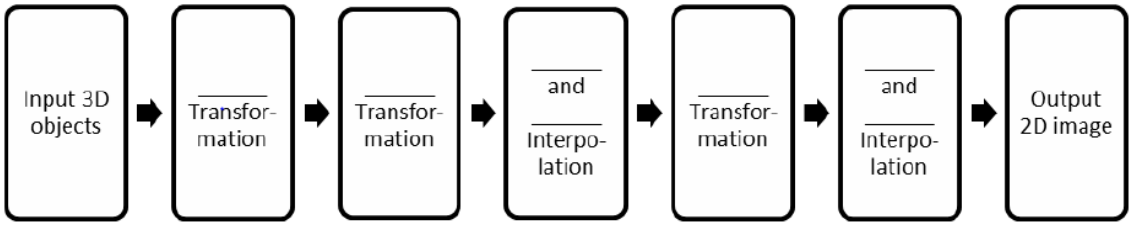
CS 112: Introduction to Computer Graphics (Winter 2020)
Written Assignment 1 - Solution

1) Fill the blanks to complete the graphics rendering pipeline. [7]

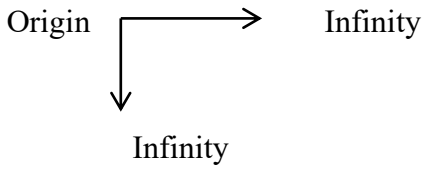


Input 3D object goes through the following transformations.

a. ModelView Transformation
b. Projection Transformation
c. Window Coordinate Transformation
d. Clipping and Vertex Interpolation
e. Rasterization and Pixel Interpolation.

This creates the output image.

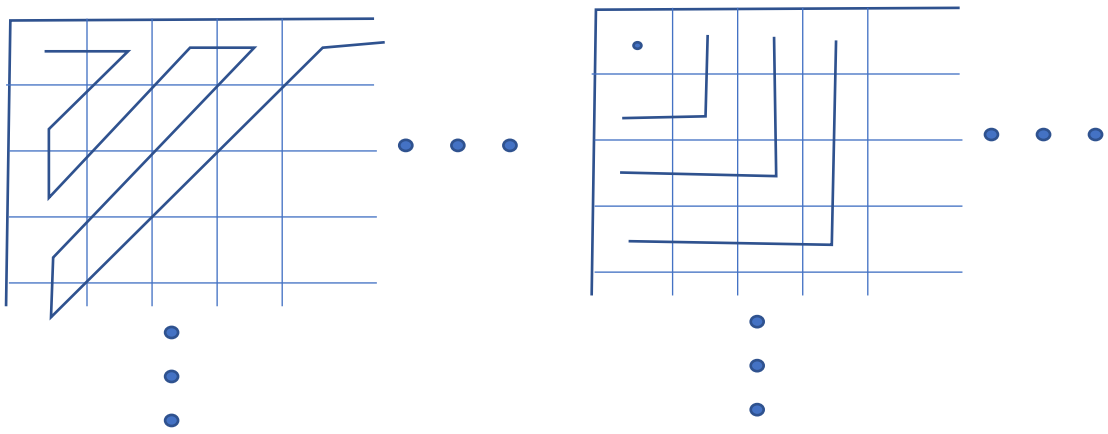
2) Many image files have linear structure. If the size of the image is 1000x1000 (WxH), then the element at position 3500 in the file would refer to the element in the fourth row and 500th column of the image. If the size of the image is infinite in both width and height, but has an origin (refer to the figure), then how would your organize your file into a linear structure? In other words, come up with a scheme of organizing the data in the file such that given the position of the element in the file, you can locate the corresponding position in the image. [4]



We need to visit pixels in an order that will assure two things.

