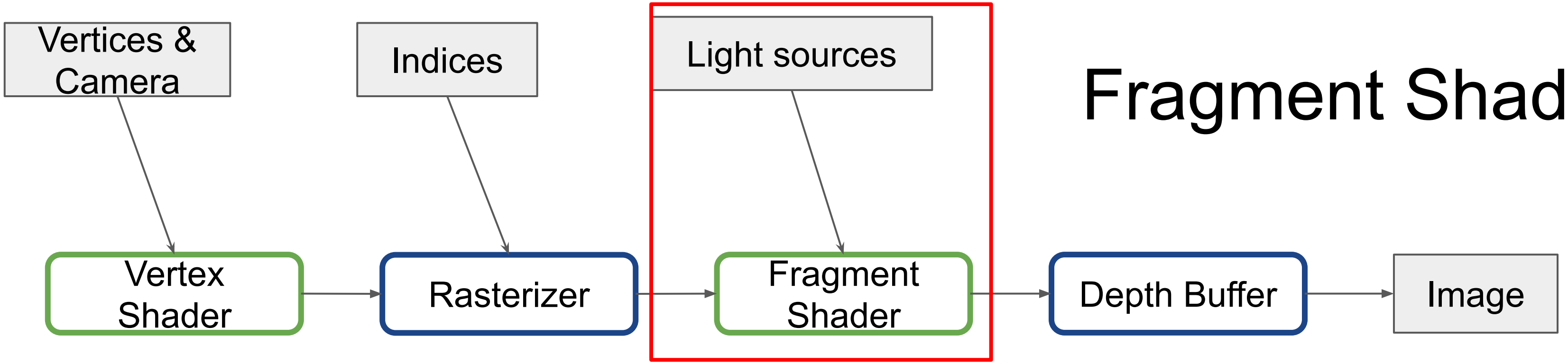


Find which pixel is inside which triangle

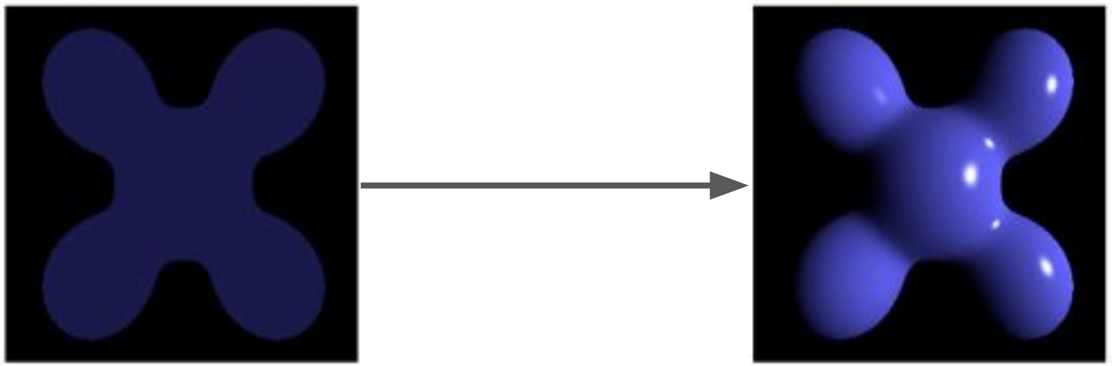
Emit fragments (candidates pixel)



