CPSC 314
Theory Assignment 2

Due Thursday October, 13 2011 in class

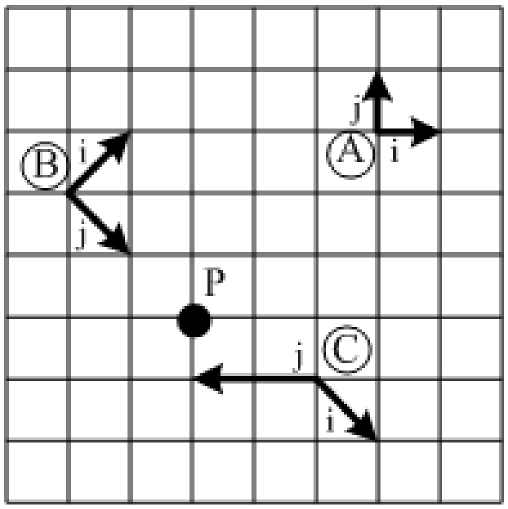
Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name:Provided Solution ____________________

Student Number: ____________________

Question 1	/ 1
Question 2	/ 1
Question 3	/ 1
Question 4	/ 1
Question 5	/ 1
Question 6	/ 1
Question 7	/ 1
TOTAL	/ 7

1. Transformation as a Change of Coordinate Frame



Derive a transformation that takes a point from frame C to frame B, i.e., determine $M_{C\to B}$, where $P_B = M_{C\to B}P_C$. Verify your solution using the coordinates of P with respect to the different frames (see your answers in assignment 0).

Solution 1 *As we see from the image, the transformation is some mix of translations, scaling and rotations, i.e. it is an affine transformation. Therefore, the trasformation of coordinates between two systems is done by multiplication by a constant matrix* $M_{C\to B}$*. As was discussed on the lectures, given two orthonormal coordinate systems* XY *and* UV*, the transformation matrix* R *from one to another is given by*

$$\begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix},$$

where $u_{\{x,y\}}, v_{\{x,y\}}$ *are the coordinates of* u, v *in* XY *system. This is 2x2 matrix, therefore it can only describe linear transformations in 2D, i.e. combination of scaling, shearing, reflections, and rotations. Here we want to describe an affine transformation in 2D, i.e. including translation as well, so in this case we need a 3x3 matrix. Let's denote the origin of* UV *system by* O^{UV}*. The 2x2 submatrix of it stays the same, only the translation component is added:*

$$\begin{pmatrix} u_x & v_x & O^{UV}_x \\ u_y & v_y & O^{UV}_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

So, in our case, to find this matrix $M_{C\to B}$ *we should only find coordinates of basis vectors and origin of* C *in system* B*. From the image we can see that* $i_C = j_B$*, and* $j_C = -i_B - j_B$*. So, in system* B *the coordinates of basis vectors are* $i_C = (0, 1)$ *and* $j_C = (-1, -1)$*. We also can find the coordinates of the origin* $O_C = (0.5, 3.5)$*. Then, following the formula 1, the transformation matrix is:*

$$\begin{pmatrix} 0 & -1 & 0.5 \\ 1 & -1 & 3.5 \\ 0 & 0 & 1 \end{pmatrix}$$

We can also verify our answer using the coordinates of P and comparing with answers in Assignment 1:

$$P_B = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0.5 \\ 1 & -1 & 3.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 0.5 \\ 1 \end{pmatrix} = P_C$$

Solution 2 *This solution works for arbitrary transformations, including non-affine ones, when matrix $M_{C\to B}$ may be not constant. Let's take arbitrary point D. If we denote its coordinates in B and C systems by (x_B, y_B), (x_C, y_C) respectively. Then it can be expressed in the following way:*

$$D = x_B\mathbf{i_B} + y_B\mathbf{j_B} + O_B = x_C\mathbf{i_C} + y_C\mathbf{j_C} + O_C \tag{2}$$