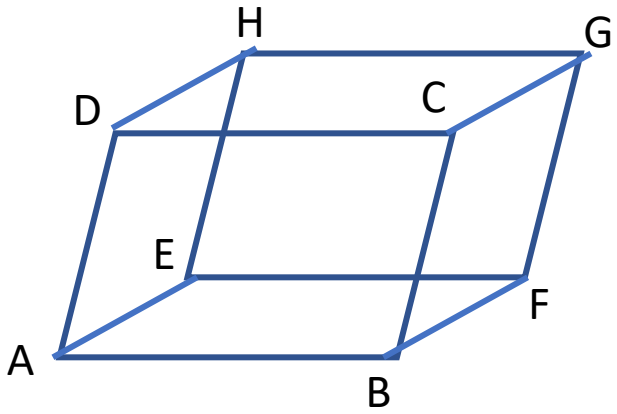
a. All the pixels will be visited
b. The indices should not contain any references to the width and height (this is the limitation of the scan-line based traversal)

Any zigzag traversal as shown below would assure this.



3) Consider a parallelepiped. A parallelepiped is a genus zero object with eight degree-three vertices and six parallelogram faces. (It is a sheared cuboid with no constraints on angle between the edges at a vertex.) Consider a 4x4 rigid model transformation matrix M. Naïve transformation of one vertex by M takes 16 multiplications and 12 additions/subtractions (4x4 matrix multiplied with a 4x1 vector). Hence for eight vertices, it would take 16x8 multiplications and 12x8 additions/subtractions. By utilizing characteristics of a rigid transformation, 16x___ multiplications and 12x___ additions/subtractions are enough to locate the transformed parallelepiped. [2+2=4]

Using rigid body transformation, you will need to only transform A, B, D and E which would take 4x16=64 multiplications and 4x12=48 additions. We will need to compute the lengths of AB, AE and AD which would be each 3x4=12 subtractions. To find the rest of the 4 points, we have to add these lengths to the appropriate one of A, B, D and E. Each of these will take 4 additions. Therefore, total of 16 additions. Therefore, we will need total of 64 multiplication and 76 additions. You may get slightly different answers based on the procedure you follow. As long as the concept is right, you have got it.