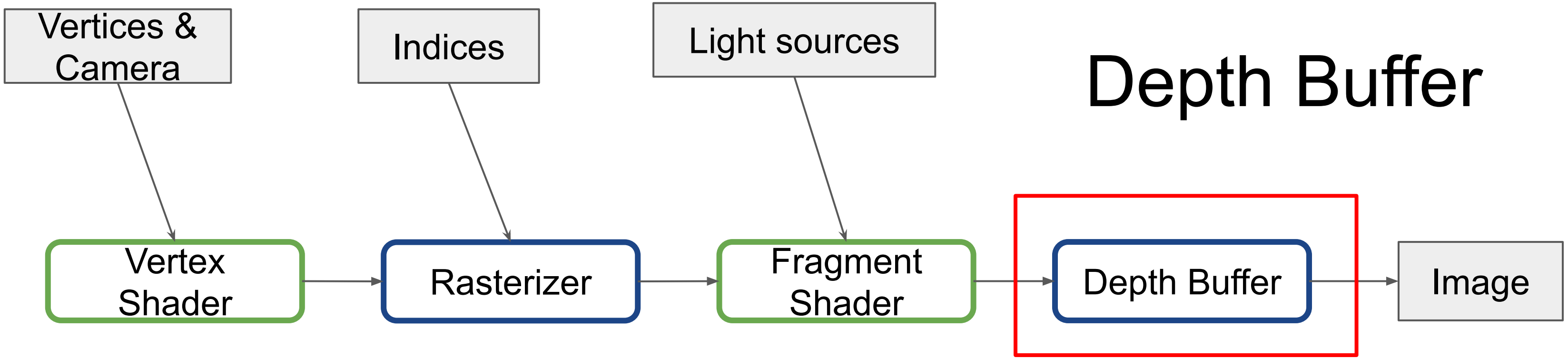
Fragment Shader



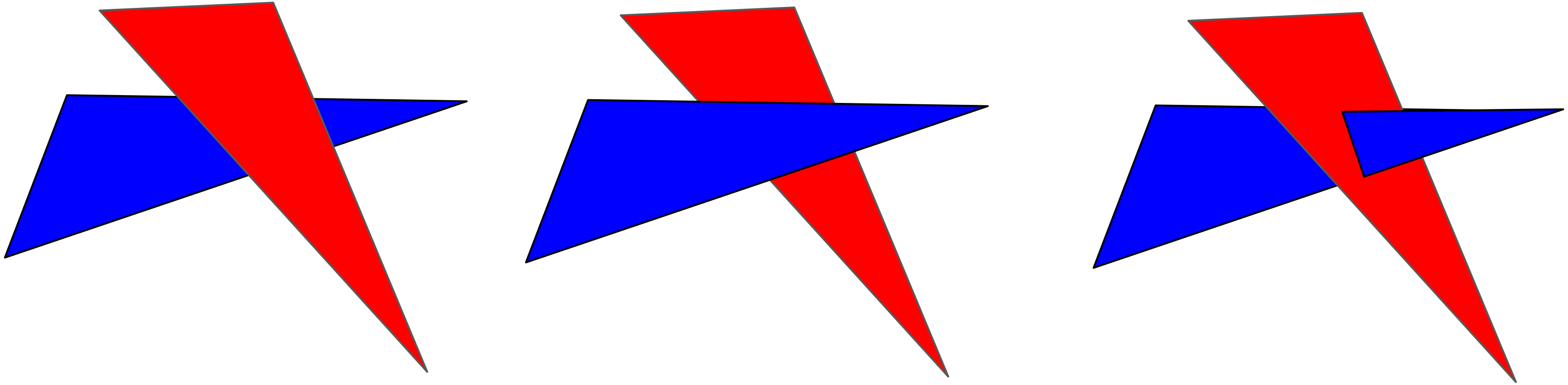
Compute the color to give each fragments



