Transformations Part 2

CS GY-6533 / UY-4533

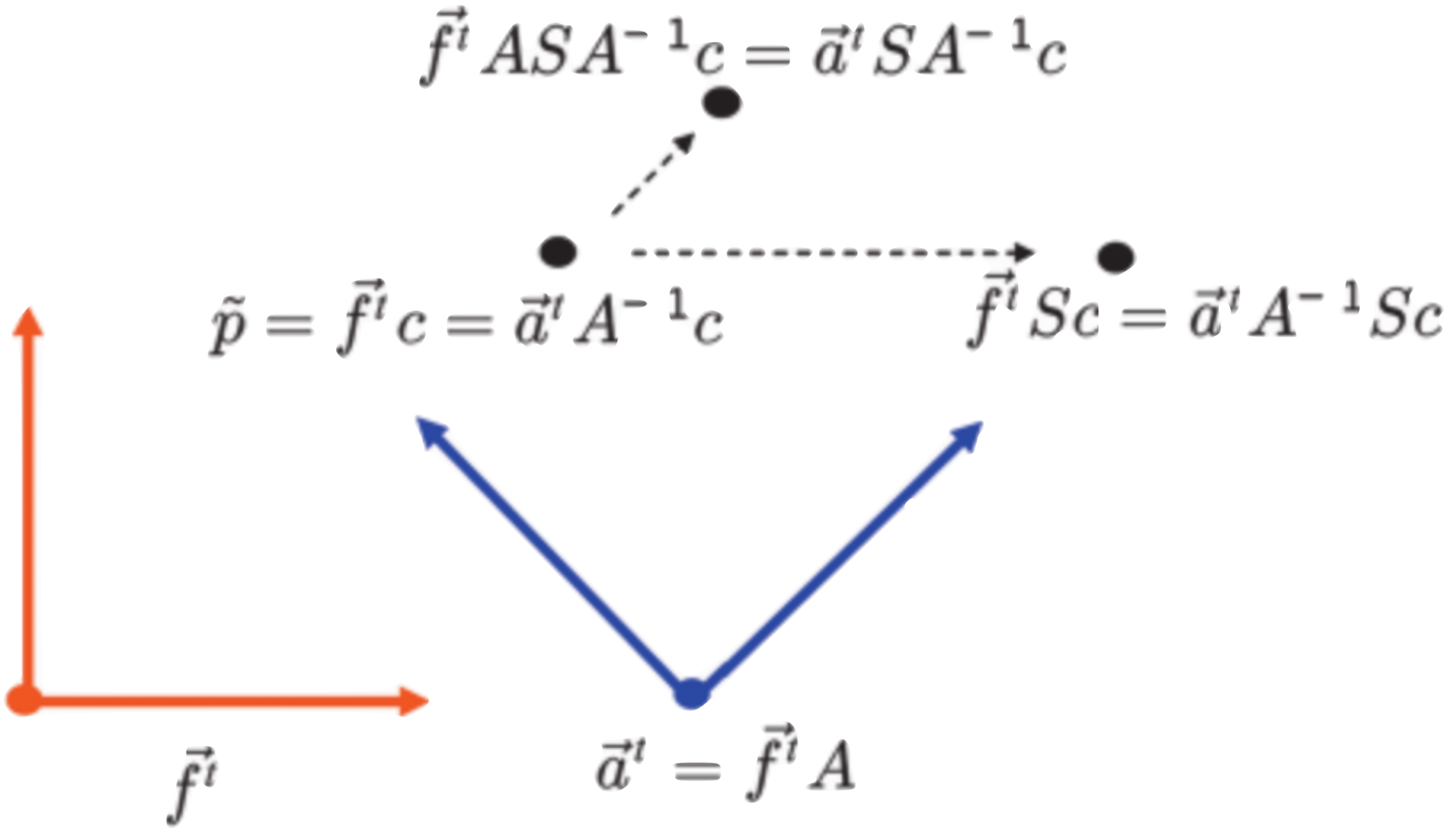
Matrix Inverse

The **inverse** of a matrix **M** is the unique matrix $\mathbf{M}^{-1}$ with the property $\mathbf{M}^{-1}\mathbf{M} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$ where **I** is the **identity matrix**.

