BASIC TRANSFORMATIONS

DUE: week of March 20, during lab
checkpoints: weeks of February 27, March 6

This assignment is intended to introduce you to the mechanics of some of the basic coordinate transformations used in computer graphics: scaling, translation, rotation and shearing. A program skeleton in C# is provided that takes care of setting up a simple GUI, so that all you need to do is implement event handlers for a collection of toolbuttons. You may use any suitable programming environment you wish (eg. JAVA, C++, VB, etc). If you decide to use a language other than C#, you will be responsible for setting up an appropriate GUI on your own.

For the moment, the basic shape you may work with is either of the letter Q designs provided with the program skeleton, though your program may eventually be tested with a quite different shape! The program skeleton already reads the data files to obtain "world" coordinates of the vertices in the character, and information about the endpoints of the line segments as described above. You may assume that the character is designed on an approximately 20 x 20 unit grid, with the y-axis upwards. The drawing logic is present, so that initially the shape is drawn in the upper left hand corner of the client area, upside down because on the computer screen, y increases downwards.

What you must do for this assignment is:

Add appropriate code to the program skeleton so that the program:

(i) initially displays your block character on screen/window, "top-upwards" with center at the center of the screen, and vertical height equal to half the vertical height of the screen/window.

(ii) responds to mouse clicks on existing toolbar buttons to do the following operations to the shape:

- translate in the indicated direction (75 pixels at a time left or right along x, 35 pixels at a time up or down along y)
- scale size up by 10%/down by 10% without changing the position of its center
- rotate about the selected axis (x, y, or z) through the geometric center by an angle of 0.05 radians.
- establish continuous rotation about the axis indicated (x, y, or z). During the rotation the center of the character may not shift position nor may the character change intrinsic size.
- shear the character in the x-direction with respect to y, either rightwards at the top by 10% or leftwards at the top by 10%. The shearing operation must leave the baseline of the character -- the image of points with y-coordinate zero in the diagram on the last page of this document -- stationary. You may assume that your character shape will not be rotated from its initial orientation when shearing is to be done. The baseline of the shape is the horizontal line that started out at y = 0 in the original paper design of the shape. You may not assume that any point in the data file will have a y-coordinate equal to zero.
- reset the character to its initial size/position.

All geometric transformations of the screen image must be implemented using the cumulative transformation matrix approach – NO EXCEPTIONS! An executable capable of meeting roughly these requirements is also available through the course Out: folder.

Further Details

(i) All transformations (translation, scaling, rotation, shearing here) must be implemented as matrix transformations, based on the use of homogeneous coordinates in three-dimensional space. This includes the initial transformations of the input object coordinates to the initial image on the screen. You must code the matrix multiplication or equivalent into this program yourself – do not make use of a matrix class library or equivalent developed in another course or by someone else.

(ii) Even if the designed character is essentially just two-dimensional, the possibility of rotations about the x-axis and the y-axis means that your character will have to be a planar object in three-dimensions, and so the homogeneous coordinates of the points must be four-component. So, even for essentially planar shapes, the data file will contain z-coordinates of 0.0 for each point. (As your program reads the point-coordinate information, it automatically sets the homogeneous coordinate to 1.0.)

(iii) Two pairs of data files are provided for your use in testing your program during development: Qpoints.200810.dat and Qlines.200810.dat produce a planar image; Qpoints3d.200810.dat and Qlines3d.200810.dat produce a three-dimensional wireframe image. Either pair of files will work with the skeleton program as well as with the full demo program. You may assume that data files are set up so that the first point is the location of the center for purposes of scaling and rotation. Your program may also assume that the baseline of the character for shearing is at y = 0 in the initial design, **but it may not assume that any point in the data file actually has a y-coordinate of 0**. (This is the second time this statement has been made – it must be important!).

(iv) Although most of the coordinates in the simple data sets provided are whole numbers, the results of transformations will almost always not be whole numbers, so the majority of the computations in the program must be done in floating point formats (double precision most likely). Usually, graphics functions require coordinates of pixels as integers, so the floating point values generated by your calculations will have to be truncated to integers when drawing occurs. The program skeletons provided include all drawing logic, so you needn't worry about this.

(v) It appears that the program must respond to many distinct "commands", but you should notice that there are really only six different types of operations involved, of which only four involve transformations. You can organize your program so that effectively each of these four types of transformations is handled by a single function or event handler. For those manipulations which involve compound transformations, only the final transformed character should be displayed, of course. If you are creating your own program entirely from scratch, the various "commands" must be invokable by a single mouse click (as with a toolbutton or speedbutton) rather than via menu choices.