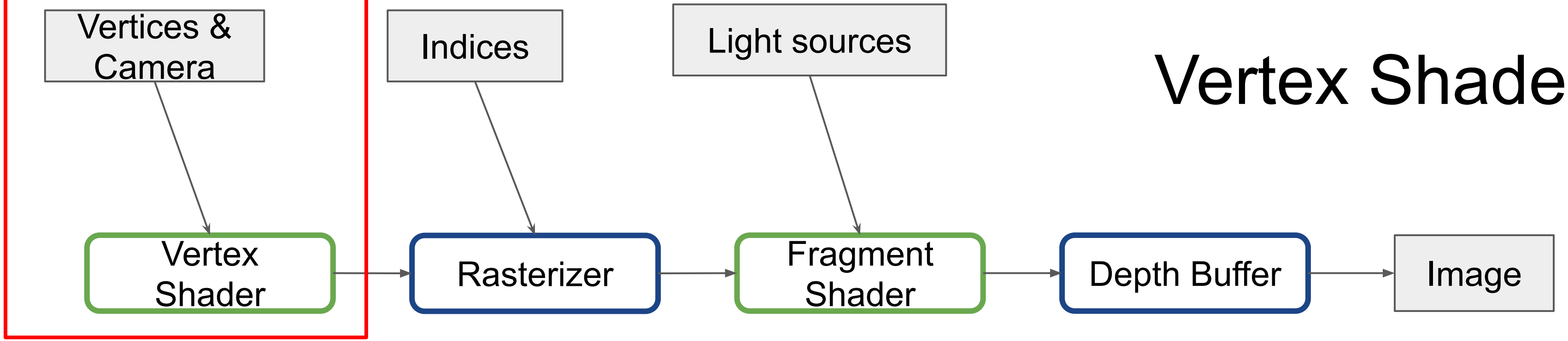
Depth Buffer



Choose which Fragment get to become a pixel using a Depth test



Vertex Shader

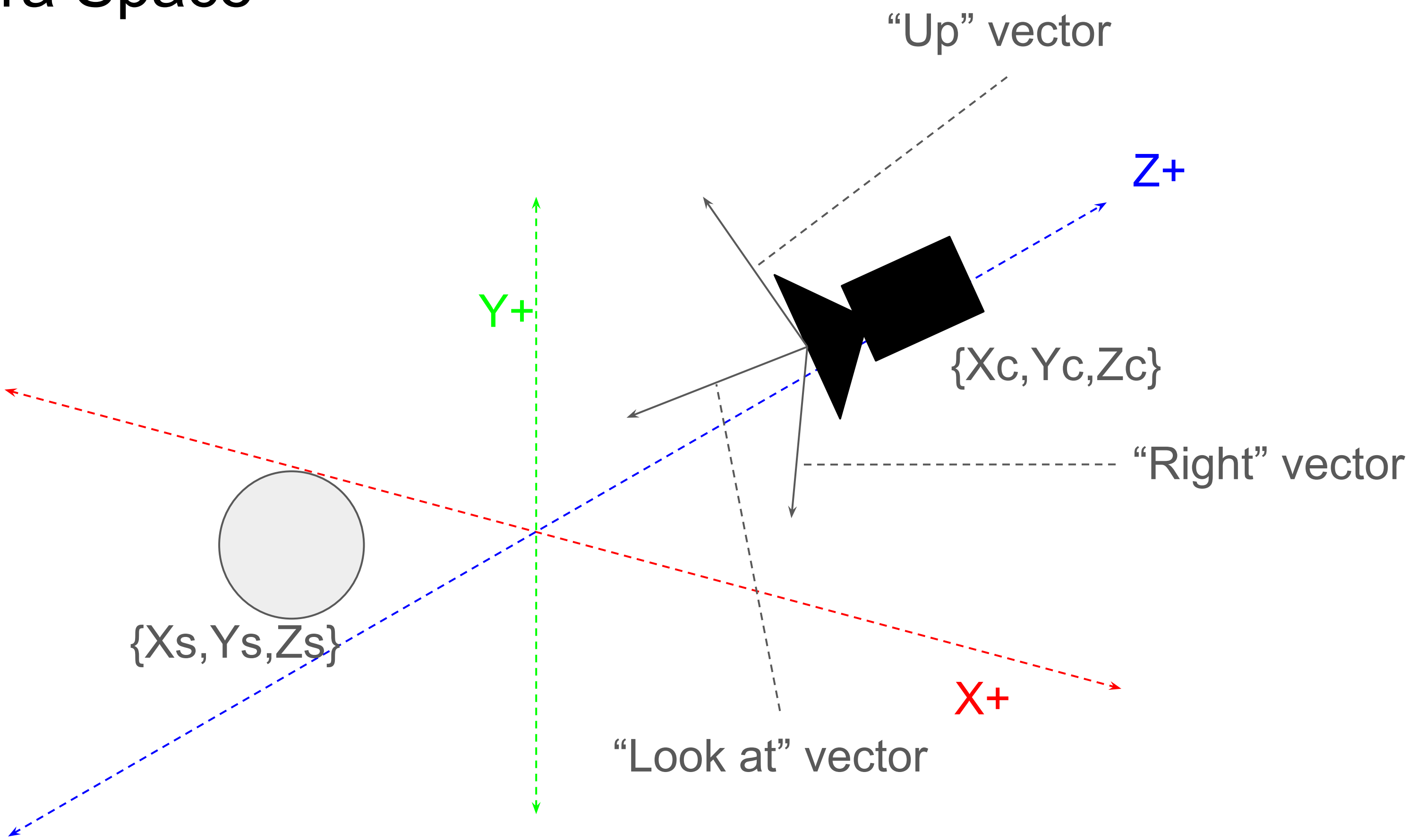


Vertex Shader

Re-express vertices in the camera coordinates system