$$\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M$$

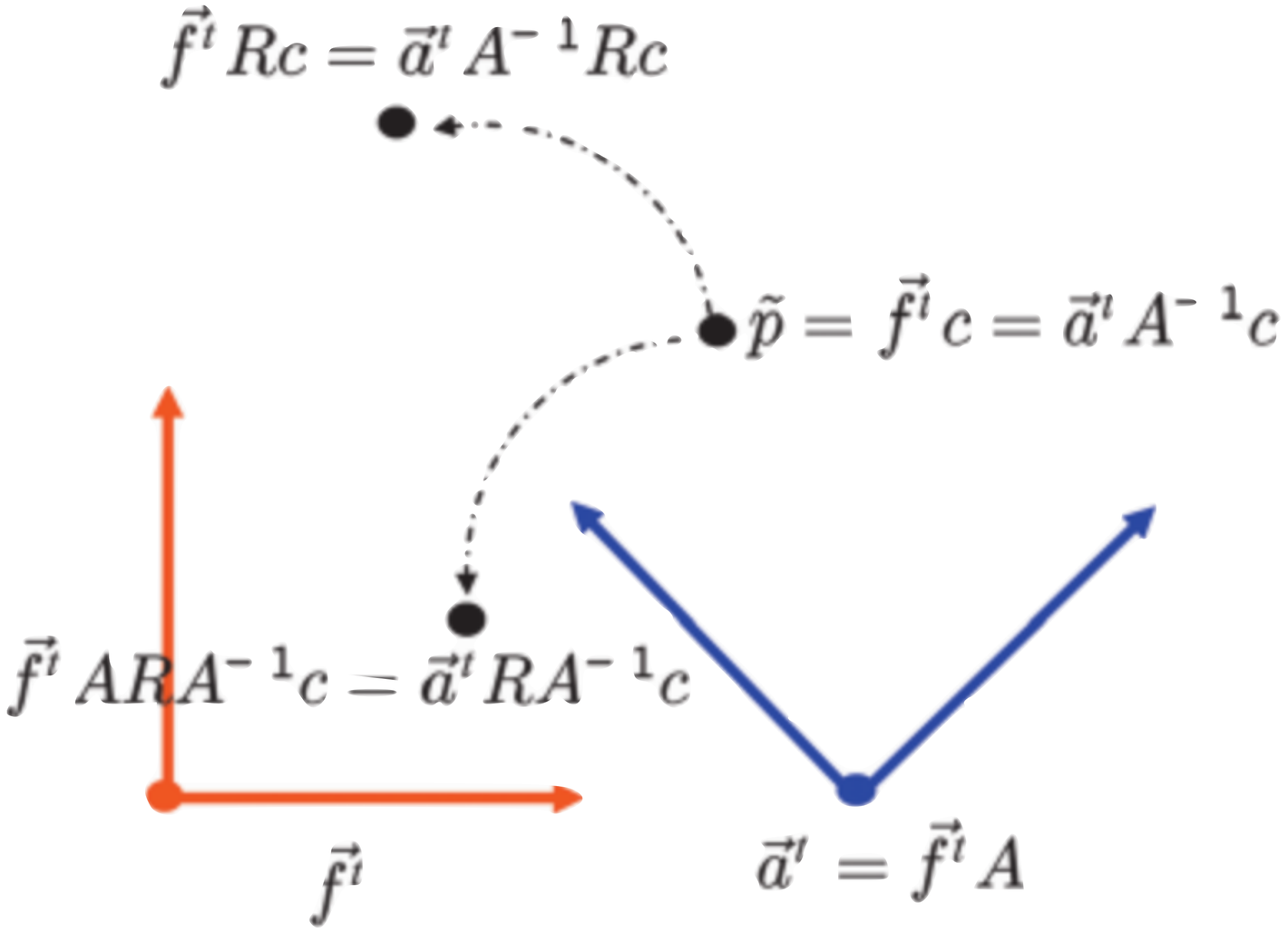
$$\vec{\mathbf{a}}^t M^{-1} = \vec{\mathbf{b}}^t$$

