Midterm Exam

Fundamentals of Computer Graphics (COMP 557)
Thurs. Feb. 19, 2015
Professor Michael Langer

The exam consists of 10 questions.
There are 2 points per question for a total of 20 points.

You may use this cover sheet if you need more space
(or ask the prof/invigilator for another exam).

LASTNAME, FIRSTNAME: ______________________________

STUDENT ID: ______________________________

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Q8	
Q9	
Q10	
TOTAL	

1. Consider a 2D space (x, y) whose points are represented using homogeneous coordinates. Give a product of matrices that performs the following. Rotate the scene by θ degrees clockwise around the point $(x, y) = (2, 3)$. That is, the given point stays fixed and all other points are moved.

<u>**SOLUTION:**</u>

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

<u>**GRADING SCHEME:**</u>

The question said ”clockwise”, whereas the examples I gave in the lectures were always counterclockwise. That’s the reason why the sign on the ”sines” is different from what you might have expected. (This was not intended to be a trick. Rather, I meant to write ”counterclockwise”.) We ignored the sign reversals on the ”sine” functions in the solution. However, we didn’t ignore the signs on the ”cosine” functions! The latter should be positive and we took off a point if you wrote them as negative.

Some of you switched the ordering of the translation matrices. We gave only 1.5/2 if you did that.

Some of you only had one of the translations. We gave only 1/2 if you did that.

2. Consider the same situation as Q1, but now give a product of matrices that maps a rectangle having opposite corners $(1, -3)$ and $(3, 3)$ to a rectangle having opposite corners $(2, 0)$ and $(10, 12)$.

SOLUTION:

There were several solutions possible.

The first solution is to shift the bottom left corner to the origin, then scale so that the rectangle has the desired final dimensions, then translate the bottom left corner to its desired final location.

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

The second solution that many people gave was first to translate by $(-2, 0)$. This moves the center of the rectangle to the origin. Then scale by the correct amount. Then translate by $(6, 6)$ which brings the center of the rectangle to the correct position i.e. $(6, 6)$.

$$\begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A third solution we saw was to scale by the correct amount, and then translate in a way that brings one of the given corners (which has been moved because of the scaling) to its desired position.

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A fourth solution was the most difficult to come up with: translate first and then scale. To figure out the translation amount, apply the inverse of the scaling on the two final corner vertices, which maps these vertices back to (0.5, 0) and (2.5, 6), respectively. Then figure out the translation required to bring the original corner points to these positions. That translation is $(0.5, 3)$, giving:

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -0.5 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

GRADING SCHEME:

As in Q1, some people switched the ordering of the translation matrices. We gave only 1.5/2 if you did that, provided that the matrices were correct.

Some people had the idea of combining a translation and a scaling but gave numbers that were calculated incorrectly. We had to make a judgment call in giving points here, since it was typically unclear where the numbers came from. A typical grade in this case was 1/2.

Many people wrote their matrices as 4×4 for this question (and for Q1). No marks were taken off for that.

3. Describe how you would obtain a 4×4 projection matrix that maps $\Re^3$ to the plane

$$3x + 2y + z = 1.$$

Assume the center of projection *i.e.* eye is (0,0,0).

You do not need to provide numerical values for the matrix (or matrices), but be sure to specify how you *would* obtain such values.

SOLUTION:

There were two general approaches to the problem. The first is the one I had in mind. Let $\mathbf{R}$ be a rotation that maps the plane normal $(3, 2, 1)$ to the z axis. Such a rotation matrix was given lecture 2 page 4, though in that case that the point $\mathbf{p}$ was a unit vector and was mapped to the unit vector $\hat{\mathbf{z}}$. But its the same idea here. Just let $\mathbf{p}$ be the normalized $(3, 2, 1)$ vector. Then, let $\mathbf{M}$ be a perspective projection onto the $z = \sqrt{3^2 + 2^2 + 1}$ plane (lecture 4 page 2.) Finally, rotate back. Thus the solution is

$$\mathbf{R^T M R}.$$

The second approach was to consider the ray from the origin through point (tx, ty, tz) and ask where it hits the given plane. This is a fine way to think about it and indeed its how we derived the projection matrix in class. The trouble with this approach is that it is trickier to convert that to the solution that I asked for, namely to give a 4×4 projection matrix. But here's how you would do it (and some students did!). The key is to note that the point of intersection is

$$3tx + 2ty + tz = 1$$

and so $t = \frac{1}{3x+2y+z}$ and the point of intersection is $(\frac{x}{3x+2y+z}, \frac{y}{3x+2y+z}, \frac{z}{3x+2y+z})$ which can be written in homogenous coordinates as

$$(x, y, z, 3x + 2y + z)$$

and so the projection matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$

GRADING SCHEME:

For the first solution we gave 1.5/2 if you forgot to rotate back.