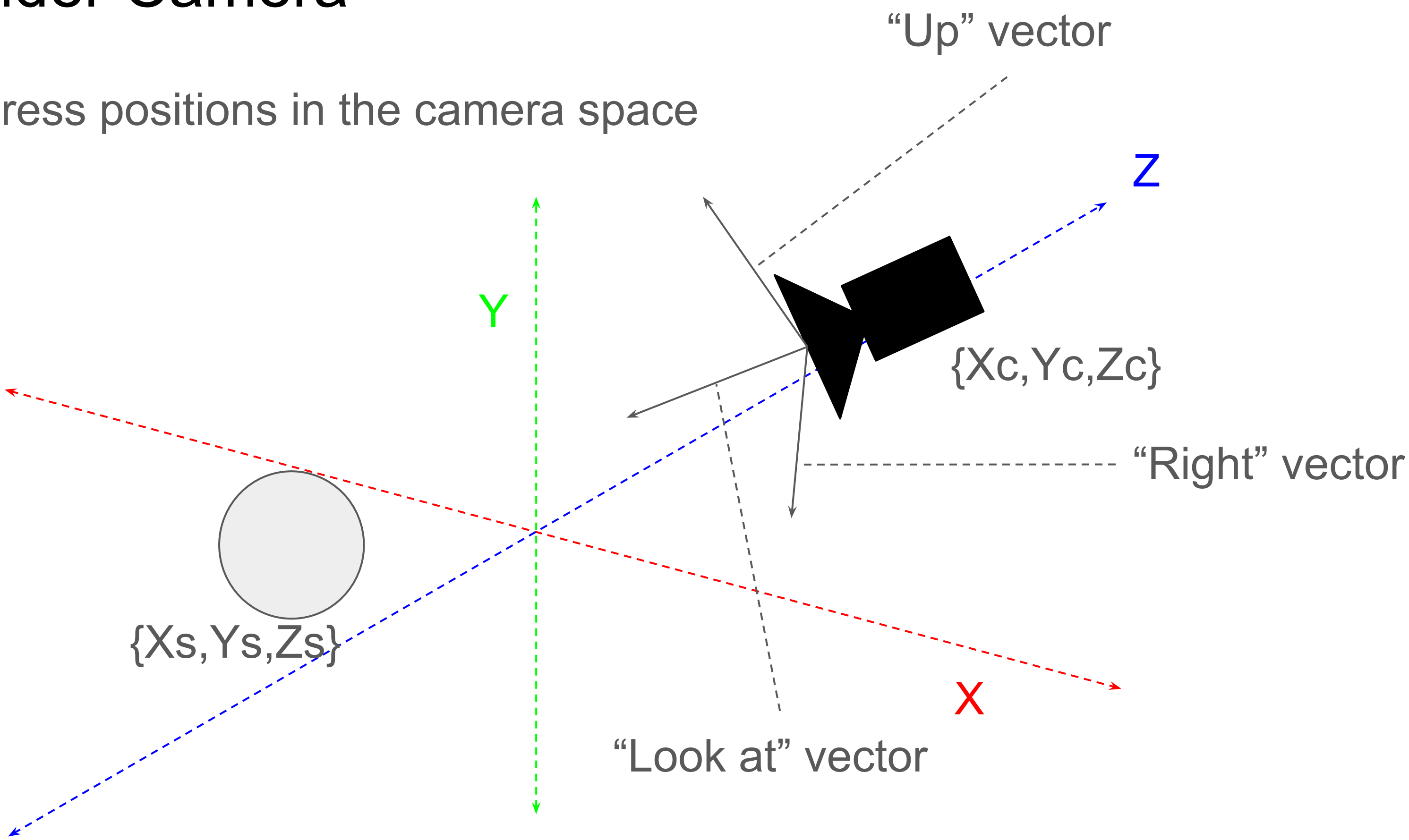
Project vertices in the frustum

Camera Space



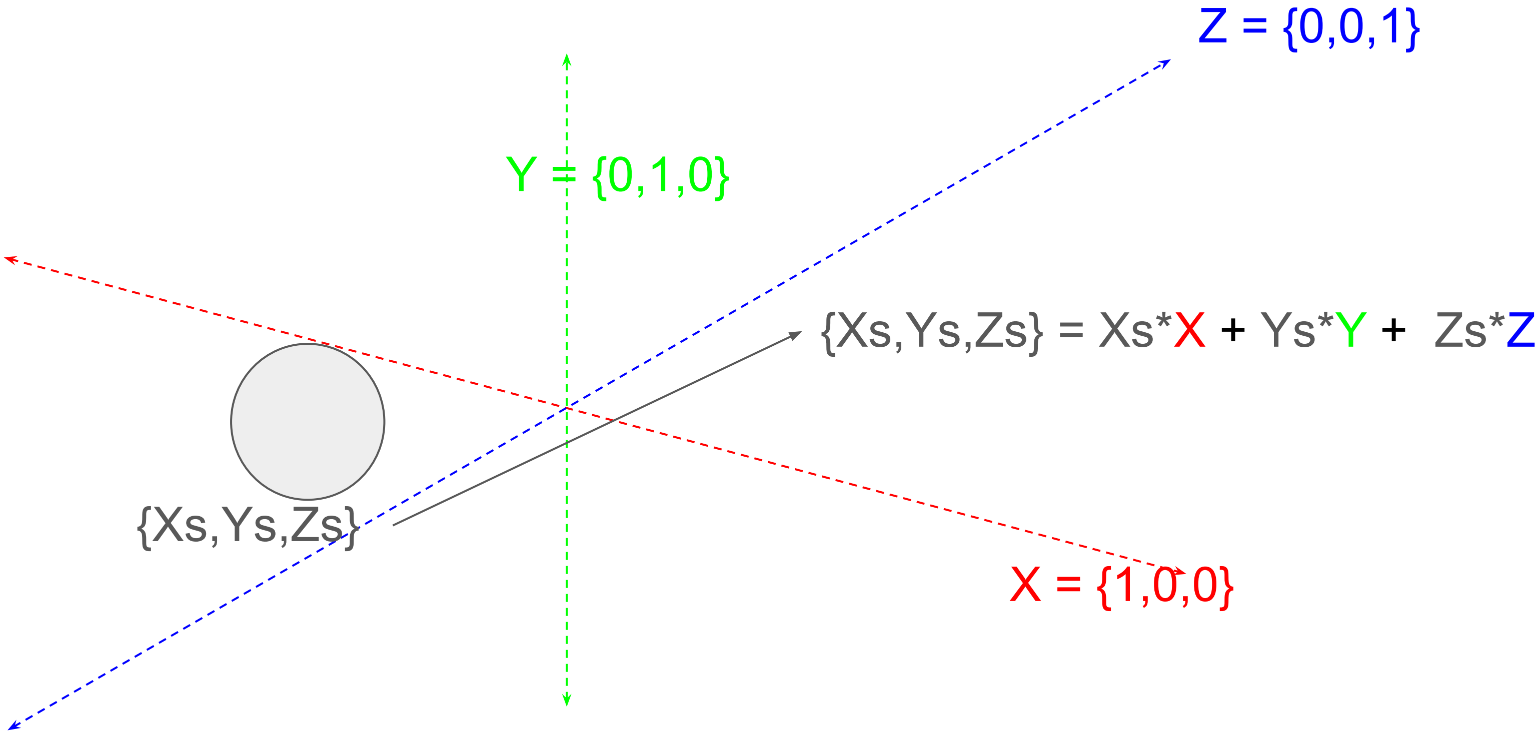
Reminder Camera

Re-express positions in the camera space



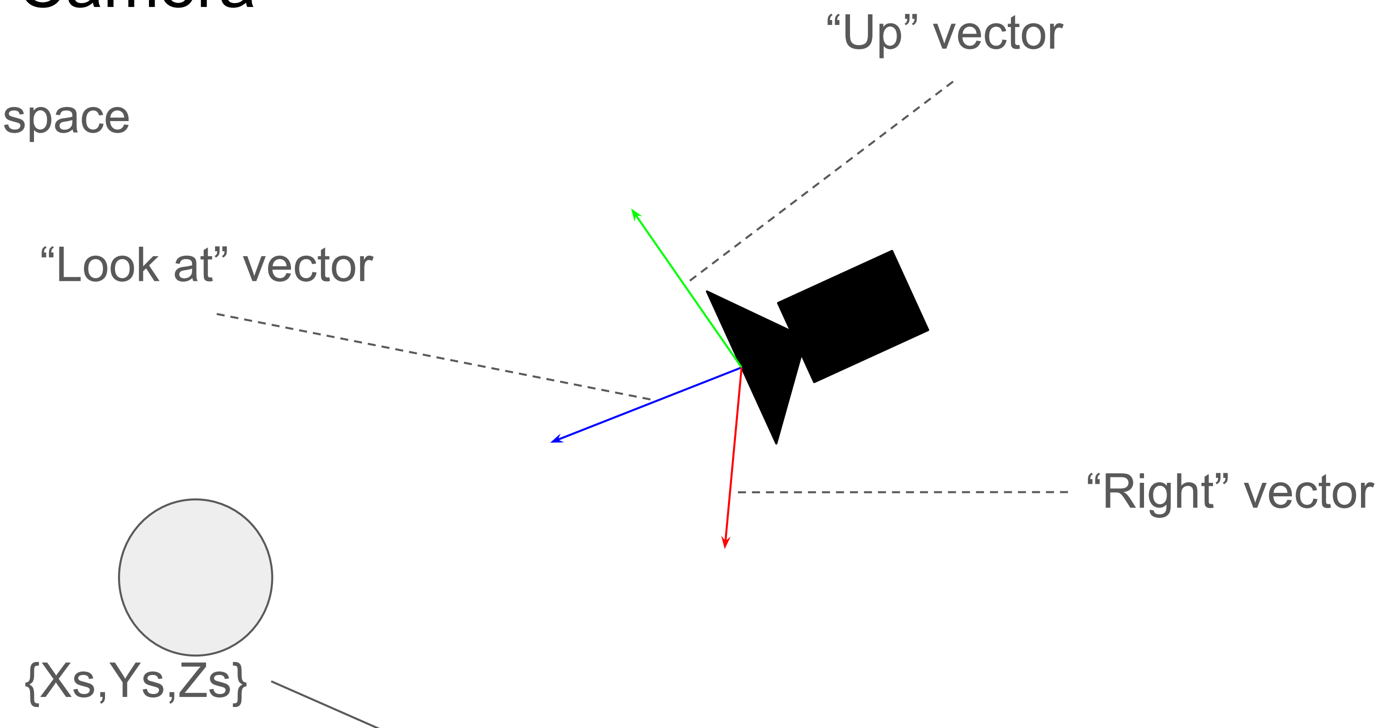
Reminder Camera

Our current coordinates :