Frame respect and left-of rule.



$$\tilde{p} = \vec{\mathbf{f}}^t \mathbf{c} \Rightarrow \vec{\mathbf{f}}^t S\mathbf{c}$$

$$\tilde{p} = \vec{\mathbf{a}}^t A^{-1}\mathbf{c} \Rightarrow \vec{\mathbf{a}}^t SA^{-1}\mathbf{c}$$


$\vec{f}^{\,t} Rc = \vec{a}^{\,t} A^{-1} Rc$
$\tilde{p} = \vec{f}^{\,t} c = \vec{a}^{\,t} A^{-1} c$
$\vec{f}^{\,t} ARA^{-1} c = \vec{a}^{\,t} RA^{-1} c$
$\vec{f}^{\,t}$
$\vec{a}^{\,t} = \vec{f}^{\,t} A$

$\vec{\mathbf{f}}^t$ is transformed by S in respect to $\vec{\mathbf{f}}^t$

$$\vec{\mathbf{f}}^t \Rightarrow \vec{\mathbf{f}}^t S$$

$\vec{\mathbf{f}}^t$ is transformed by S in respect to $\vec{\mathbf{a}}^t$

$$\vec{\mathbf{f}}^t = \vec{\mathbf{a}}^t A^{-1} \Rightarrow \vec{\mathbf{a}}^t S A^{-1}$$

Auxiliary frame.

$$\vec{\mathbf{a}}^t = \vec{\mathbf{f}}^t A$$

$$\vec{\mathbf{f}}^t$$

$$= \vec{\mathbf{a}}^t A^{-1}$$

$$\Rightarrow \vec{\mathbf{a}}^t M A^{-1}$$

$$= \vec{\mathbf{f}}^t A M A^{-1}.$$

Multiple transformations.