For the second solution, we gave part marks if you found the point of intersection but didn't provide the matrix. The number of part marks was a judgment call.

One final comment: The question mentions center of projection but it did not explicitly say "perspective projection". Some students used orthographic projection instead. We did not penalize for this mistake since the question could have been more explicit.

4. Consider the projective transformation:

$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & f_0 + f_1 & -f_0 f_1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

(a) Which points in $\Re^3$ get mapped to points at infinity?

(b) Which points at infinity get mapped to points in $\Re^3$?

SOLUTION:

(a) The plane $z = 0$ gets mapped to infinity. To see this, multiply the matrix by $(x, y, z, 1)$ and examine the 4th coordinate. For the point to be mapped to infinity, the 4th coordinate should be 0, namely $0x + 0y + 1z = 0$, or $z = 0$. Thus, points on the plane $z = 0$ get mapped to points at infinity.

(b) A general point at infinity $(x, y, z, 0)$ gets mapped to $(f_0 x,\ f_0 y,\ (f_0 + f_1)z,\ z)$. By inspection, all points at infinity get mapped to $\Re^3$, except those points at infinity for which $z = 0$. The points get mapped to other points at infinity.

GRADING SCHEME:

For (a) we gave either 0/1 or 1/1.

For (b), we gave 0.5/1 if you wrote all points at infinity get mapped to $\Re^3$, but neglected $z = 0$.

5. How could bounding volumes be used to speed up clipping? At what stage of the pipeline should this clipping occur? Justify your answer.

SOLUTION:

The bounding volume would be tested against the clipping planes, i.e. the boundaries of the view volume. If the bounding volume lies outside the view volume, then all the surfaces inside the bounding volume must lie outside the view volume too (trivial rejection). If the bounding volume lies entirely within the view volume, then all surfaces must lie in the view volume too (trivially accept). If the bounding volume partly intersects the view volume, then there's not much you can say and you would need to expand the bounding volume (to sub-bounding volumes or to surfaces).

As for when this clipping should be done... here are two arguments that are both good.

On the one hand, if the entire bounding volume can be rejected/clipped, then this should be done as early as possible, namely before the objects in the bounding volume vertices are sent into the pipeline. i.e. it can be done by CPU. That way, you don't send the complex object through the pipeline if you don't need to. (Also, note the bounding volume is chosen to have a simple shape. The projective mapping would likely make this a more complex shape.)

On the other hand, if you could argue that this check *should* be done at the usual place where clipping is done. If the bounding volume were entirely in the view volume, then all surfaces within the bounding volume would be entirely within the view volume as well. Thus, if you can trivially accept the whole bounding volume, then you can trivially accept all of its surfaces too.

One might even argue that a combination of both would be best. Do the checking for trivial rejection before the pipeline (in CPU) and do the checking for trivial acceptance at the clipping stage.

GRADING SCHEME:

Since this was an open book exam, we didn't give any points to you if you just stated what bounding volumes are an how they are used in *ray casting*. Why not? Because the question isn't about ray casting, but rather it is about clipping. We gave 1/1 if you explained that the bounding volume (not just a vertex!) needs to be tested against the clipping planes i.e. view volume boundaries, and you argued about trivial acceptance or rejection of the bounding volume.

For the pipeline location, if you just said essentially that clipping happens at the clipping stage but you gave no further explanation, then we gave 0/1 only – unless your answer in the first part indicated that you understood.

Several of you said clipping should be done at the vertex stage of the pipeline. We generally gave the point for this if the argument made sense. However, note that the vertex stage typically refers to the coordinate transformations only. It is the first stage of processing on the graphics card (GPU), but alot happens before the vertex stage, namely the CPU can do alot of work before sending anything to the graphics card.

6. Claim: "If a scene contains quadric surfaces, then the depth buffer method (hidden surface removal) can only be applied if these quadric surfaces are first discretized into polygons." Is this claim true or false? Briefly explain.

   SOLUTION:

   False. The depth buffer method just needs to test, for each pixel, whether the object is visible at that pixel. For quadric surfaces, this test can be done for each pixel by solving a quadratic equation. You don't need to approximate the quadric with polygons.