To find expression for x_B, y_B through x_C, y_C we need to express all the vectors in one coordinate system. Let's take A as the reference system. Then $\mathbf{i_B} = \mathbf{i_A} + \mathbf{j_A}$, $\mathbf{j_B} = \mathbf{i_A} - \mathbf{j_A}$, $i_C = \mathbf{i_A} - \mathbf{j_A}$, $j_C = -2i_A$. Substituting this into 2, we get:

$$\mathbf{i_A}(x_C - 2y_C - 1) + \mathbf{j}(-x_C - 4) = \mathbf{i_A}(-5 + x_B + y_B) + \mathbf{j_A}(-1 + x_B - y_2)$$

As $\mathbf{i_A}$ and $\mathbf{j_A}$ are linearly independent, we get system of equations:

$$x_C - 2y_C - 1 = -5 + x_B + y_B \tag{3}$$
$$-x_C - 4 = -1 + x_B - y_B \tag{4}$$

Solving this for x_B, y_B, we obtain:

$$\begin{pmatrix} x_B \\ y_B \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0.5 \\ 1 & -1 & 3.5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \end{pmatrix}$$

2. Given a line segment $S = (P_0, P_1)$ in 2D and a point P, write an algorithm to find if the point is on the line segment. Note that your algorithm should operate in the Euclidean (not discrete) space.

   Solution 1 *One of possible solutions is the following algorithm. First we check if P is on the line going through P_0, P_1 and if that's true, check if P is between them. To find the equation of the line, we write out the general line equation $Ax + By + C = 0$, then use the fact that it goes through P_0 and P_1:*

$$\begin{cases} Ax_{P_0} + By_{P_0} + C = 0 \\ Ax_{P_1} + By_{P_1} + C = 0 \end{cases}$$

We solve this system for A, B, C (remembering the fact that line equation coefficients are defined up to a constant factor, so we can take, for example, $A = 1$ if we know that the line is not horizontal), and then check if that holds for P as well:

$$Ax_P + By_P + C \overset{?}{=} 0$$

If that is true, then the only thing we have to check is that if P is within the box formed by P_0, P_1, i.e. check:

$$\begin{cases} x_{P_0} \leq x_P \leq x_{P_1} \\ y_{P_0} \leq y_P \leq y_{P_1} \end{cases}$$

If that's true, the point is on the segment, otherwise - no.

Solution 2 *Probably the easiest algorithm is the following. Calculate Euclidian distances $||\overline{P_0P}||, ||\overline{PP_1}||, ||\overline{P_0P_1}||$ and check if $||\overline{P_0P}|| + ||\overline{PP_1}|| = ||\overline{P_0P_1}||$. If yes, return true, otherwise false.*

3. Decompose the following complex transformations in homogeneous coordinates into a product of simple transformations (scaling, rotation, translation, shear). Pay attention to the order of transformations.

   (a)

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

As a reminder, for aribtrary 4x4 transformation matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & t_1 \\ a_{21} & a_{22} & a_{23} & t_2 \\ a_{31} & a_{32} & a_{33} & t_3 \\ 0 & 0 & 0 & w \end{pmatrix}$$

the (t_1, t_2, t_3) is the translation vector of the coordinate system, w is the uniform scaling factor, and 3x3 matrix $a_{i,j}$ describes the rotations, scalings, mirroring and shearing.

Here looking at the structure of the matrix (let's denote it C), we can see that it is a product of two transformations: rotation A and uniform scaling B, where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = Rot(-90, z)$$

is rotation by 90 degrees clockwise around z-axis, and

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} = Scale(1/2, 1/2, 1/2),$$

is a uniform scaling by 1/2. The last thing we can notice that those matrices are commutative, i.e. $AB = BA = C$.

(b)

$$\begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denoting the given matrix C, *we can notice that it is the product of two matrices:*

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = Scale(1, 2, 3),$$

and

$$B = \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = Translate(-2, 1, 0).$$

Now we have to notice that if we first scale, and then translate, the translation will be done in respect to the new scaled basis vectors, therefore the translation component in the resulting matrix will be scaled as well. So, $C = BA$.

