The Projection Matrix

Lecture 5

Robb T. Koether

Hampden-Sydney College

Wed, Aug 30, 2017
Outline

1. The World Coordinate System
2. The Projection Matrix
3. The Vertex Shader
4. Uniform Shader Variables
5. Assignment
1. The World Coordinate System

2. The Projection Matrix

3. The Vertex Shader

4. Uniform Shader Variables

5. Assignment
Definition (World Coordinate System)

The **world coordinate system** is the single coordinate system in which all objects are placed when the scene is rendered.
The default world coordinate system is a “square” with $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$, regardless of the size or shape of the window.

Typically, this is not the best choice.

To change the world coordinate system, we need a transformation.

The function `ortho2D()` will produce the appropriate transformation matrix (called the projection matrix), if we specify the coordinates of the window boundaries: left, right, bottom, top.
The projection matrix produced by `ortho2D()` is

\[
P = \begin{pmatrix}
\frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\
0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

where \( l = \text{left}, \ r = \text{right}, \ b = \text{bottom}, \ t = \text{top}, \)
Matrix multiplication $\mathbf{X}' = \mathbf{P}\mathbf{X}$ will perform the transformation.

\[
\begin{pmatrix}
  x' \\
  y' \\
  0 \\
  1
\end{pmatrix}
= \begin{pmatrix}
  \frac{2}{r-\ell} & 0 & 0 & -\frac{r+\ell}{r-\ell} \\
  0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
  x \\
  y \\
  0 \\
  1
\end{pmatrix}
\]
The default projection matrix uses \( \ell = -1, r = 1, b = -1, \) and \( t = 1, \) which produces the identity matrix.

Then the projection matrix is

\[
P = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} = I.
\]

Every point \( X \) is left unchanged: \( PX = IX = X. \)
Suppose our scene is drawn in a rectangle with left = $-4$, right = 4, bottom = $-3$ and top = 3.

Then the projection matrix is

$$P = \begin{pmatrix}
\frac{1}{4} & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$

Map the corners ($-4, -3$), (4, $-3$), (4, 3), and ($-4, 3$).

Map the point (2, 1).
Suppose our scene is drawn in a rectangle with left = 0, right = 8, bottom = 0 and top = 4.

Then the projection matrix is

$$ P = \begin{pmatrix} 
\frac{1}{4} & 0 & 0 & -1 \\
0 & \frac{1}{2} & 0 & -1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 
\end{pmatrix} $$

Map the point (4, 2).
Map the point (2, 1).
The Vertext Shader

The multiplication by $P$ takes place in the vertex shader (because the vertices are stored in the GPU buffer).

Therefore, we must pass the projection matrix to the vertex shader.

The shader will multiply it by the vertex to transform it.
Outline

1. The World Coordinate System
2. The Projection Matrix
3. The Vertex Shader
4. Uniform Shader Variables
5. Assignment
A **uniform** shader variable is a shader variable whose value does not change during the processing of the vertices of a primitive, i.e., during a call to `glDrawArrays()`. Its value is set by the application program and passed to the shader before calling `glDrawArrays()`.
Uniform Shader Variables

Passing a Shader Variable

```c
GLint glGetUniformLocation(program, var_name);
```

- In the application program, we must get a shader location for the uniform variable.
- The `glGetUniformLocation()` will return a location.
The functions `glUniform*()` and `glUniformMatrix*()` will pass the value(s) to the shaders.

- The third parameter of `glUniformMatrix*()` tells whether the matrix is stored in row-major order (row by row rather than column by column).
- See p. 48 of the Red Book.
This code will create the projection matrix and pass it to the shaders.

"proj" is the name of the uniform variable in the shader.

It is a really good idea to keep the same name in order to avoid confusion.

Later, we will have many uniform variables.
uniform mat4 proj;

layout (location = 0) in vec2 vPosition;

void main()
{
    gl_Position = proj * vec4(vPosition, 0.0f, 1.0f);
}

- In the shader program, we simply declare the variable to be uniform.
- The name must match the name specified in the application program.
- Then multiply it by the position vector and assign to gl_Position.
Assignment

- Assignment 5, to be turned in by Monday.
- Read pp. 203 - 210, User Transformations.