CG CSE 533 Lab 03

View Projection

September 21, 2022

1 Introduction

In this lab we will be looking into the viewing matrices and understand the global picture of the transformation of world space to screen space. In the previous labs we have worked in screen space coordinates i.e. we did not need to perform any transformations to map world space coordinates to screen space, the coordinates of the objects we rendered were directly used as screen coordinates. In order to transform the coordinates from one coordinate space to the next we perform matrix multiplications, the model, view and projection matrices.

2 Transformations

There are 3 primary transformations that we apply for rendering a scene -

2.1 World Transform

It refers to the coordinates of the object in the real world. The world transform matrix or the model matrix transforms the models vertices/normals from object space into world space relative to some origin.

2.2 View Transform

This transformation corresponds to the view one would see through a camera. The it creates a view relative to the camera or viewpoint within world. The view space is also referred as camera space or eye space since this is space from which you view the object.

2.3 Projection Transform

Once the coordinates have been transformed to the view space we project them to clip coordinates. These coordinates form the NDC (normalazed device coordinates) and are processed to be between -1.0 and 1.0 to check which will be displayed on the screen. Perspective projection also takes place at this stage. Post this step the coordinates are transformed to screen coordinates via the viewport transform which transforms the coordinates from 1.0 and 1.0 to the coordinate range defined by `glViewport`.These coordinates are sent to rasterizer.

2.3.1 Setup

build the program by running make and run Lab3 executable thus generated.

3 Under the hood

3.1 Model View Projection - MVP

3.1.1 Model Matrix

We setup our model matrix in the function `setupModelTransformation` which generates the rotation and translation matrix. This matrix is then passed to the vertex shader using `uniform` variable.

3.1.2 View Matrix

We then setup the view matrix in `setupViewTransformation` which consists of the the `lookat` matrix. The look at matrix consists of 3 parts, the eye, center and up vectors. The eye determines the position of the viewer, the center vector informs the position where the camera is looking at. While the up vector tells about the orientation of the camera, specifying the up direction for the camera. This matrix is then passed to the vertex shader using a uniform variable.

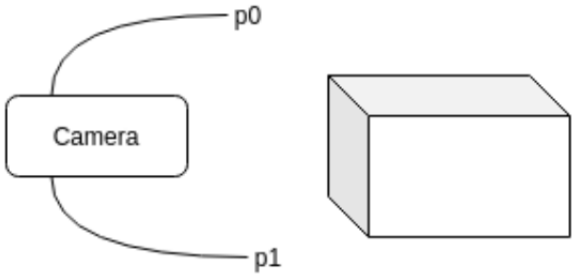

Figure 1: Move the camera starting from top point p0 to point p1 along a parabola.

3.2 Projection Matrix

Next we setup the projection matrix `setupProjectionTransformation` where we specify the fov (field of view), aspect, near and far.

4 Task

Add implementation for moving the camera vertically while looking at the same point and move the camera along a parabolic path from top to bottom.

4.1 Hint

Use a slider to control the movement of camera from top to bottom as done in previous translation task. Use the equation of parabola to deduce the appropriate z-coordinate and update the look at matrix. Thus, pass the coordinates to update the view matrix. Note, do not change the point you are looking at i.e. the center in `lookat`.

5 Submission Instructions

Submit the output along with code for evaluation. Further record screen and upload a gif/video demonstrating the widgets. You can use Kazam to record. Upload the zip file of code and output. Name the zip file as Lab03_<name_ roll no>.zip

6 References

- http://davidlively.com/programming/graphics/coordinate-spaces/
- https://learnopengl.com/Getting-started/Coordinate-Systems