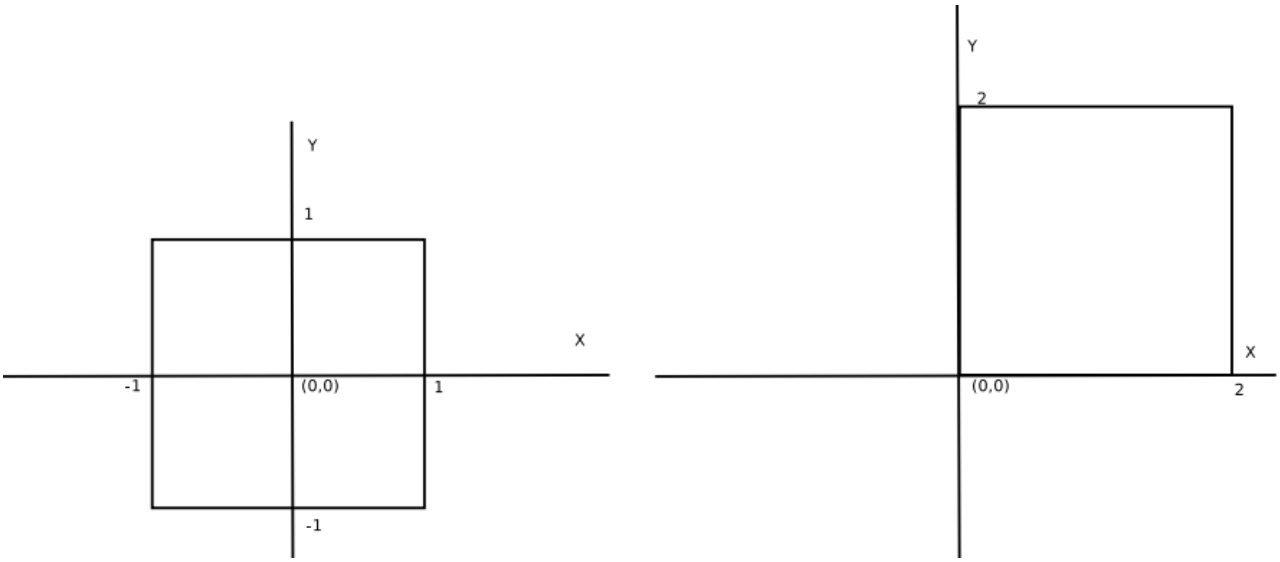


4) One reason that homogeneous coordinates are attractive is that the 3D points at infinity in Cartesian coordinates can be explicitly represented by homogeneous coordinates. How can this be done? [3]

This will be done by using w=0 in the homogeneous coordinate. This represents only a direction which is equivalent to point at infinity.

5) Consider a 2D square on the XY plane with side 2 units, the center at the origin and four sides parallel or perpendicular to the coordinate axes. Draw the picture of the transformed square after performing the following sequence of OpenGL commands. (Remember OpenGL post-multiplies the matrices in the order it is received, and finally the point is also post-multiplied.) (1.414 is the approximation of sqrt(2).)

glRotatef(45,0,0,1);
glTranslatef(1.414,0,0);
glRotatef(45,0,0,1);

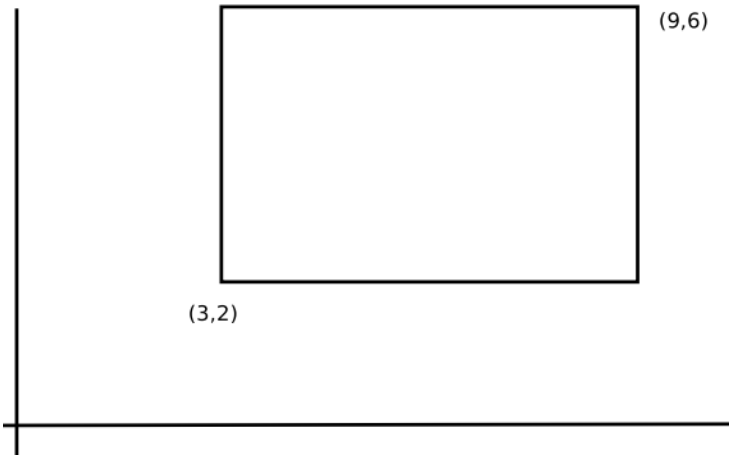


Reduce the number of OpenGL function calls and thus give the new sequence of OpenGL function calls to effect the same transformation. [3+4=7]

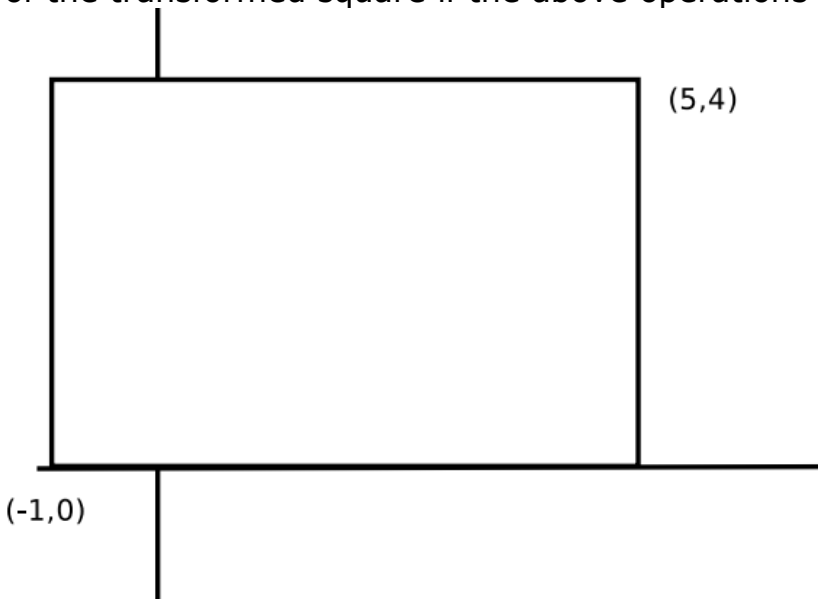
glTranslatef(1, 1, 0)
glRotatef(90, 0, 0, 1)

