CSE4204

LAB-3 : Transformation Matrices, Perspective projection and camera transformation

Transformation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\boldsymbol{V' = S \times V}$$

```
var vertexShaderSource =
    `attribute vec3 a_coords;
     attribute vec3 a_colors;
     uniform mat4 u_Scale;
     varying vec3 v_color;

     void main() {
         gl_Position = u_Scale*vec4(a_coords, 1.0);
         v_color = a_colors;
     }`;
```

Scale Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

```
u_scale_location = gl.getUniformLocation(prog, "u_Scale");
var Sx = 1.5;
var Sy = 0.75;
var Sz = 1.0;
var scaleMatrix = new Float32Array( [Sx,   0.0,  0.0,  0.0,
                                     0.0,  Sy,   0.0,  0.0,
                                     0.0,  0.0,  Sz,   0.0,
                                     0.0,  0.0,  0.0,  1.0] );

gl.uniformMatrix4fv(u_scale_location, false, scaleMatrix);
```

Column Major

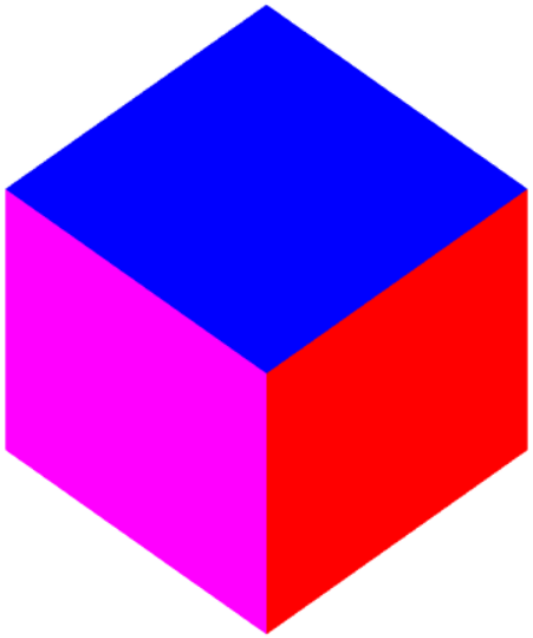
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

```
u_scale_location = gl.getUniformLocation(prog, "u_Scale");
var Sx = 1.5;
var Sy = 0.75;
var Sz = 1.0;
var scaleMatrix = new Float32Array( [Sx,   0.0,   0.0,   0.0,
                                     0.0,  Sy,    0.0,   0.0,
                                     0.0,  0.0,   Sz,    0.0,
                                     0.0,  0.0,   0.0,   1.0] );

gl.uniformMatrix4fv(u_scale_location, false, scaleMatrix);
```