$$\vec{\mathbf{f}}^t \Rightarrow \vec{\mathbf{f}}^t T R.$$

$$\vec{\mathbf{f}}^t \Rightarrow \vec{\mathbf{f}}^t T = \vec{\mathbf{f}}'^t.$$

$$\vec{\mathbf{f}}^t T \Rightarrow \vec{\mathbf{f}}^t T R,$$

$$\vec{\mathbf{f}}'^t \Rightarrow \vec{\mathbf{f}}'^t R.$$

Frames in Computer Graphics

World, Object and Eye frames

$\vec{\mathrm{w}}^{\,t}$ - world frame

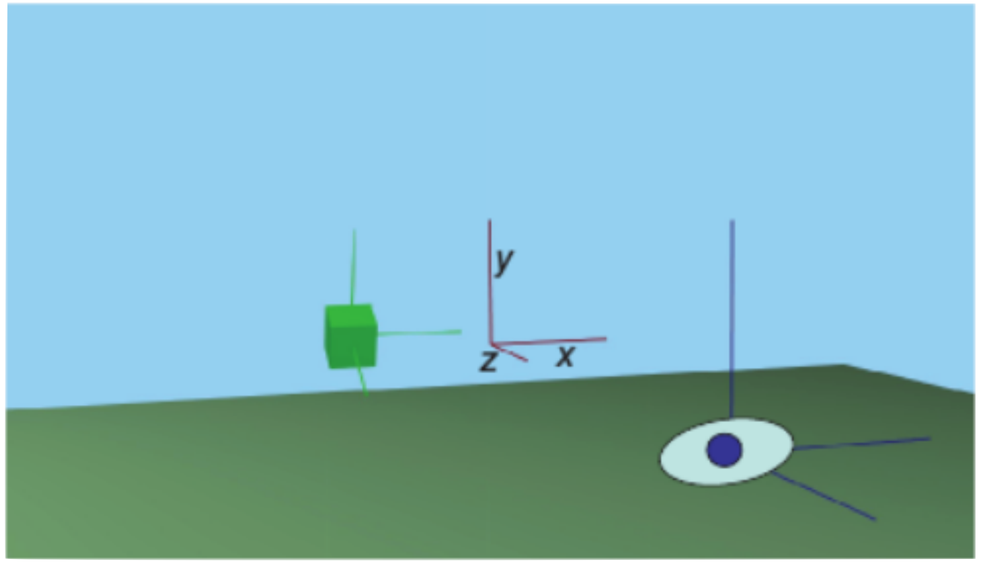
$\vec{\mathrm{o}}^{\,t}$ - object frame

$$\vec{\mathbf{o}}^{t} = \vec{\mathbf{w}}^{t} O .$$

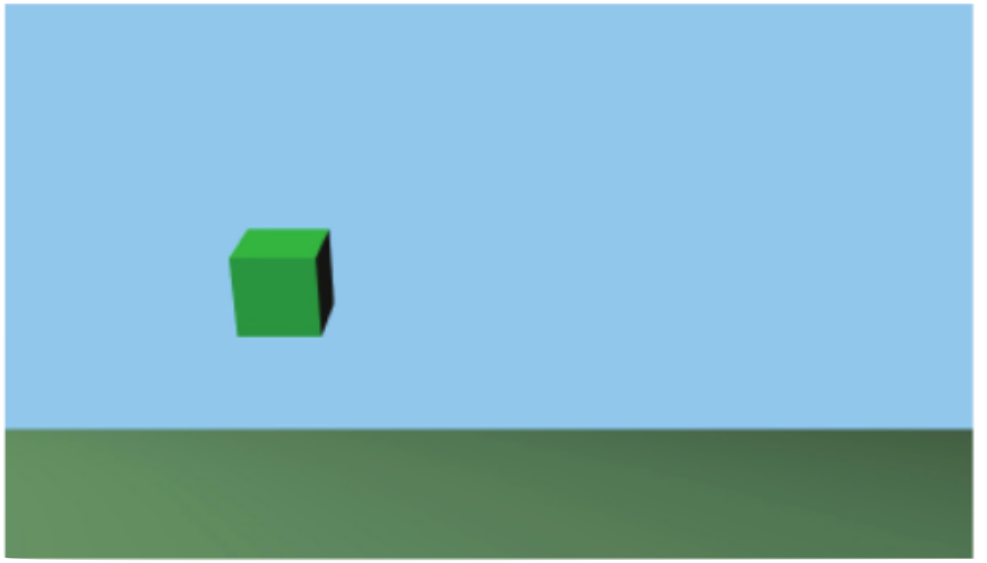
$\vec{e}^{\,t}$ - eye frame

$$\vec{\mathbf{e}}^{t} = \vec{\mathbf{w}}^{t} E.$$

$$\tilde{p} = \vec{\mathbf{o}}^t \mathbf{c} = \vec{\mathbf{w}}^t O \mathbf{c} = \vec{\mathbf{e}}^t E^{-1} O \mathbf{c}$$



(a) The frames



(b) The eye's view

Using transformations in our code.

2D example.

Modelview matrix.

$$\tilde{p} = \vec{\mathbf{o}}^t \mathbf{c} = \vec{\mathbf{w}}^t O \mathbf{c} = \vec{\mathbf{e}}^t E^{-1} O \mathbf{c}$$

vertex.glsl

```
attribute vec4 position;

uniform mat4 modelViewMatrix;

void main() {
 gl_Position = modelViewMatrix * position;
}
```

