

EECS 487 Homework Set 1

10 problems totaling 100 points.

Due: 21 October 2009, 1:40 pm, in lecture

You are allowed to consult both online and offline sources, including humans, to help solve these problems. If you do consult outside sources, i.e., other than yourself and the teaching staff, you **MUST** cite them. You do not need to cite the teaching staff nor the textbooks nor the lecture notes. These are the only exceptions. What you turn in must be your individual work. Your classmates can give you an idea on how to approach a problem, but they cannot give you any solution to these problems. If you find an online solution to any of these problems, you are allowed to consult them, but not to use them verbatim. You must phrase your solution in such a way that shows you have understood the problem and solution. Violation of any part of the above policy will be a violation of the Honor Code. If you turn in a handwritten solution, please write legibly. Illegible scribble will earn zero points.

- 1.) **Antialiasing.** (5 pts) In the midpoint algorithm for line drawing (consider only the case where $0 \leq m \leq 1$ for this problem) each iteration has to make the decision whether to increment y ; one must choose whether to

plotPixel(x, y, r, g, b, a)

or

plotPixel(x, y + 1, r, g, b, a).

The line can be automatically antialiased by turning on both of these pixels with correct alpha values. Explain in a few sentences how the alpha values for the two pixels can be chosen.

- 2.) **Ellipse Rasterization.** (15 pts) Derive a version of the midpoint algorithm to rasterize an ellipse of the form

$$\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$$

which is an ellipse with center (h, k) and axes a and b (the larger of the two is the major axis and the other is the minor) as the function

drawEllipse(int h, int k, int a, int b).

Be sure to provide an algorithm that uses integers only. Do not simply write it down, but show a clear understanding of how it is reached. You are allowed to assume that the function

plotPixel(x, y)

exists and you do not have to worry about coloring the ellipse.

Hint/Advice: Use symmetry to simplify the algorithm; only perform calculations in one quadrant (the first is ideal). Be careful to check whether the x - or y -axis

is the major axis. Does the algorithm iterate over x , y , or a little of each? One is encouraged to implement the algorithm to test if it works (if interested just alter the display callback function of *lab2*).

- 3.) **Clipping.** (5 pts) Given the viewport (the portion of the *world* visible)

$$|x| + |y| \leq 1$$

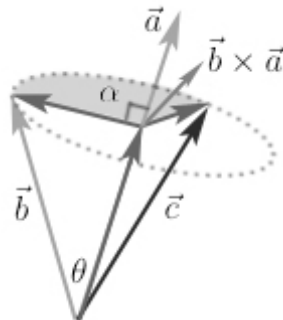
find the clipped endpoints of the line segment \overrightarrow{PQ} with $P = (-3, -2)$ and $Q = (5, 1)$.

- 4.) **Rotation of a vector around an arbitrary axis.** Given a vector \vec{b} , an axis \vec{a} , and an angle α , find a formula (or function) that produces the vector \vec{c} created by rotating \vec{b} an angle α around \vec{a} counterclockwise (in a right-handed system). Find an expression for \vec{c} in the following two ways:

i.) **Matrix Rotations (crunching it out).** (10 pts) Assume the tails of both \vec{a} and \vec{b} are at the origin. Find the matrix that rotates the axis \vec{a} around the z -axis into the xz -plane (you will need to find the angle needed and plug it into $\hat{R}_z(\theta)$) and apply that matrix to \vec{b} . Then find the matrix that rotates the new axis in the xz -plane around the y -axis to coincide with the z -axis, that is find ϕ for $\hat{R}_y(\phi)$, and apply that matrix to \vec{b} after the first matrix. Now apply a rotation by α around the z -axis and undo (apply the inverses in the correct order) the first two transformations. Thus $\vec{c} = \hat{R}_z^{-1}(\theta)\hat{R}_y^{-1}(\phi)\hat{R}_z(\alpha)\hat{R}_y(\phi)\hat{R}_z(\theta)\vec{b}$. Remember that $\hat{R}(-\beta) = \hat{R}^{-1}(\beta)$.