Get the code

rb.gy/seyoi

3D Cube

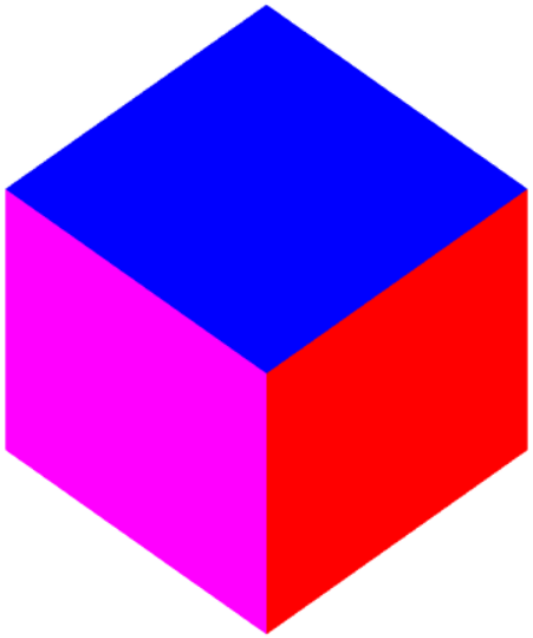


```
var indices = new Uint8Array([
        0, 1, 2,      0, 2, 3,    // Front face
        4, 5, 6,      4, 6, 7,    // Back face
        8, 9, 10,     8, 10, 11,  // Top face
        12, 13, 14,   12, 14, 15, // Bottom face
        16, 17, 18,   16, 18, 19, // Right face
        20, 21, 22,   20, 22, 23  // Left face
    ]);
```

```
var coords = new Float32Array( [
                // Front face
               -0.5, -0.5,  0.5,
                0.5, -0.5,  0.5,
                0.5,  0.5,  0.5,
               -0.5,  0.5,  0.5,

               // Back face
               -0.5, -0.5, -0.5,
               -0.5,  0.5, -0.5,
                0.5,  0.5, -0.5,
                0.5, -0.5, -0.5,

               // Top face
               -0.5,  0.5, -0.5,
               -0.5,  0.5,  0.5,
                0.5,  0.5,  0.5,
                0.5,  0.5, -0.5,

               // Bottom face
               -0.5, -0.5, -0.5,
                0.5, -0.5, -0.5,
                0.5, -0.5,  0.5,
               -0.5, -0.5,  0.5,

               // Right face
                0.5, -0.5, -0.5,
                0.5,  0.5, -0.5,
                0.5,  0.5,  0.5,
                0.5, -0.5,  0.5,

               // Left face
               -0.5, -0.5, -0.5,
               -0.5, -0.5,  0.5,
               -0.5,  0.5,  0.5,
               -0.5,  0.5, -0.5
                                    ] );
```

```
var colors = new Float32Array( [
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,

        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,

        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,

        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,

        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,

        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0
        ] );
```

Depth Test + Face Culling



```
gl.clearColor(1.0, 1.0, 1.0, 1.0);
gl.enable(gl.DEPTH_TEST);
gl.enable(gl.CULL_FACE);
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
gl.drawElements(gl.TRIANGLES, 3*12, gl.UNSIGNED_BYTE, 0);
```

Rotation in 3D

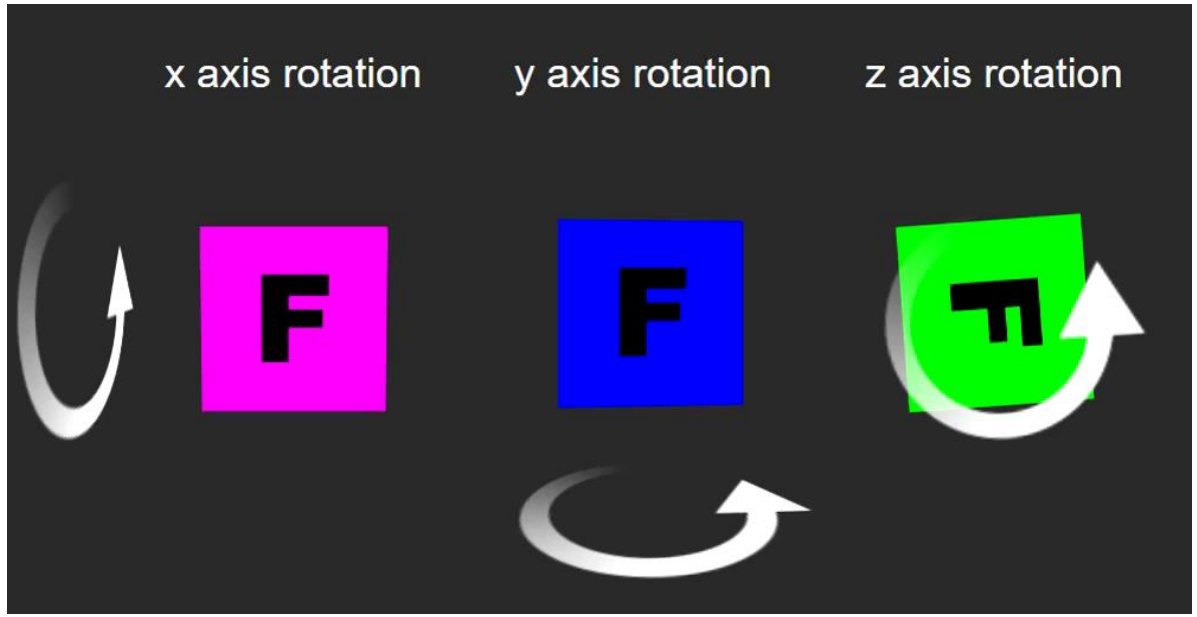
CCW → +ve rotation

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{V'} = \boldsymbol{R} \times \boldsymbol{V}$$