(vi) The program skeleton is built around three main arrays: one containing the initial coordinates of the points, one containing a cumulative transformation matrix, and one to hold the transformed points for rendering the character (making use of the unchanging information in the array containing the information from the lines file). Your code should not be making any changes at all to the two matrices of coordinates, but only to the cumulative transformation matrix.

(vii) The appearance of continuous rotation can be simulated with a loop generating successive individual rotations by some small angle, say 0.05 or 0.1 radians each. For rotations, you'll need to use the *sin()* and *cos()* functions. Recall that these expect angles expressed in radians, so there's no real advantage in introducing degrees into the program. Once set in motion, the rotation should continue until a user-initiated event occurs.

Evaluation and Deadlines

Assignments must be demonstrated to a laboratory instructor on the dates of the checkpoints. Except in unusual circumstances, you are expected to demonstrate your program no later than the tutorial session that week scheduled for your set. Also, a zip file of the various files in your project must be submitted to the COMP 4560 course folder on the In: drive by the end of that lab period. (Use a name of the form *LastnameFirstname.setnum.asgn5.zip.* Include only those files necessary to regenerate the project – omit temporary files created during the build or compile process.).

I will also be checking your works in progress on the following dates:

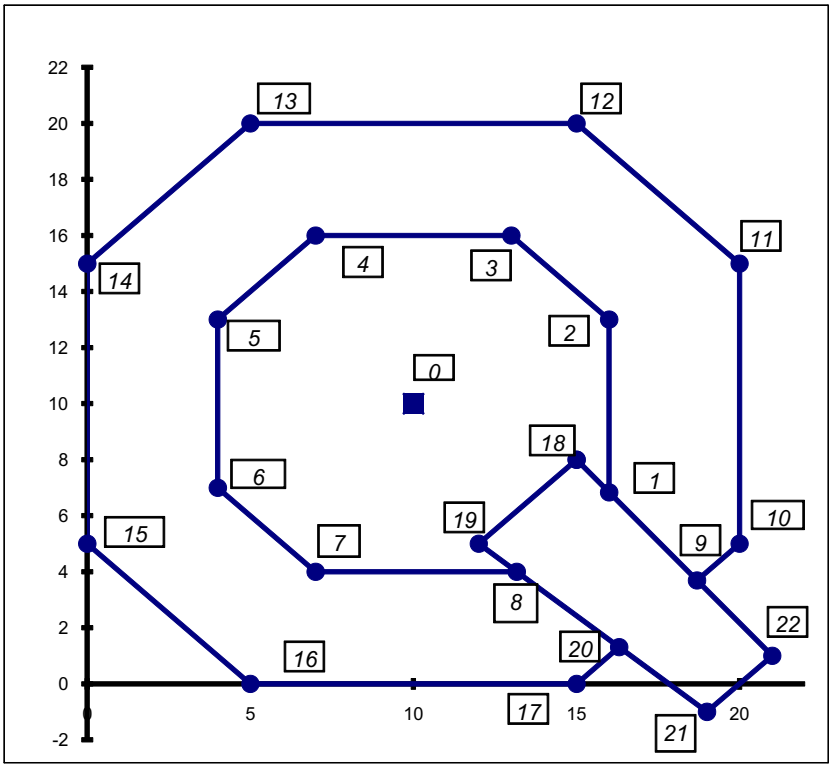
Week of February 27: shape initializes correctly
Week of March 6 : translation, scaling, and rotation by 0.05 radians
Week of March 20: the finished product (shear, and continuous rotation)

The marks for this assignment are the sum of two assessments: (i) 30 marks based on a final demonstration that your program has all of the functionality described above and that you have submitted the zip file as described, and, (ii) 10 marks for a convincing explanation of the implementation of one or more of the sets of event handlers in your program. **Since this is a major assignment, it is worth approximately half your total assignment mark**.

When you demonstrate your program for marking, we will use a data file which conforms to the principles stated above, but yields a shape different from the shapes in the "letter Q" files provided for your use during the program development stage. Thus, you need to make sure that your program does not make use of any features of the "letter Q" shape that might not apply to any other three-dimensional shape.

COMP 4560: Assignment #5 (February, 2008)

Block Character Design Example



Point	x	y	z
0	10	10	0
1	16	6.833	0
2	16	13	0
3	13	16	0
4	7	16	0
5	4	13	0
6	4	7	0
7	7	4	0
8	13.167	4	0
9	18.692	3.692	0
10	20	5	0
11	20	15	0
12	15	20	0
13	5	20	0
14	0	15	0
15	0	5	0
16	5	0	0
17	15	0	0
18	15	8	0
19	12	5	0
20	16.308	1.308	0
21	19	-1	0
22	21	1	0

QPOINTS.DAT

Lines	
1	2
2	3
3	4
4	5
5	6
6	7
7	8
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	20
19	21
21	22
22	18
18	19

QLINES.DAT