6) Consider the same square as in Question 1 at the initial position. Draw the picture of the transformed square after performing the following sequence of OpenGL operations. [3+3+4=10]
Case 1:

glScalef(3,2,1);
glTranslatef(2,2,0);



Case 2:
Draw the picture of the transformed square if the above operations were swapped.



If we swap the function calls as in Case-2, but still want the output of Case-1, what parameters should we pass to the two commands?

```
glTranslatef(6,4,0);
glScalef(3,2,0);
```

7) The inverse R^{-1} of a rotation matrix R is its transpose R^T. For R, the inter-relationship between different row vectors is **orthonormal**, and the inter-relationship between different column vectors is **orthonormal**. [2+2=4]

8) In 3D, show $R_z(\theta 1).\ R_z(\theta 2) = R_z(\theta 2).\ R_z(\theta 1)$. What does this tell about the properties of rotation around coordinate axes? Show that $R_z(\theta 1 + \theta 2) = R_z(\theta 1).\ R_z(\theta 2)$. Using this property show that rotation about

any arbitrary axis denoted by R_a also follows the property, $R_a(\theta 1).R_a(\theta 2) = R_a(\theta 2).R_a(\theta 1) = R_a(\theta 1 + \theta 2)$. [10]

9) Given notation a_{ij} means the entry of a matrix at *i*th row and *j*th column, the scaling matrix for scaling an object by a scale factor 3 along an arbitrary direction given by vector *u = (1, 2, 1)* rooted at *(5, 5, 5)* will be:

***T* (5, 5, 5) R^T *SRT* (-5, -5, -5)**

10) A viewer is defined by the following. (a) Eye position: *(0, 0, 0),* (b) View Up Vector: *(0, 2, 0),* (c) Equation of the image plane*:* $x+y+z = 6$. Find the matrix that will be generated by the function call **gluLookAt.** Let the left, right, top and bottom planes be at -2, +2, 4, and 8 respectively. Let the far plane be at 10. Find the perspective projection matrix given by the function call **glFrustum**. Find what would be projected coordinates of a point *P = (10, 4, 6)* for this viewer. [10+10+2=22]

The first task is to find the "center" position, i.e. the projection of the eye point (0, 0, 0) onto the image plane. The unnormalized plane normal is (1, 1, 1). It's easy to infer that the projection of (0, 0, 0) onto the plane has the form (0, 0, 0) + *k* (1, 1, 1), and it's exactly (2, 2, 2). As expected, plugging these coordinates into the equation makes it hold true: 2 + 2 + 2 = 6. Now we know that the near clipping plane is at distance $\sqrt{(2^2 + 2^2 + 2^2)} = \sqrt{12} = 2\sqrt{3}$. This will be useful for the second part of the question. We now have the eye point (0, 0, 0), the center point (2, 2, 2), and the up vector (0, 2, 0), which normalized is (0, 1, 0).

We let *f* be the normalized vector (center-eye), or $f = (1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})$.
$s = (f \times up)/|f \times up|$ and $u = s \times f$. We construct matrix M as:

$$M = \begin{pmatrix} s0 & s1 & s2 & 0 \\ u0 & u1 & u2 & 0 \\ -f0 & -f1 & -f2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \\ -0.40825 & -0.8165 & -0.40825 & 0 \\ \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

For the glFrustum, we know n, f, l, r, t and b, which is all we need to compute the matrix.