Rotation in 3D

```
var vertexShaderSource =
    `attribute vec3 a_coords;
     attribute vec3 a_colors;
     uniform mat4 u_RotY;
     varying vec3 v_color;

     void main() {
         gl_Position = u_RotY*vec4(a_coords, 1.0);
         v_color = a_colors;
     }`;
```

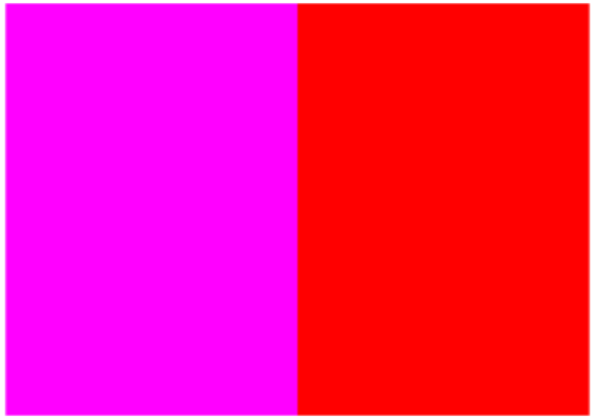
$$V' = R_y \times V$$

Rotation in 3D

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");

var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [
        Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
        0.0,            1.0,   0.0,            0.0,
        Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
        0.0,            0.0,   0.0,            1.0 ] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);
```

Rotation in 3D



```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");

var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [
        Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
        0.0,            1.0,   0.0,            0.0,
        Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
        0.0,            0.0,   0.0,            1.0 ] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);
```

Get the code

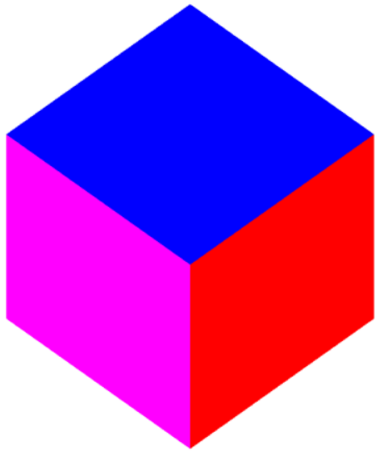
rb.gy/bqtxa

Composite Transformation

$$V' = R_x \times R_y \times V$$

```
var vertexShaderSource =
    `attribute vec3 a_coords;
     attribute vec3 a_colors;
     uniform mat4 u_RotY;
     uniform mat4 u_RotX;
     varying vec3 v_color;