fragment.glsl

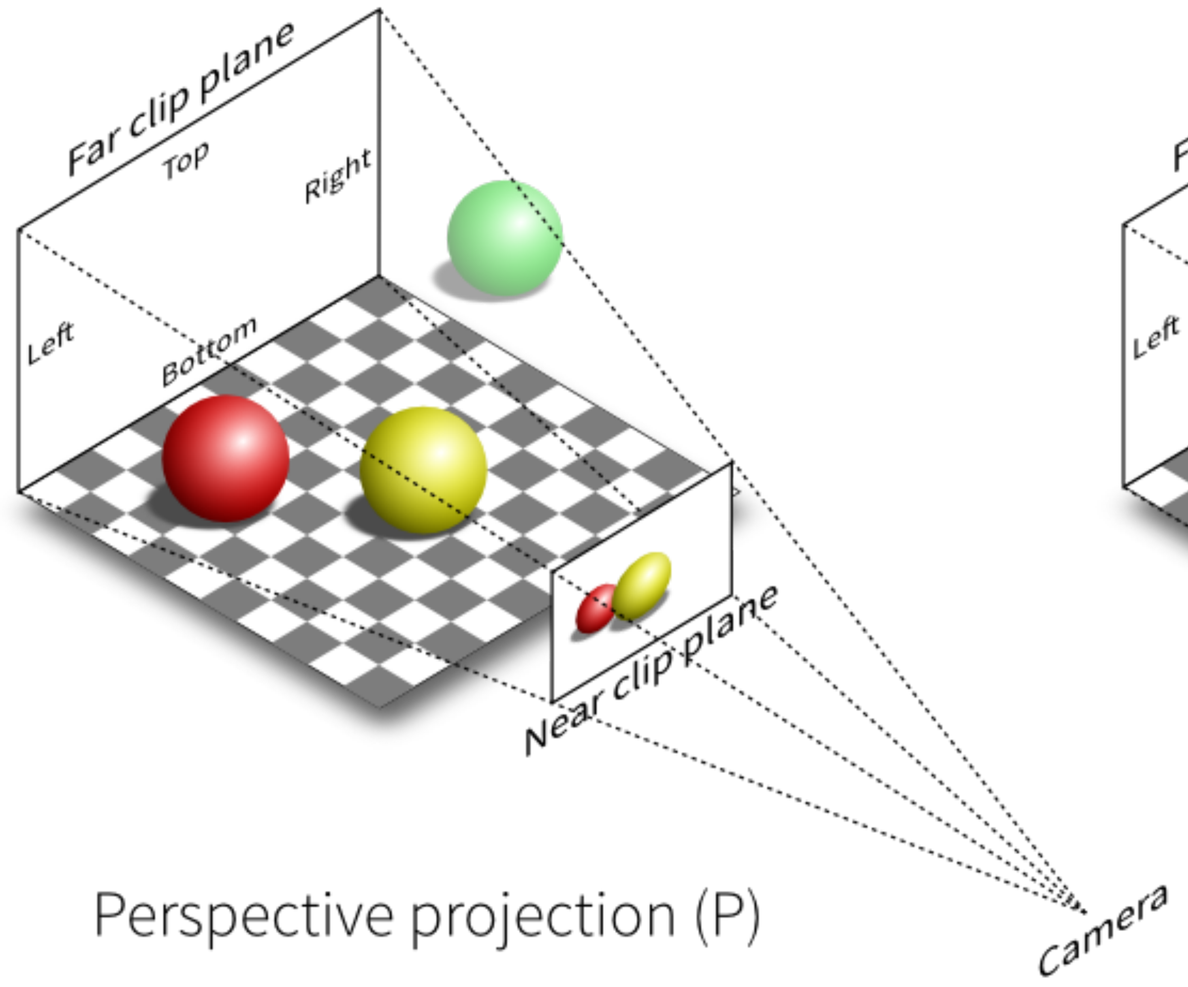
```
void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}
```

```
#include "matrix4.h"

    GLuint modelviewMatrixUniformLocation;

// init

    modelviewMatrixUniformLocation = glGetUniformLocation(program, "modelViewMatrix");



// render


    Matrix4 objectMatrix;
    objectMatrix = objectMatrix.makeZRotation(45.0);

    Matrix4 eyeMatrix;
    eyeMatrix = eyeMatrix.makeTranslation(Cvec3(-0.5, 0.0, 0.0));
    Matrix4 modelViewMatrix = inv(eyeMatrix) * objectMatrix;

    GLfloat glmatrix[16];
    modelViewMatrix.writeToColumnMajorMatrix(glmatrix);
    glUniformMatrix4fv(modelviewMatrixUniformLocation, 1, false, glmatrix);
```