So we have:
A = (right + left)/(right - left)
B = (top + bottom)/(top - bottom)
C = -(far + near)/(far - near)
D = -(2 × far × near)/(far - near)
E = (2 × near)/(right - left)
F = (2 × near)/(top - bottom)

And the matrix P is:

$$P = \begin{pmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1.73205 & 0 & 0 & 0 \\ 0 & -1.73205 & -3 & 0 \\ 0 & 0 & -2.06002 & -10.60023 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Finally, in order to compute the projected coordinates of $p = (10, 4, 6)^T$, we simply multiply $p_{pr} = P{\cdot}M{\cdot}p$, and the result should be:
$(-4.8990, 40.2979, 13.1869, 12.5470)^T = (-0.39045, 3.21175, 1.051, 1)^T$

11) The model transformation for our scene is a rotation R about the Y axis in the counter clockwise direction by 90 degrees, followed by a translation T in the positive X direction by 20 units. What is the resulting transformation? [3+3=6]

The transformation is R followed by T i.e. M = TR, where

$$R = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 0 & 0 & 20 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

12) Choose the correct answer: If V is a vertex in our scene mentioned in Question 8, then after model transformation, the transformed vertex is computed as (a) RxTxV **(b) TxRxV**.[4]

13) The view transformation for our scene is the identity matrix. What is the position and orientation of the OpenGL camera? [5]

It means that the camera is at the origin, looking towards the negative Z axis, with the up vector along the positive Y axis.

14) Consider three vertices A, B and C. Choose the normal vector of face ABC? [2]

a. |AB| * |AC|
b. (B - A) × (C - A)
c. (B - A) · (C - A)
d. ((B - A) + (C - B) + (A - C)) / 3

15) When is the vertex illumination computed in the OpenGL pipeline? _____ (a) **Between the model-view and the projection transformations**, or (b) After the projection transformation, before clipping. [2]

16) OpenGL can be instructed to cull (avoid rendering) triangles that are facing away from the viewer. Given viewing direction V and a triangle ABC both in world coordinates, assuming orthographic projection, how can you tell whether ABC is facing the viewer or not? What additional information do you need to compute the orientation of the triangle if we use perspective projection? [4+3=7]

In orthographic projection, the view direction V is always the same. Therefore, a dot product with the face's normal would be enough to cull the face when $N \cdot V > 0$. For perspective projection, different pixels in the screen use different view directions. Therefore, we would need a point P in the triangle to compute the view direction from the eye E and test whether $N \cdot (P - E) > 0$.

17) You change the normal vectors of an OpenGL triangle, but don't change the position of the vertices. Which components of the color seen by the viewer might change? _____[2]

a. Ambient
b. Diffuse
c. Specular

18) Definition: Silhouette edges are the edges in the manifold that have one back-facing polygon AND one front facing polygon incident on it.

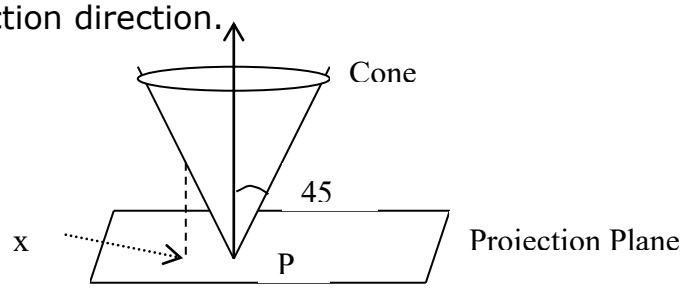
(1) How do you compute the silhouette edges of a manifold?
For each edge in the mesh, check whether one adjacent triangle is front-facing and the other is back-facing. Then it is a silhouette. Otherwise it is not.

(2) In OpenGL you can draw only back-facing polygons, or only front facing polygons. If you render the manifold(front facing polygons), then clear the frame-buffer but not the depth buffer, then again render only the back facing polygons. What do expect to see?
Since the depth values from the first rendering are still in the depth buffer, no pixels should appear. However, depending on the particular depth-test function being used, a very thin, and perhaps intermittent, silhouette might appear.

