CSCI 4250/5250 Homework 5 (Due beginning of class, Tuesday Oct 31st)

1. Given the 3D cube example in programs: ortho.js and ortho.html (available on the course web page), if the view position and the orthographic viewing volume is changed into each of the following situations, how will the final 2D image change from its original image? Justify your answer.
2. mvMatrix=lookAt(vec3(-4, 0, 0), at, up); // pMatrix does not change

The ‘camera’ will be to the left of the box, and looking to the right. You can see the yellow side. This is justified by the first parameter of lookAt being the eye position, which in this case is set to x = -4, y = 0, z = 0, so the camera is 4 to the left.

1. mvMatrix=lookAt(vec3(3, 3, 3), at, up); // pMatrix does not change

The camera in this case is above, to the right, and in front of the box, so the three sides will be red on the left (front), cyan on the top (top), and blue on the right (right). This can be justified the same way as part a).

1. mvMatrix=lookAt(vec3(3, 3, 3, at, up); pMatrix=ortho(-3, 3, -3, 3, -1, 1);

In this case, the camera angle is unchanged from part b), however the ortho projection changes to be 2 wider (one on each side), 2 shorter (one each top and bottom, and the range is reduced by 8, and changed to be from 1 in front to 1 behind the camera. This means the box will look thinner horizontally, and bigger vertically, except the near and far planes do not intersect the box (the far plane is too close) so nothing is rendered.

1. pMatrix= ortho(-6, 6, -3, 3, 2, 10); // mvMatrix does not change

In this case the ortho projection is twice as wide, so the box will be half the previous width (if it was drawn), and the vertical field is unchanged. The distance range is changed from 1 in front and behind to from 2 in front to 10 in front. From this angle the ‘2’ near-distance isn’t enough to intersect the corner, so we see the same image as in part b, only thinner and wider (wider from the change in part c)

1. pMatrix=ortho(0, 4, 0, 3, 2, 10); // mvMatrix does not change

In this example, the ortho projection is shifted so that it has a width of 4, skewed 2 to the right, a height of 3, skewed 1.5 up vertically, and with the same depth range as part d. This will show the only the top-right quadrant of the cube vertices. So we will be able to see the right half of the cyan, and some of the upper part of the blue sides, and none of the red. It will appear in the bottom left corner of the canvas as this is where the ortho projection clips the shape.

1. Given: mvMatrix=lookAt(vec3(4, 4, - 4), at, up);

pMatrix=ortho(-2, 2, -4, 4, -10, 10);

show:

• the mvMatrix

First subtract the look-at coordinates from the eye coordinates

(4, 4, -4) – (0, 0, 0) = (4, 4, -4)

Next normalize the result to get the view direction

View direction = n = (0.5774, 0.5774, -0.5774)

Then get the perpendicular vector by normalizing the cross product of the up vector and the view direction.

Perpendicular vector = u = (-0.7071, 0, -0.7071)

Now find the new up vector by normalizing the cross product of the view direction and the perpendicular vector

New up = v = (-0.4082, 0.8165, 0.4082)

Next find the negative dot product of the View Direction and Eye, the Perpendicular vector and Eye, and the new Up vector and Eye

-dot(n, eye) = -6.282

-dot(u, eye) = 0

-dot(v, eye) = 0

Finally, append the results into a new 4x4 matrix as follow

Vec4(U, dotU)

Vec4(V, dotV)

Vec4(N, dotN)

Vec4(0, 0, 0, 1)

Which gives (needs to be transposed for column major):

((-0.7071, 0, -0.7071, 0)

(-0.4082, 0.8165, 0.4082, 0)

(0.5774, 0.5774, -0.5774, -6.282)

(0, 0, 0, 1))

• the pMatrix

The pMatrix is easier. Begin by calculating the width, height, and depth (right – left, etc)

W = 4

H = 8

D = 20

Then define a new matrix as follows (again, needs to be transposed for column major order)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0 / W | 0 | 0 | - (left+right) / W |
| 0 | 2.0 / H | 0 | -(top+bottom) / H |
| 0 | 0 | 2.0 / D | -(near+far) / D |
| 0 | 0 | 0 | 1 |

Which gives:

|  |  |  |  |
| --- | --- | --- | --- |
| 0.5 | 0 | 0 | 0 |
| 0 | 0.25 | 0 | 0 |
| 0 | 0 | -0.1 | 0 |
| 0 | 0 | 0 | 1 |

• the coordinates of a point F(1, 1, -1) when converted into the final clip coordinates. (show intermediate steps in deriving the results)

pMatrix \* mvMatrix \* point = (0, 0, 0.5196, 1)

1. Changing the orthographic viewing volume in problem 2) to a frustum with left=-2, right=2, bottom=-4, top=4 for the near plane, and the near plane at distance 4 and far plane at distance 10 from the eye/camera. How would you call the perspective function to set up the corresponding pMatrix in the .js program?

pMatrix = perspective (fov, aspect, 4, 10), where

aspect = (2- -2)/(4- -4)

fov = 2\*arctan(1/2\*(top-bottom)/Near) = 2\*arctan(1/2\*(4- -4)/4)

1. With the perspective viewing volume defined in problem 3), what will be the x and y coordinates of the two points F(1, 1, -1) and B(1, 1, 1) when projected onto the near plane?