Reminder Camera

The camera space



{Xs,Ys,Zs} = Xs' * R + Ys' * U + Zs' * L

Reminder Camera

The Mathematical problem

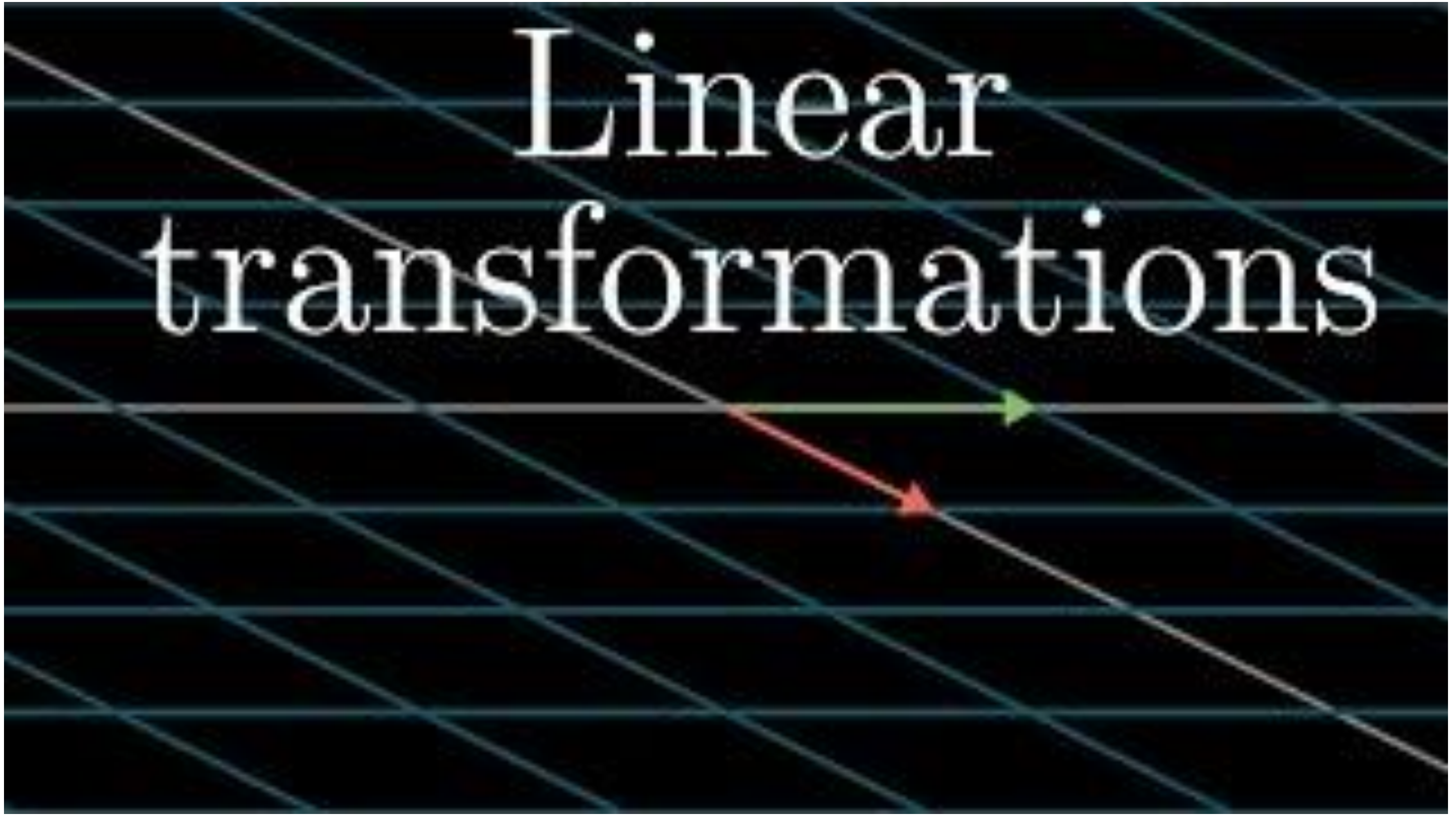
$\{Xs, Ys, Zs\} = Xs' * R + Ys' * U + Zs' * L$

$\{Xs, Ys, Zs\} = Xs * X + Ys * Y + Zs * Z$

$Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L$

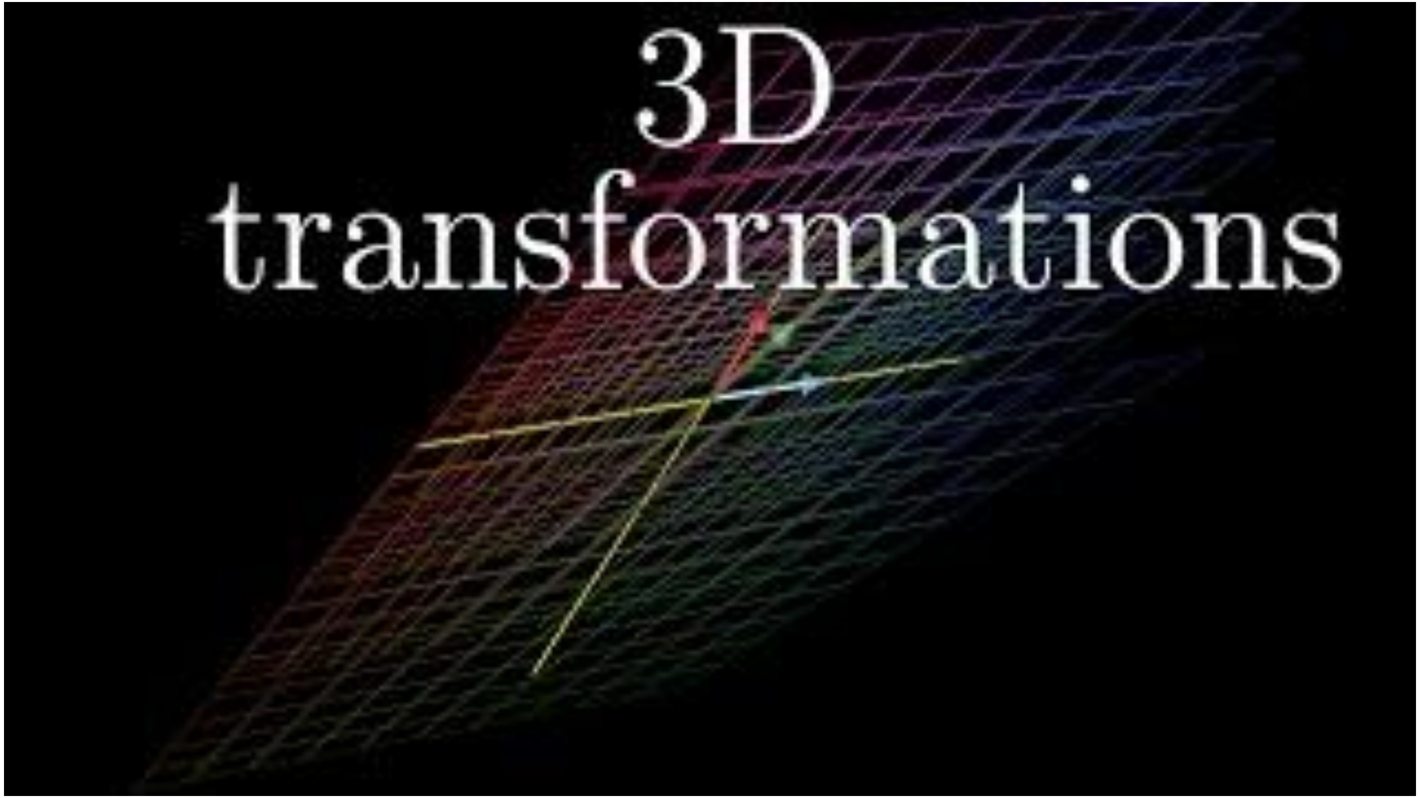
<u>The problem Find Xs’, Ys’, and Zs’</u>