(c) What is the inverse of the transformation matrix in part (b) of this question?

While in general case we would have to do a full-blown inversion of the matrix, here we can save some time, since we know the decomposition of the matrix: $C = BA$. *Moreover, we know, that inverse transformation for scaling is again scaling, but with inverse factors. In the same way, inverse transformation for translation is translation by the negated original vector. So,*

$$C^{-1} = A^{-1}B^{-1} = Scale(1, \frac{1}{2}, \frac{1}{3})Translate(2, -1, 0) = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

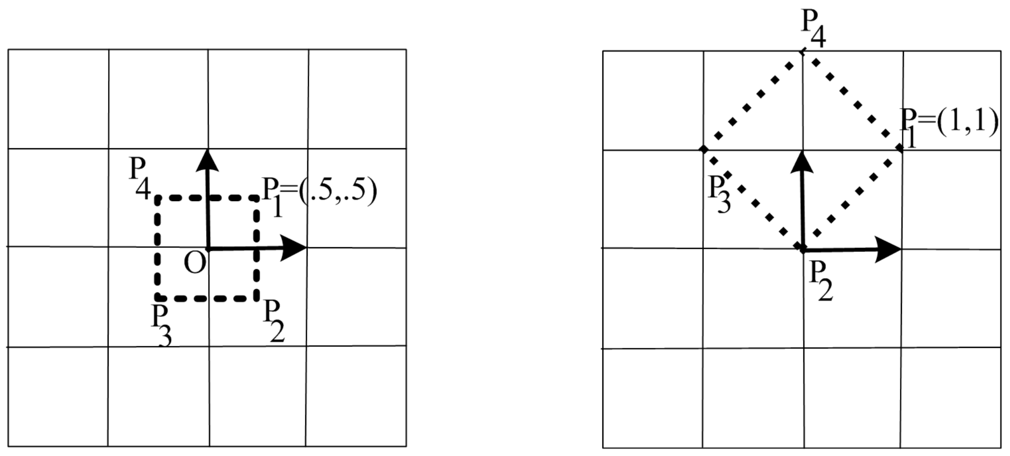
(d) Give the sequence of OpenGL transformations that would produce the same transformation matrix as in part (a) of this question.

Each time we're drawing something in OpenGL, we're using local coordinates, and each time we peform any transformation in OpenGL, we right-multiply the current matrix. Therefore, as in the previous question the order of transformations was based on world coordinates, here we reverse the order:

$$glRotate(-90, 0, 0, 1); glScalef(0.5, 0.5, 0.5);$$

but as we figured out earlier, in this particular case the order of transformations does not matter.

4. Write down the 2D transformation matrix that maps the unit square centered at the origin as shown on the left to the square on the right in the figure below. Show your work.



One of the ways of describing the transformation is to first translate the square by 1 up, then rotate it by 45 degrees CW, and finally scale it by $\sqrt{2}$*:*

$$M = Translate(0,1)Rot(-45)Scale(\sqrt{2},\sqrt{2},\sqrt{2}) = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Another, absolutely legitimate way of describing this transormation is that we first rotate the square by 45 degrees CW, then scale it by $\sqrt{2}$*, and finally translate it up by 1 (before the scaling we wound translate it by* $\frac{1}{\sqrt{2}}$*, but now the axes have different lengths). Here we have to be cautious: the direction 'up' is no longer y-axis after the rotation, in new coordinates it is vector (-1/2, 1/2). So,*

$$M = Rot(-45)Scale(\sqrt{2},\sqrt{2},\sqrt{2})Translate(-0.5,0.5) = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