     void main() {
         gl_Position = u_RotX*u_RotY*vec4(a_coords, 1.0);
         v_color = a_colors;
     }`;
```

Composite Transformation

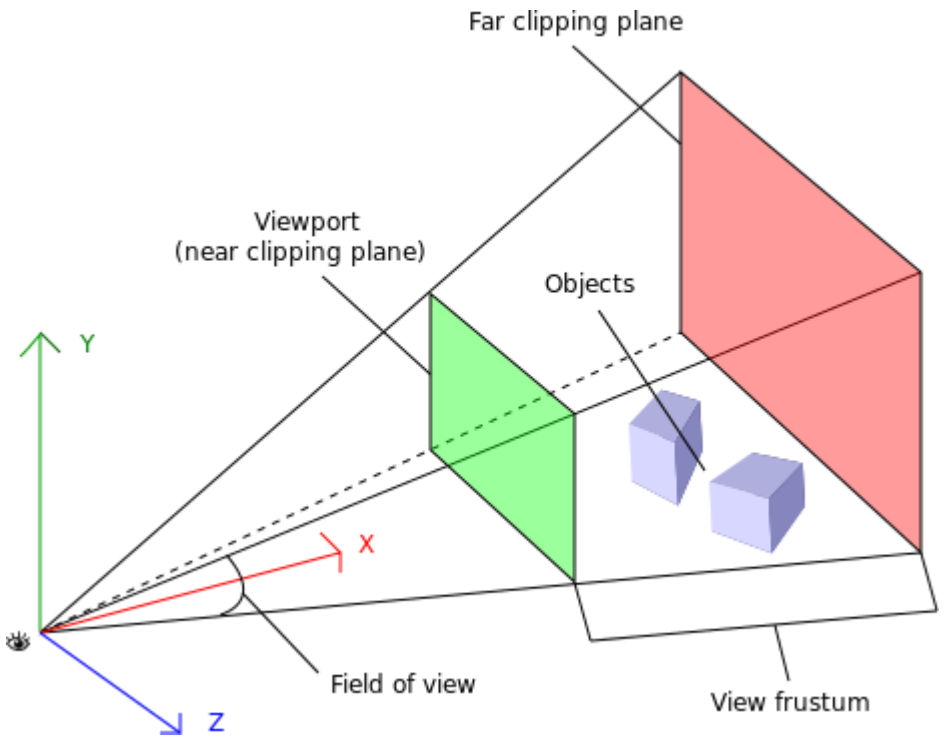


```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");
var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
                                        0.0,            1.0,   0.0,            0.0,
                                        Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
                                        0.0,            0.0,   0.0,            1.0] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);

var u_rotateX_location = gl.getUniformLocation(prog, "u_RotX");
var thetaX = 45;
var rad = thetaX*Math.PI/180;
var rotateXMatrix = new Float32Array( [1.0,  0.0,            0.0,            0.0,
                                        0.0,  Math.cos(rad),  Math.sin(rad),  0.0,
                                        0.0, -Math.sin(rad),  Math.cos(rad),  0.0,
                                        0.0,  0.0,            0.0,            1.0] );
gl.uniformMatrix4fv(u_rotateX_location, false, rotateXMatrix);
```

Get the code

rb.gy/nwomo

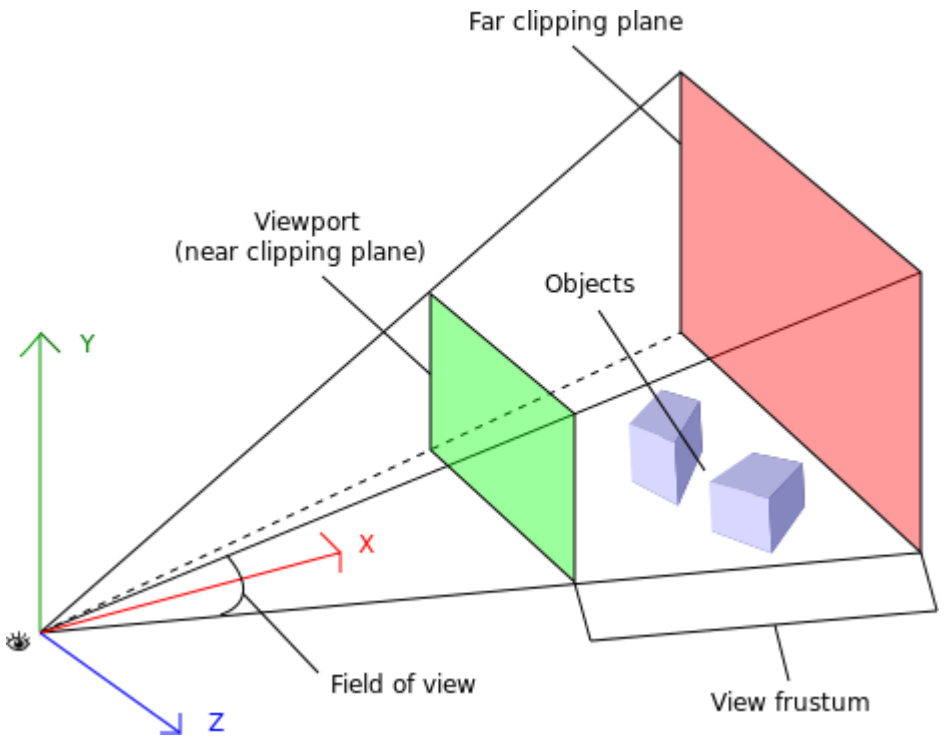
Perspective Projection



https://www.oreilly.com/library/view/webgl-up-and/9781449326487/ch01.html

Perspective Projection

$$\text{persMat} = \begin{bmatrix} \frac{1}{aspect * \tan(\frac{fov}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\frac{fov}{2})} & 0 & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & -\frac{2 * far * near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



https://www.oreilly.com/library/view/webgl-up-and/9781449326487/ch01.html

Perspective Projection