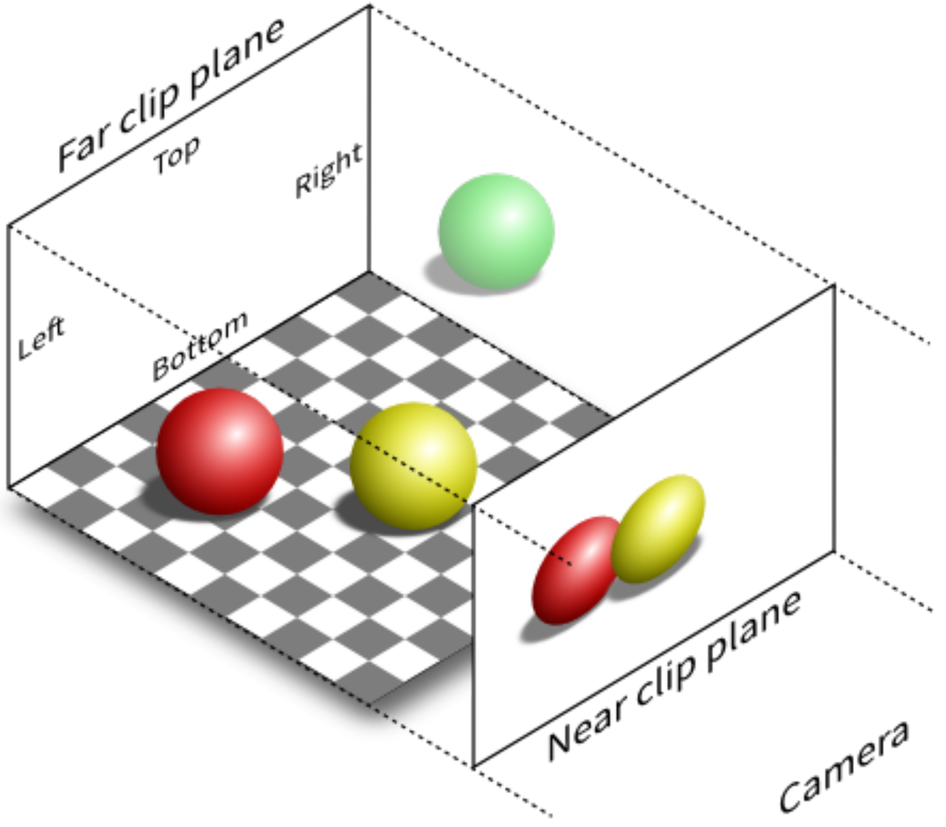
Moving into 3D.

```
// add GLUT_DEPTH to glutInitDisplayMode
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);


void init() {
    glCullFace(GL_BACK);
    glEnable(GL_CULL_FACE);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);
    glReadBuffer(GL_BACK);
}
```

Projection



Perspective projection (P)



Orthographic projection (O)

vertex.glsl

```
attribute vec4 position;
attribute vec4 color;

uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;

varying vec4 varyingColor;

void main() {
  varyingColor = color;
  gl_Position = projectionMatrix * modelViewMatrix * position;
}
```