(3) Assume that the "thickness" of a line is an attribute of a line. Thickness of three means that the line would be drawn three pixels "thick". In question (2), if the thickness of the line was one and now is increased to three only for the second rendering (rendering of back faces), what do you expect to see? [5+2+3=10]

The added pixel on either side of the lines used to render the triangles should make the silhouette edges display one pixel in the outer side. In other words, the silhouette of the model would appear.

19) Consider orthographic projection. The projection plane is perpendicular to the projection direction.

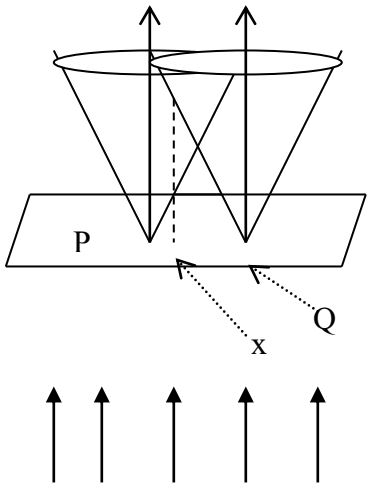


Projection Direction

The surface of the cone makes an angle 45 degrees with the axis of the cone. The axis is parallel to the projection direction, and the apex (P) is on the projection plane. Assume that you are drawing the cone (and not the projection plane). What will be the depth value at any arbitrary point x on the plane? Express it in terms of distance(x, P), which is the distance between point x and P. [5]

z=distance(x,P)

20) Choose another point Q on the plane, and construct a similar cone as the one resting on P. The cone resting on P is colored red, and the one resting on Q is colored green. After projecting these two cones, the pixels on the projection plane would get the color of the point on the cone that is closer to the projection plane. For example, the vertical line at point x intersects the cone at P first and hence point x would get the red color. Use your answer to Question 12 and show that for each pixel which color should be assigned to it. [5]



Proiection Direction

Pixels will receive two colors, with different depth values. The lower depth value gets rendered. Since the depth (Z) value is the distance from the pixel to the apex of the cone, each pixel will get the color of the cone to which it is closer.

21) Certain region of pixels in the projection plane would get red color and certain region of pixels would get the green color. Interpret the boundary of the regions with red color and green color using your answer to Question 13. What will be the shape of the curve of this boundary (straight line, circle, ellipse, etc.)? [5]

It is a straight line.

22) Rasterize the line (P1, P2) where P1= (2,5), and P2=(8,15). Find the coordinates and the color of each pixel rasterized by this line segment, given the color of P1 is 0.8 and that of P2 is 0.1. Also show that the center of the pixel that is rasterized by this line is at most at a distance 0.5 from the actual line. [15+10+5 = 30]

The line has slope > 0, so the X and Y coordinates need to be swapped, then we'll rasterize the swapped coordinates (5,2) and (15,8), and finally swap the coordinates back. The rasterized pixels are: (2,5), (3,6), (4,8), (4,9), (5,10), (5,11), (6,12), (6,13), (7,14), (8,15).

The color decreases 7/11 units (0.063) each step from P1 till P2. So, the values are: .80, .73, .66, .59, .52, .45, .38, .31, .24, .17, .10. In order to do the proof, check the distance from each pixel to the closest point in the explicit line equation, and make sure the limits are fine.

23) Draw the results of clipping of a triangle ABC defined by A=(500,100), B=(800,460) and C=(400,500) against a window whose x_{min} = 300, x_{max}=700, y_{min}=200 and y_{max}=500, using Sutherland Hodgeman's method. Show the vertices remained in the window (including the ones newly created by clipping) for all the steps of the pipeline clearly. It does not matter if you do it clock-wisely or counter clock-wisely. [20]

Clip against the lines in the following order: Top, right, bottom, left. Clipping against the top line doesn't change the situation. The right line clips vertex B and produces vertices AB(700,340) and BC(700,470). The bottom line removes vertex A and introduces AC(475,200) and AAB(583.33,200).
Again, clipping against the left line doesn't change anything.