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.

a. mvMatrix=lookAt(vec3(-4, 0, 0), at, up); // pMatrix does not change

The eye change position to -4 in x, look directly at: left (GCAE) yellow.

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

The eye change position to front, right, top and look at 3 side: front(ABDC) red, right (DBFH) blue, top (AEFB) cyan.

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

We cannot see the image of the cube anymore.

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

The eye change position to higher top right and look at 3 side: back (AEFB) cyan, right (DBFH) blue, back (EFHG) magenta. The image get smaller in width and taller in height.

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

The eye look directly at EF line and see 2 side: (EFHG) magenta, and (AEFB) cyan in the bottom corner of the view port.

2) Given: mvMatrix=lookAt(vec3(4, 4, - 4), at, up); pMatrix=ortho(-2, 2, -4, 4, -10, 10);

show:

* the mvMatrix: n = eye - look = (4,4,-4) —> normalize n: (0.58, 0.58, -0.58)

u = up x n = (0,1,0) x (4,4,-4) = (-4, 0, -4) —> normalize u: (-0.7, 0, -0.7)

v = n x u = (4,4,-4) x (-4,0,-4) = (16, -32, -16) —> normalize v: (0.4, -0.81, -0.4)

dx = -eye . u = (-4,-4,4) . (0.58, 0.58, -0.58) = -6.92

dy = -eye . u = (-4,-4,4) . (-0.7, 0, -0.7) = 0

dz = -eye . u = (-4,-4,4) . (0.4, -0.81, -0.4) = 3.26

V = [ -0.7 0 -0.7 -6.92

0.4 -0.81 -0.4 0

0.58 0.58 -0.58 3.26

0 0 0 1 ]

* the pMatrix:

( Copied the result from console.log )

P = [0.16666666666666666, 0, 0, -0]

[0, 0.3333333333333333, 0, -0]

[0, 0, -0.25, -1.5]

[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)

(No answer)

3) 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?

frustum(left, right, bottom, top, near, far)

-> frustum(-2, 2, -4, 4, 4, 10)

aspect = (right-left)/(top-bottom) = 0.5

fov = 2\* arctan(1/2\*(top-bottom)/N) = 90 (Degree) = 1.57 (rad)

=> perspective(fov, aspect, near, far) = (1.57, 0.5, 4, 10)

4) 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?

P = [ 2N/ (r-l) 0 (r+l)/(r-l) 0]

0 2N(t-b) (t+b)(t-b) 0

0 0 a b

0 0 -1 0]

P = [ 2 0 0 0]

0 1 0 0

0 0 -7/3 -40/3

0 0 -1 0]

Fprojected = P . F = (2, 1, 47) => xF = 2; yF = 1

Bprojected = P . B = (2, 1, -47) => xB = 2; yB = 1