Linear Transformations



https://youtu.be/kYB8IZa5AuE?si=vo2LXj1Vz_gwdasT

Linear Transformations



https://youtu.be/rHLEWRxRGiM?si=aKhXMsqAqGqWmSQv

Question :

$$Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L$$

<u>What is the matrix that solve this problem</u>

(1 minute alone)

(2 minutes with your neighbors)

(5 minutes with the whole group)

Question :

Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L

"Right" vector

"Look at" vector

$$\begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix} = \begin{bmatrix} Rx & Ux & Lx \\ Ry & Uy & Ly \\ Rz & Uz & Lz \end{bmatrix} \begin{bmatrix} Xs' \\ Ys' \\ Zs' \end{bmatrix}$$

"Up" vector

Question :

Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L

"Right" vector

"Look at" vector

$$\begin{bmatrix} Xs' \\ Ys' \\ Zs' \end{bmatrix} = \begin{bmatrix} Rx & Ux & Lx \\ Ry & Uy & Ly \\ Rz & Uz & Lz \end{bmatrix}^{-1} \begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix}$$

"Up" vector

Question :

Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L

"Right" vector

"Look at" vector

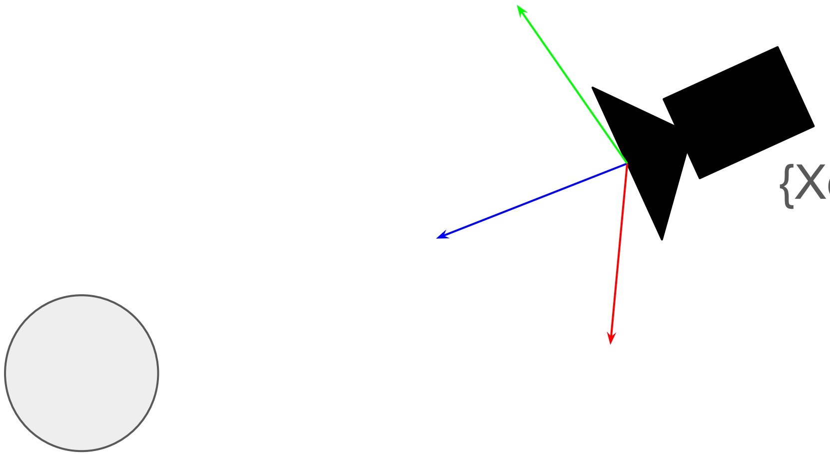
$$\begin{bmatrix} Xs' \\ Ys' \\ Zs' \end{bmatrix} = \begin{bmatrix} Rx & Ry & Rz \\ Ux & Uy & Uz \\ Lx & Ly & Lz \end{bmatrix} \begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix}$$

"Up" vector

Problem with Linear transformations

What about the camera coordinates ?

The camera space



{Xc,Yc,Zc}

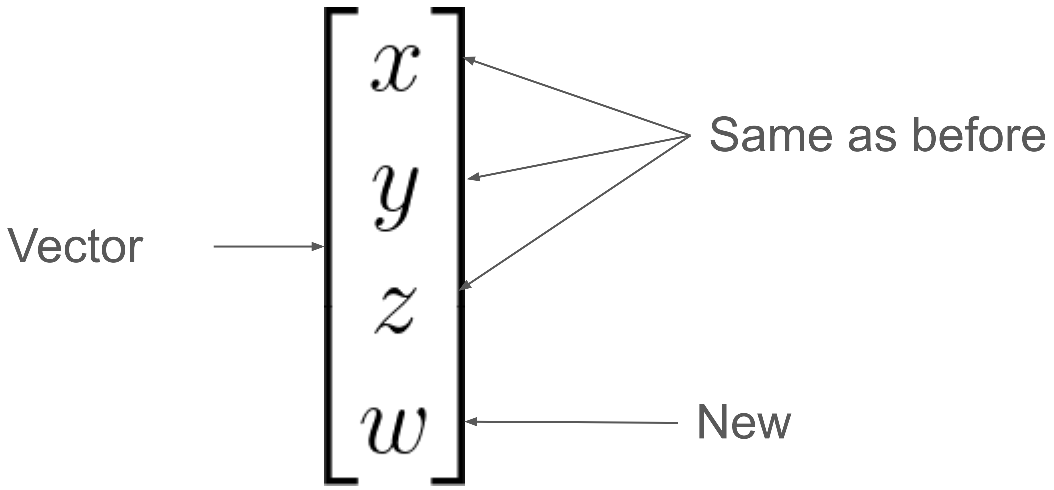
{Xs,Ys,Zs}

{Xs,Ys,Zs} = Xs’ * R + Ys’ * U + Zs’ * L

Homogeneous Coordinates

Allow us to “move” the origin of the frame

Using 4 coordinates instead of 3 : Homogeneous coordinates



Linear Transformation

Xs * X + Ys * Y + Zs * Z = Xs' * R + Ys' * U + Zs' * L

$$\begin{bmatrix} Xs' \\ Ys' \\ Zs' \\ Ws \end{bmatrix} = \begin{bmatrix} Rx & Ry & Rz & 0 \\ Ux & Uy & Uz & 0 \\ Lx & Ly & Lz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xs \\ Ys \\ Zs \\ Ws \end{bmatrix}$$