```
`attribute vec3 a_coords;
 attribute vec3 a_colors;
 uniform mat4 u_RotY;
 uniform mat4 u_RotX;
 uniform mat4 u_Scale;
 uniform mat4 u_Trans;
 uniform mat4 u_Pers;
 varying vec3 v_color;

 void main() {

     mat4 M = u_Trans*u_RotX*u_RotY*u_Scale;
     gl_Position = u_Pers*M*vec4(a_coords, 1.0);
     v_color = a_colors;
 }`;
```

Perspective Projection

```
u_matrix_pers_location = gl.getUniformLocation(prog, "u_Pers");
```

```
var aspect = 1.0;
var fov = 45.0;
var far = 5.0;
var near = 2.0;

var pa = 1.0/(aspect*Math.tan((fov/2)*Math.PI/180));
var pb = 1.0/(Math.tan((fov/2)*Math.PI/180));
var pc = -(far+near) / (far-near);
var pd = -(2.0*far*near) / (far-near);

var persMat = new Float32Array( [pa,   0.0,  0.0,  0.0,
                                 0.0,  pb,   0,    0.0,
                                 0.0,  0.0,  pc,  -1.0,
                                 0.0,  0.0,  pd,   0.0] );
```

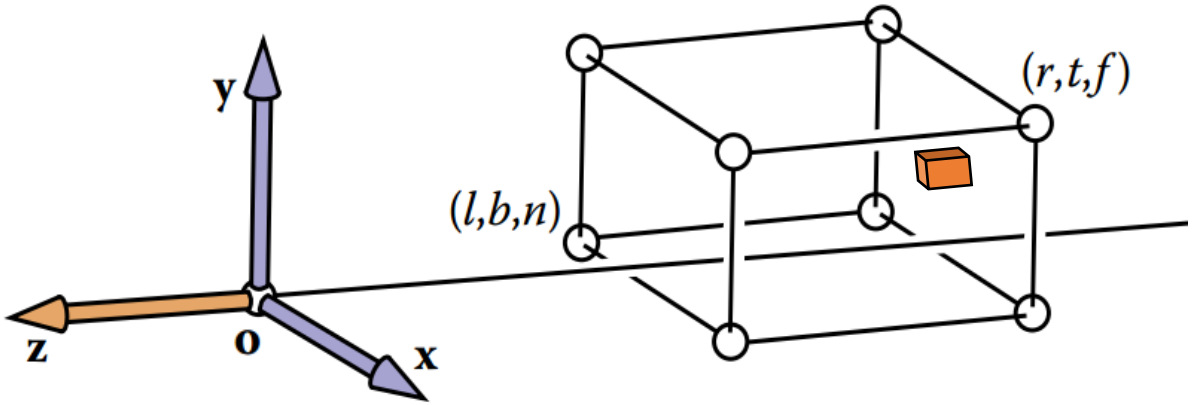
```
gl.uniformMatrix4fv(u_matrix_pers_location, false, persMat);
```

- Get the code

rb.gy/l2r1n

Camera

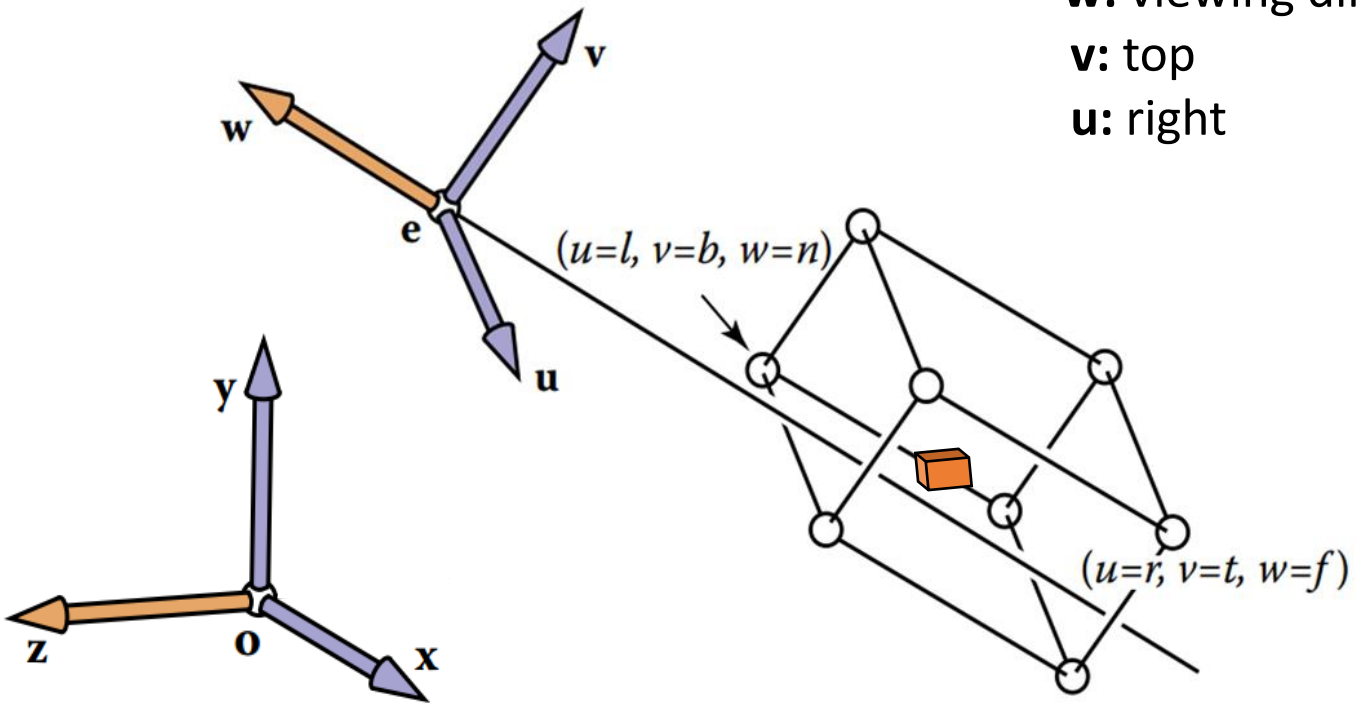
- We'd like to be able to change the viewpoint in 3D and look in any direction.



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Camera

e: camera/eye position
- **w:** viewing direction
v: top
u: right

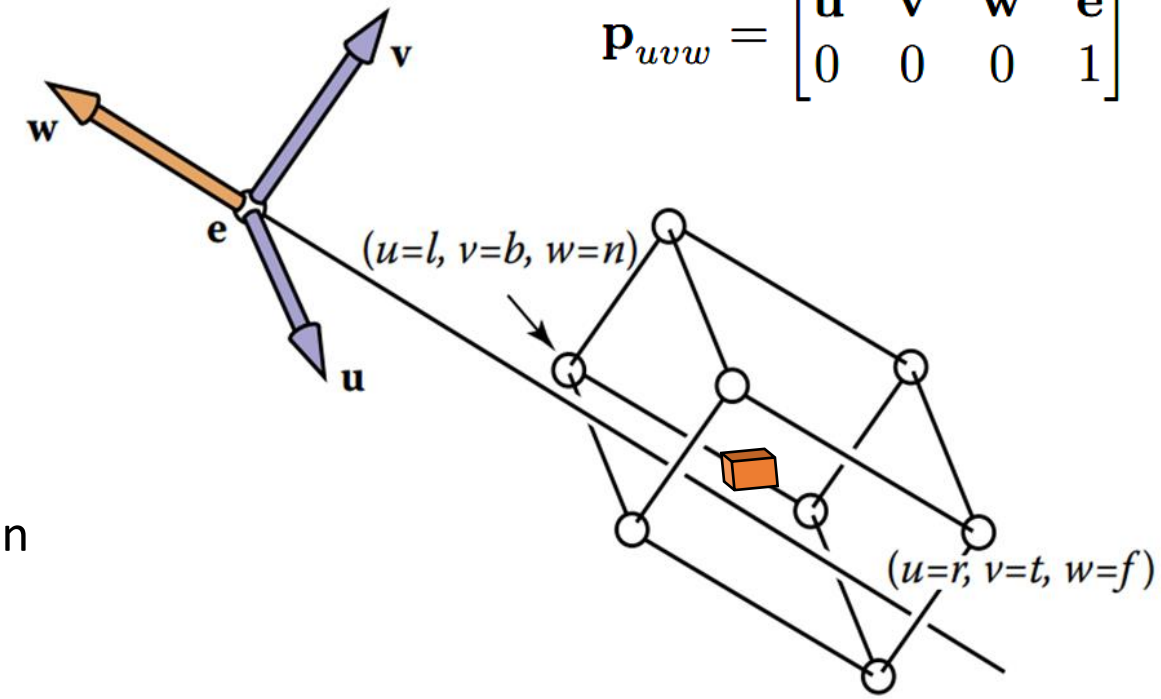


Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Camera

$$\begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$\mathbf{p}_{uvw} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}_{xyz}.$$



e: camera/eye position
- **w:** viewing direction
v: top
u: right

Camera