fragment.glsl

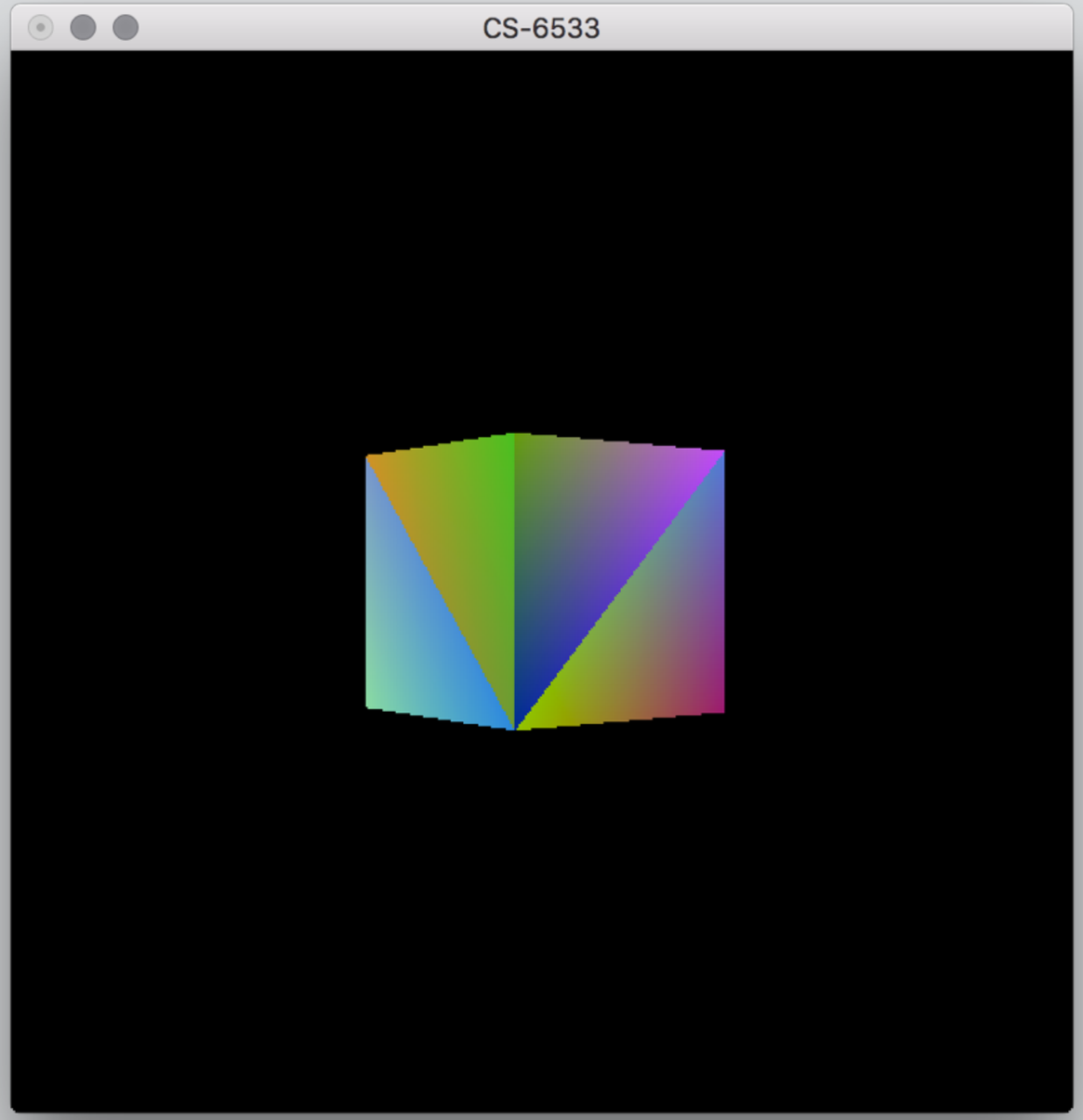
```
varying vec4 varyingColor;

void main() {
    gl_FragColor = varyingColor;
}
```

```
GLfloat cubeVerts[] = {
    -1.0f,-1.0f,-1.0f,
    -1.0f,-1.0f, 1.0f,
    -1.0f, 1.0f, 1.0f,
    1.0f, 1.0f,-1.0f,
    -1.0f,-1.0f,-1.0f,
    -1.0f, 1.0f,-1.0f,
    1.0f,-1.0f, 1.0f,
    -1.0f,-1.0f,-1.0f,
    1.0f,-1.0f,-1.0f,
    1.0f, 1.0f,-1.0f,
    1.0f,-1.0f,-1.0f,
    -1.0f,-1.0f,-1.0f,
    -1.0f,-1.0f,-1.0f,
    -1.0f, 1.0f, 1.0f,
    -1.0f, 1.0f,-1.0f,
    1.0f,-1.0f, 1.0f,
    -1.0f,-1.0f, 1.0f,
    -1.0f,-1.0f,-1.0f,
    -1.0f, 1.0f, 1.0f,
    -1.0f,-1.0f, 1.0f,
    1.0f,-1.0f, 1.0f,
    1.0f, 1.0f, 1.0f,
    1.0f,-1.0f,-1.0f,
    1.0f, 1.0f,-1.0f,
    1.0f,-1.0f,-1.0f,
    1.0f, 1.0f, 1.0f,
    1.0f,-1.0f, 1.0f,
    1.0f, 1.0f, 1.0f,
    1.0f, 1.0f,-1.0f,
    -1.0f, 1.0f,-1.0f,
    1.0f, 1.0f, 1.0f,
    -1.0f, 1.0f,-1.0f,
    -1.0f, 1.0f, 1.0f,
    1.0f, 1.0f, 1.0f,
    -1.0f, 1.0f, 1.0f,
    1.0f,-1.0f, 1.0f
};
```

```
GLfloat cubeColors[] = {
        0.583f,  0.771f,  0.014f, 1.0f,
        0.609f,  0.115f,  0.436f, 1.0f,
        0.327f,  0.483f,  0.844f, 1.0f,
        0.822f,  0.569f,  0.201f, 1.0f,
        0.435f,  0.602f,  0.223f, 1.0f,
        0.310f,  0.747f,  0.185f, 1.0f,
        0.597f,  0.770f,  0.761f, 1.0f,
        0.559f,  0.436f,  0.730f, 1.0f,
        0.359f,  0.583f,  0.152f, 1.0f,
        0.483f,  0.596f,  0.789f, 1.0f,
        0.559f,  0.861f,  0.639f, 1.0f,
        0.195f,  0.548f,  0.859f, 1.0f,
        0.014f,  0.184f,  0.576f, 1.0f,
        0.771f,  0.328f,  0.970f, 1.0f,
        0.406f,  0.615f,  0.116f, 1.0f,
        0.676f,  0.977f,  0.133f, 1.0f,
        0.971f,  0.572f,  0.833f, 1.0f,
        0.140f,  0.616f,  0.489f, 1.0f,