Affine Transformation

General case

$$\begin{bmatrix} x + wTx \\ y + wTy \\ z + wTz \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Affine Transformation

For a vertex: w = 1

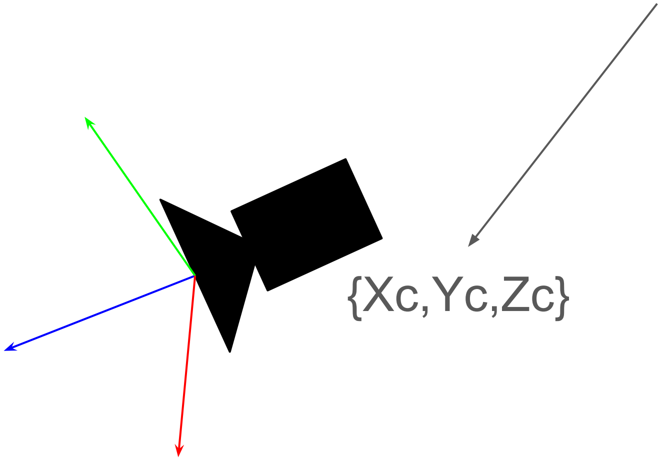
$$\begin{bmatrix} x + Tx \\ y + Ty \\ z + Tz \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Add the vector T to the vertex

Translate the camera to the origin

What about the camera coordinates ?

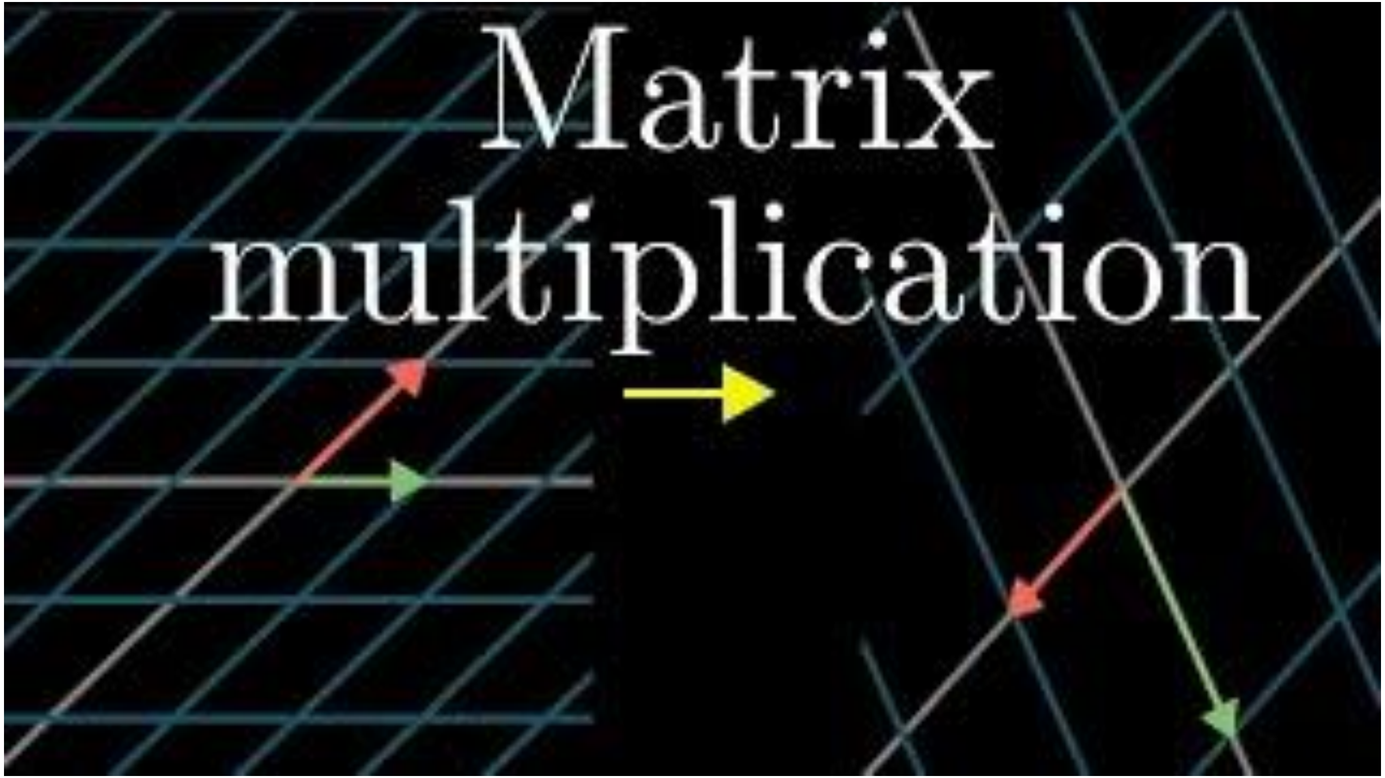
$$\begin{bmatrix} 1 & 0 & 0 & -Xc \\ 0 & 1 & 0 & -Yc \\ 0 & 0 & 1 & -Zc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Camera Space

$$\begin{bmatrix} Xs' \\ Ys' \\ Zs' \\ Ws \end{bmatrix} = \begin{bmatrix} Rx & Ry & Rz & 0 \\ Ux & Uy & Uz & 0 \\ Lx & Ly & Lz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -Xc \\ 0 & 1 & 0 & -Yc \\ 0 & 0 & 1 & -Zc \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xs \\ Ys \\ Zs \\ Ws \end{bmatrix}$$

