

3D Graphics

Lecture 12

Robb T. Koether

Hampden-Sydney College

Fri, Sep 15, 2017

Outline

- 1 3D Graphics
- 2 The View Matrix
- 3 The Projection Matrix
- 4 Assignment

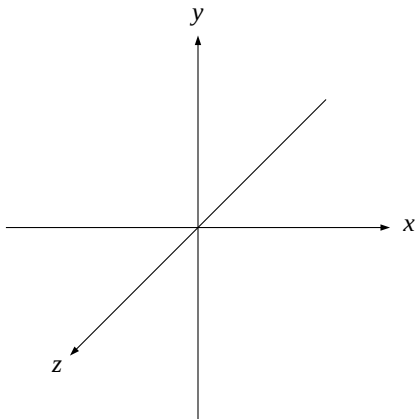
Outline

- 1 3D Graphics
- 2 The View Matrix
- 3 The Projection Matrix
- 4 Assignment

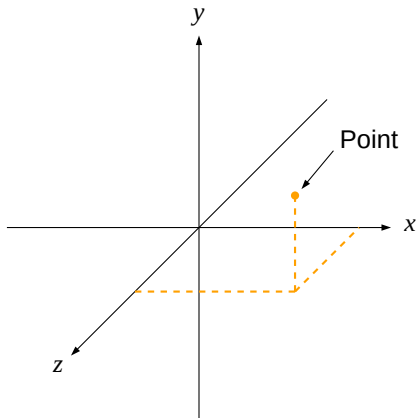
The 3D Coordinate System

- The 3-dimensional coordinate system has 3 axes.
 - The x -axis runs from left to right.
 - The y -axis runs from bottom to top.
 - The z -axis runs from back to front.
- They form a **right-hand** coordinate system.
- Vertices will consist of three **floats**, for x , y , and z .
- The fourth coordinate, w , should be set to `1.0`.

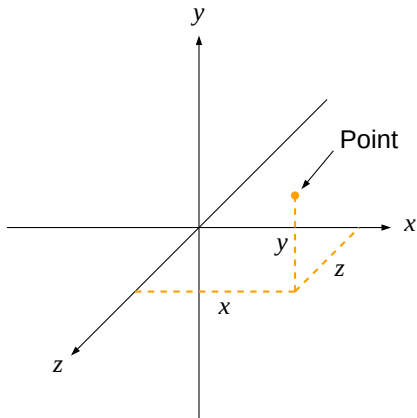
The Coordinate Axes



The Coordinate Axes



The Coordinate Axes



Outline

- 1 3D Graphics
- 2 The View Matrix**
- 3 The Projection Matrix
- 4 Assignment

The Viewpoint

- In three dimensional graphics, we must establish a viewpoint, usually referred to as the **eye** or the **camera** position.
- We must also specify the **look point**, i.e., the point at which we are looking, usually the center of our scene.
- And we must specify the camera's orientation, usually straight up.
- The default values are
 - eye = $(0, 0, 0)$, the origin.
 - look = $(0, 0, -1)$, down the negative z -axis.
 - up = $(0, 1, 0)$, up the positive y -axis.

The `lookAt()` Function

The `lookAt()` Function

```
mat4 lookAt(vec3 eye, vec3 look, vec3 up);
```

- `eye` is the location of the **eye point**.
- `look` is the location of the **look point**.
- `up` is the **up vector**. It points in the “upward” direction with respect to the camera.
- The `lookAt()` function returns a 4×4 **view matrix** that represents the transformation of moving and reorienting the scene from the origin to the eye point.

The `lookAt()` Function

The `lookAt()` Function

```
vec3 eye(5.0f, 4.0f, 3.0f);  
vec3 look(0.0f, 0.0f, 0.0f);  
vec3 up(0.0f, 1.0f, 0.0f);  
  
mat4 view = lookAt(eye, look, up);  
  
GLuint view_loc = glGetUniformLocation(program, "view");  
glUniformMatrix4fv(view_loc, 1, GL_FALSE, view);
```

- We need to pass the view matrix to the vertex shader.
- Because it will be the same matrix for all vertices, we should pass it as a uniform parameter.

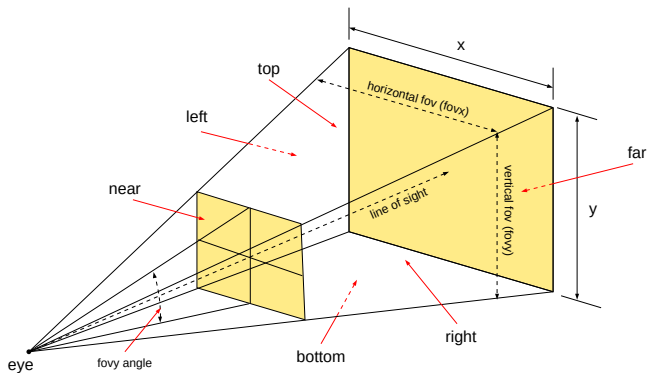
Outline

- 1 3D Graphics
- 2 The View Matrix
- 3 The Projection Matrix**
- 4 Assignment

The View Frustum

- A **frustum** is a truncated pyramid.
- The **view frustum** encloses the part of the scene that will be rendered.
- The vertex (of the untruncated pyramid) is located at the eye point.
- We may think of the base of the view frustum as the plane onto which the scene is projected.

The View Frustum



The `perspective()` Function

The `perspective()` Function

```
mat4 perspective(GLfloat fovy, GLfloat aspect,  
                 GLfloat near, GLfloat far);
```

- `fovy` is the vertical angle of the field of view.
- `aspect` is the aspect ratio (width/height).
- `near` is the distance from the eye to the near plane.
- `far` is the distance from the eye to the far plane.
- The `perspective()` function returns a 4×4 **projection matrix** that represents the transformation of creating the perspective view of the scene.

The perspective() Function

The perspective() Function

```
mat4 proj = perspective(60.0f, 16.0f/9.0f, 0.1f, 100.0f);
```

```
GLuint proj_loc = glGetUniformLocation(program, "proj");  
glUniformMatrix4fv(proj_loc, 1, GL_FALSE, proj);
```

- We need to pass the projection matrix to the vertex shader.
- Because it will be the same matrix for all vertices, we should pass it as a uniform parameter.

Outline

- 1 3D Graphics
- 2 The View Matrix
- 3 The Projection Matrix
- 4 Assignment**

Assignment

Assignment

- Assignment 11.
- Read pp. 217 - 220, Perspective projection.
- Also, read the `perspective()` and `lookAt()` functions in `vmath.h`.