```
var vertexShaderSource =

        `attribute vec3 a_coords;
         attribute vec3 a_colors;
         uniform mat4 u_RotY;
         uniform mat4 u_RotX;
         uniform mat4 u_Scale;
         uniform mat4 u_Trans;
         uniform mat4 u_Basis;
         uniform mat4 u_Eye;
         uniform mat4 u_Pers;
         varying vec3 v_color;

         void main() {

             mat4 M = u_Trans*u_RotX*u_RotY*u_Scale;
             mat4 V = u_Basis*u_Eye;
             mat4 P = u_Pers;
             mat4 MVP = P*V*M;
             gl_Position = MVP*vec4(a_coords, 1.0);
             v_color = a_colors;
         }`;
```

Camera

```
u_matrix_basis_location = gl.getUniformLocation(prog, "u_Basis");
u_matrix_eye_location = gl.getUniformLocation(prog, "u_Eye");
```

```
var basisMat = new Float32Array([ 1, 0, 0, 0,
                                  0, 1, 0, 0,
                                  0, 0, 1, 0,
                                  0, 0, 0, 1]);

var xe = 0.5;
var ye = 1.0;
var ze = 3.0;
var eyeMat = new Float32Array([1,   0,   0,   0,
                               0,   1,   0,   0,
                               0,   0,   1,   0,
                             -xe, -ye, -ze,   1]);
```

```
gl.uniformMatrix4fv(u_matrix_basis_location, false, basisMat);
gl.uniformMatrix4fv(u_matrix_eye_location, false, eyeMat);
```

- Get the code

rb.gy/a6l3d