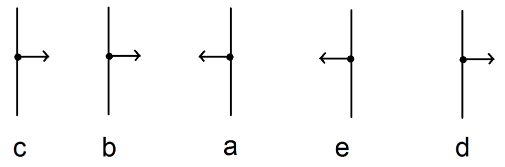
   GRADING SCHEME:

   If you just quoted the algorithm in the lecture notes where it says to loop over each polygon, then you would answer TRUE. However, the algorithm works fine if you change it to object instead of polygon. There is nothing about the algorithm that requires it to be polygon. As long as you can figure out if a pixel lies in the projection of the object, then the algorithm works fine. That is certainly the case with quadrics, as you can see from the ray casting lecture where I showed how to intersect a ray with a quadric.

   Most of you got either 0/2 or 2/2 on this one. There wasn't much middle ground in the answers, and explanations were typically very brief.

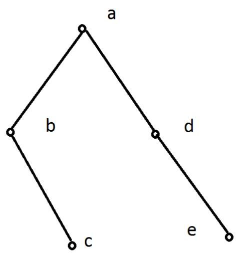
7. (a) Construct a BSP tree for a 2D scene below. For each subtree, choose the edge from the list in alphabetical order. In particular, the root of the tree is edge `a`.

   (b) What is the order of edges drawn for a viewer that is located between edges `a` and `b` ? (Ignore back face culling.)



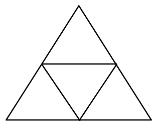
SOLUTION:

(a)



(b) The order of edges drawn is d, e, a, c, b.

8. Consider an equilateral triangle. Partition it into four equilateral subtriangles as sketched below.



Then delete the central subtriangle leaving three subtriangles. Repeating this recursively, infinitely many times, gives a fractal.

(a) Show that the surface area of this fractal is 0.

(b) Give an expression for the dimension of this fractal. Hint: $C = S^D$

SOLUTION:

(a) Assume that the original triangle has unit area. Then removing the central triangle leaves area $\frac{3}{4}$. Removing the central triangle of each of these leaves area $(\frac{3}{4})^2$. After n levels, area is $(\frac{3}{4})^n$ which goes to 0 as $n \to \infty$.

(b) C = 3, S = 2, so D = $\log 3 / \log 2$.

9. Suppose you would like to fit a cubic curve $\mathbf{p}(t) = (x(t), y(t), z(t))$ to two given 3D points $\mathbf{p}(0)$ and $\mathbf{p}(1)$ and suppose $\mathbf{p}'(1)$ and $\mathbf{p}''(1)$ are also given. That is, you are given the positions at $t = 0$ and 1 and you are given the first and second derivatives at $t = 1$.

   Show how to find the coefficients of this cubic curve.

SOLUTION:

$$\mathbf{p}(t) = \; = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

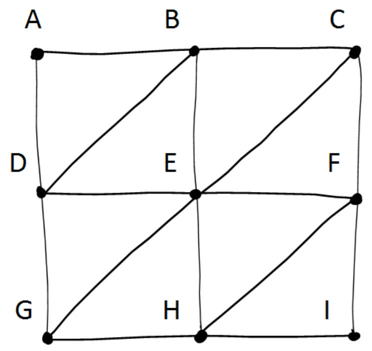
Taking derivatives with respect to t and substituting $t = 0$ and $t = 1$ gives:

$$[\, \mathbf{p}(0) \quad \mathbf{p}(1) \quad \mathbf{p}'(0) \quad \mathbf{p}''(0) \,] = \; = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} 0 & 1 & 3 & 6 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

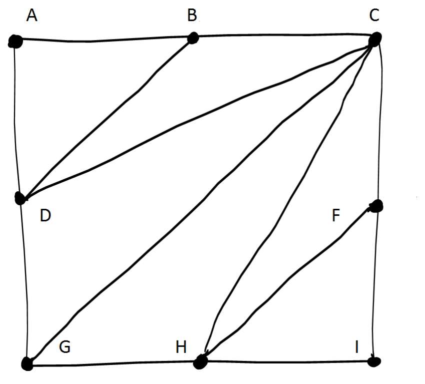
Right multiplying by the inverse of the 4×4 matrix on the right side gives the coefficients.

10. Simplify the mesh below by collapsing the edge EC onto vertex C. Draw the simplified mesh.

    Is it possible to have more edge collapses and also to maintain the overall square shape? If so, which edges can be collapsed (and onto which vertices) ?



SOLUTION:



One vertex, one edge, and two faces are lost.

Yes, AB can be collapsed to A (and D can be collapsed to A). Similarly, H can be collapsed to I (and F can be collapsed to I).