The Projection Matrix

Lecture 5

Robb T. Koether

Hampden-Sydney College

Wed, Sep 4, 2019

Outline

Outline

World Coordinates

Definition (World Coordinate System)

The world coordinate system is the single coordinate system in which all objects are placed when the scene is rendered.

World Coordinates in 2D

ortho2D()

```
void ortho2D(int left, int right, int bottom, int top)
```

- 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 Viewport

glViewport()

```
void glViewport(int x, int y, int width, int height)
```

- The viewport is that rectangular part of the window in which the drawing is done.
- The `glViewport()` function sets the viewport. The parameters `x` and `y` are the coordinates of the lower-left corner of the viewport.
- The projection matrix maps the world coordinates into the viewport coordinates.
- It is standard practice to set the viewport to the full window.
- The viewport is initialized to the full window.

Outline

The Projection Matrix

- The projection matrix produced by `ortho2D()` is

$$\mathbf{P} = \begin{pmatrix} \frac{2}{r-\ell} & 0 & 0 & -\frac{r+\ell}{r-\ell} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $\ell =$ left, $r =$ right, $b =$ bottom, $t =$ top,

The Projection Matrix

- Matrix multiplication $\mathbf{X}' = \mathbf{PX}$ will perform the transformation from world coordinates to viewport coordinates.

$$\begin{pmatrix} x' \\ y' \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{r-\ell} & 0 & 0 & -\frac{r+\ell}{r-\ell} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}$$

The Projection Matrix

The Projection Matrix

- The default projection matrix uses $\ell = -1$, $r = 1$, $b = -1$, and $t = 1$, which produces the identity matrix.
- Then the projection matrix is

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{I}.$$

- Every point $\mathbf{X}$ is left unchanged: $\mathbf{PX} = \mathbf{IX} = \mathbf{X}$.

The Projection Matrix

The Projection Matrix

- Suppose our scene is drawn in a rectangle with left $= -4$, right $= 4$, bottom $= -3$ and top $= 3$.
- Then the projection matrix is

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

- Map the corners $(-4, -3)$, $(4, -3)$, $(4, 3)$, and $(-4, 3)$.
- Map the point $(2, 1)$.

The Projection Matrix

The Projection Matrix

- Suppose our scene is drawn in a rectangle with left $= 0$, right $= 8$, bottom $= 0$ and top $= 4$.
- Then the projection matrix is

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

- Map the point $(4, 2)$.
- Map the point $(2, 1)$.

Outline

The Vertex 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

Uniform Shader Variables

- 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

```
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, which we need to save.

Uniform Shader Variables

Passing a Shader Variable

```
void glUniform*(location, value);
void glUniform*(location, count, values);
void glUniformMatrix*(location, count, GL_TRUE, values);
```

- 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.

Passing the Projection Matrix

Passing the Projection Matrix

```
mat4 proj = ortho2D(left, right, bottom, top);

GLuint proj_loc = glGetUniformLocation(program, "proj");

glUniformMatrix4fv(proj_loc, 1, GL_TRUE, proj);
```

- This code with 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.

Using the Projection Matrix

Using the Projection Matrix

```
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`.

Outline

Assignment

Assignment

- Read pp. 203 - 210, User Transformations.