Matrix multiplication



https://www.youtube.com/watch?v=XkY2DOUCWMU

Camera Space

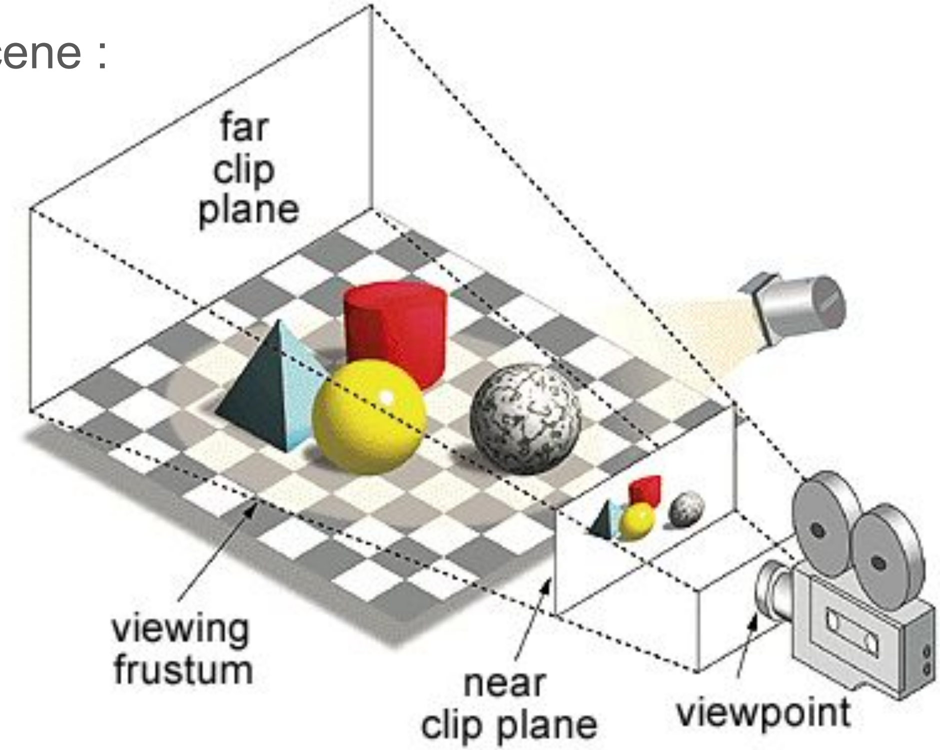
View
Matrix

$$\begin{bmatrix} Rx & Ry & Rz & 0 \\ Ux & Uy & Uz & 0 \\ Lx & Ly & Lz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -Xc \\ 0 & 1 & 0 & -Yc \\ 0 & 0 & 1 & -Zc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reminder: Camera

Frustum : the visible part of the scene :

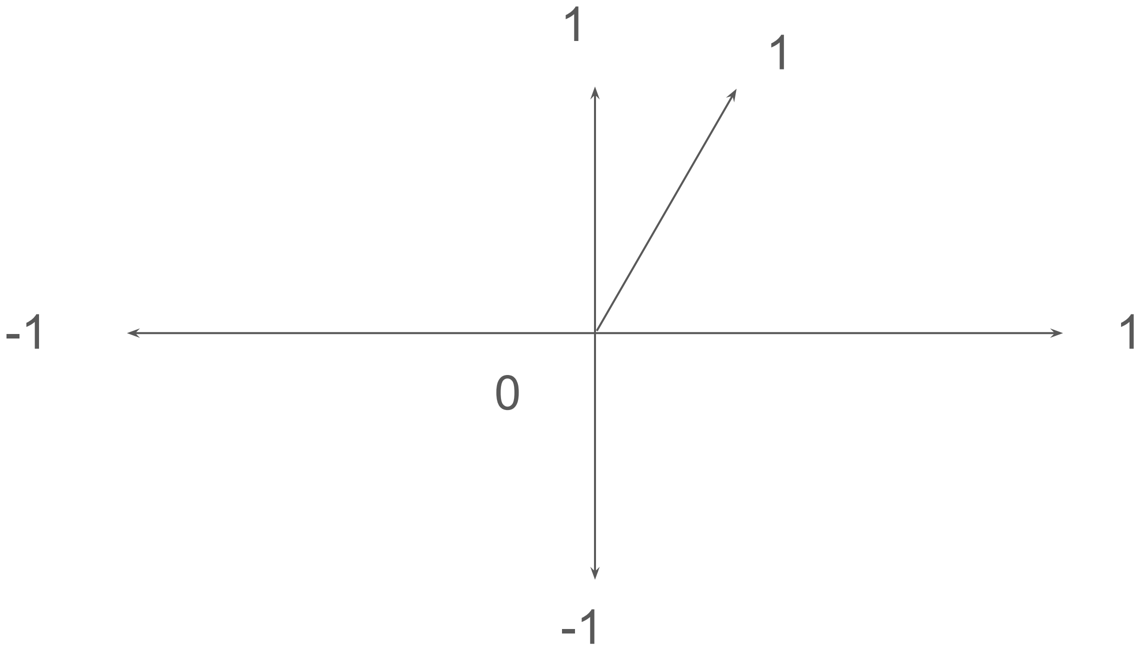
- Near plane
- Far plane
- Aspect ratio
- Field of View



Projection Perspective

Objective : we want to express the visible space in the following space

x in [-1,1]
y in [-1,1]
z in [0,1]



Perspective Projection matrix

- Near plane = n
- Far plane = f
- Aspect ratio = a
- Field of View = fov

$$s = \frac{1}{tan(fov/2)}$$

$$\begin{bmatrix} s/a & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & -\frac{f*n}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Problem

After the projection the w coordinate of the vertex is modified.

To solve this issue we normalize all vertex coordinates by dividing them by w

Summary Vertex shader

For each vertex v, the vertex shader compute a vertex v’ such that :

$$v_{tmp} = [proj]\,[view]\,v$$

$$v' = \frac{v_{tmp}}{v_{tmp}.w}$$