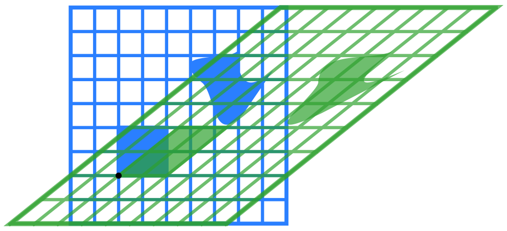
5. Answer yes/no and provide a short explantion. All the transformations are in 3D.
   (a) Does perspective transformation preserve parallel lines? *No, as it transforms, for example, two lines going from the viewer to the horizon, to two meeting lines.*
   (b) Is shear * translate = translate * shear? *No, since*

$$Shear(\alpha)Translate(x,y) = \begin{pmatrix} 1 & \alpha & x+\alpha y \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \neq \begin{pmatrix} 1 & \alpha & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} = Translate(x,y)Shear(\alpha)$$

   (c) Is shear1 * shear2 = shear2 * shear1?
   No, for example:

$$\begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix} = \begin{pmatrix} \alpha\beta+1 & \alpha \\ \beta & 1 \end{pmatrix} \neq \begin{pmatrix} 1 & \alpha \\ \beta & 1+\alpha\beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix}\begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$$

(d) Does shear preserve lenghts?
No, it's clear on the example of horizontal shear of a square with sides parallel to axis. It preserves the lengths of horizontal sides, but changes the lengths of the vertical sides (see figure).



6. Given the triangle T with vertices $P_1 = (1,0,0)$,$P_2 = (0,2,0)$, $P_3 = (0,1,1)$ and the transformation

$$S = \begin{pmatrix} 1 & -0.5 & 0 & 0 \\ 0.5 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(a) Compute the vertices of the triangle after applying the transformation S to it.
First we convert coordinates of each point to homogeneous ones by adding 1 in the end, i.e. $P_1^H = (1,0,0,1)$,$P_2^H = (0,2,0,1)$, $P_3^H = (0,1,1,1)$, *and then, by direct multiplication and conversion from the homogeneous coordinates, we get:* $P_1' = (1,0.5,0), P_2' = (-1,4,0), P_3' = (-0.5,2,1)$

(b) Compute the normal of the triangle before and after applying the transformation S to it.
We calculate the original normal by vector product: $n = (\frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}) = (P_2 - P_1) \times (P_3 - P_1)$. *To calculate the transformed normal and avoid unnecessary brute-force calculation of vector product, and norms, we can use the formula for transformation of normals:*

$$n' = Qn$$

$$Q = (S^{-1})^T$$

Calculating of the inverse matrix is easy in this case: as most of the matrix is unit, we need to inverse only 2x2 matrix

$$\begin{pmatrix} 1 & -0.5 \\ 0.5 & 2 \end{pmatrix} = \begin{pmatrix} \frac{8}{9} & \frac{2}{9} \\ -\frac{2}{9} & \frac{4}{9} \end{pmatrix}$$

so, the inverted and transposed matrix is equal to:

$$(S^{-1})^T = \begin{pmatrix} \frac{8}{9} & -\frac{2}{9} & 0 & 0 \\ \frac{2}{9} & \frac{4}{9} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and

$$n' = \frac{1}{\sqrt{21.3125}}(3.5, 2, 2.25)$$

7. Prove that transforming a line segment using an affine transformation is equivalent to transforming its end points, or in other words prove that

$$T((1-u)P_1 + uP_2) = (1-u)T(P_1) + uT(P_2)$$

Why are the two claims above equivalent?
Affine transformations can be represented by multiplication of 4x4 matrices by homogeneous coordinates of the points. I.e. in other words, there exists such matrix M_T so that for any point p, $T(p) = M_T p^H$, where p_H are the homogeneous coordinates of p. Then the original equation is equivalent to:

$$M_T((1-u)P_1^H + uP_2^H) = (1-u)M_T P_1^H + uM_T P_2^H,$$

but matrix multiplication is linear (i.e. distributive and commutative with a scalar), therefore $M_T((1-u)P_1^H + uP_2^H) = M_T(1_u)P_1^H + M_T uP_2^H = (1-u)M_T P_1^H + uM_T P_2^H$, which completes the proof.