        0.997f,  0.513f,  0.064f, 1.0f,
        0.945f,  0.719f,  0.592f, 1.0f,
        0.543f,  0.021f,  0.978f, 1.0f,
        0.279f,  0.317f,  0.505f, 1.0f,
        0.167f,  0.620f,  0.077f, 1.0f,
        0.347f,  0.857f,  0.137f, 1.0f,
        0.055f,  0.953f,  0.042f, 1.0f,
        0.714f,  0.505f,  0.345f, 1.0f,
        0.783f,  0.290f,  0.734f, 1.0f,
        0.722f,  0.645f,  0.174f, 1.0f,
        0.302f,  0.455f,  0.848f, 1.0f,
        0.225f,  0.587f,  0.040f, 1.0f,
        0.517f,  0.713f,  0.338f, 1.0f,
        0.053f,  0.959f,  0.120f, 1.0f,
        0.393f,  0.621f,  0.362f, 1.0f,
        0.673f,  0.211f,  0.457f, 1.0f,
        0.820f,  0.883f,  0.371f, 1.0f,
        0.982f,  0.099f,  0.879f, 1.0f
};
```

```
#include "matrix4.h"

    GLuint modelviewMatrixUniformLocation;
      GLuint projectionMatrixUniformLocation;

// init

    modelviewMatrixUniformLocation = glGetUniformLocation(program, "modelViewMatrix");
    projectionMatrixUniformLocation = glGetUniformLocation(program, "projectionMatrix");

// render


    Matrix4 objectMatrix;
    objectMatrix = objectMatrix.makeZRotation(45.0);

    Matrix4 eyeMatrix;
    eyeMatrix = eyeMatrix.makeTranslation(Cvec3(-0.5, 0.0, 0.0));
    Matrix4 modelViewMatrix = inv(eyeMatrix) * objectMatrix;

    GLfloat glmatrix[16];
    modelViewMatrix.writeToColumnMajorMatrix(glmatrix);
    glUniformMatrix4fv(modelviewMatrixUniformLocation, 1, false, glmatrix);

  Matrix4 projectionMatrix;
    projectionMatrix = projectionMatrix.makeProjection(45.0, 1.0, -0.1, -100.0);

    GLfloat glmatrixProjection[16];
    projectionMatrix.writeToColumnMajorMatrix(glmatrixProjection);
    glUniformMatrix4fv(projectionMatrixUniformLocation, 1, false, glmatrixProjection);
```

Drawing a cube.


CS-6533