ii.) **Clever Projection (thinking it through).** (10 pts) In rotating \vec{b} around \vec{a} one can decompose \vec{b} into two pieces, one that stays the same ($\text{proj}_{\vec{a}}\vec{b}$) and one that rotates ($\text{proj}_{\vec{a}^\perp}\vec{b}$). Now find two orthogonal vectors \vec{u} and \vec{v} that form the plane that $\text{proj}_{\vec{a}^\perp}\vec{b}$ should rotate in but make sure that they have the same magnitude as $\text{proj}_{\vec{a}^\perp}\vec{b}$. Then the new piece will be of the form $\cos(\alpha)\vec{u} + \sin(\alpha)\vec{v}$.

Hint/Advice: One of the two vectors used to rotate $\text{proj}_{\vec{a}^\perp}\vec{b}$ can be $\text{proj}_{\vec{a}^\perp}\vec{b}$ itself and the other can be parallel to $\vec{b} \times \vec{a}$. The following diagram may help:



5.) **Barycentric Coordinates.** Two methods for computing barycentric coordinates were discussed in lecture.

- i.) (5 pts) Describe these two ways in your own words (do not copy verbatim from the lecture notes).
- ii.) (10 pts) Show or explain whether and when one of the two ways is computationally cheaper than the other.

Hint/Advice: Use operation count to determine the computation cost of the two methods.

6.) **Perspective transformation.** Assume the viewer is at the origin looking down the positive x -axis with the y -axis pointing upward and the z -axis pointing to the right. Let n be the distance to the near plane, f be the distance to the far plane, l to the left, r to the right, t to the top, and b to the bottom plane.

- i.) (5 pts) Derive the perspective projection matrix for the above transformation.
- ii.) (5 pts) Given the parallelogram

$$(-1, 1, -1), (-1, 0, 0), (-1, 0, 1), (-1, 1, 0)$$

that is *behind* the viewer, show what it maps to after the transformation and draw a picture with before and after.

- iii.) (5 pts) Prove that the transformation preserves the ordering of z values.
- iv.) (5 pts) Show that a perspective transformation maps lines to lines.

Hint/Advice: Given a line segment $\vec{p} + t(\vec{q} - \vec{p})$ with $t \in [0, 1]$, the projection matrix (M_p) will produce a segment

$$M_p \vec{p} + t(M_p \vec{q} - M_p \vec{p}) \equiv \vec{p}' + t(\vec{q}' - \vec{p}').$$

The homogenized segment is then

$$\frac{\vec{p}' + t(\vec{q}' - \vec{p}')}{h_{p'} + t(h_{q'} - h_{p'})}$$

where $\vec{p}' = (p'_1, p'_2, p'_3, h_{p'})^T$ and $\vec{q}' = (q'_1, q'_2, q'_3, h_{q'})^T$. Now find a function $f(t)$ such that the above can be written

$$\frac{\vec{p}'}{h_{p'}} + f(t) \left(\frac{\vec{q}'}{h_{q'}} - \frac{\vec{p}'}{h_{p'}} \right).$$

7.) **Rotations.** The orientation of any system has three degrees of freedom (three independent values will specify it). For instance an orientation can be specified with *pitch*, *yaw*, and *roll*, or it can be specified by rotation angles α , β , and γ .

- i.) (5 pts) Another way to specify an orientation is by specifying a vector (or axis) \vec{a} and an angle θ to rotate around that vector. The axis has three components and the angle is a fourth. Explain how this system really only has three independent values (degrees of freedom).

- ii.) (5 pts) Another way to specify an orientation is to give a vector that points forward, one that points up, and one that points right. Three vectors have nine components yet this also has only three independent values as well. Explain or show the three degrees of freedom.
- 8.) (5 pts) **Homogeneous Coordinates.** Show that any general transformation maps homogeneous points to their expected points after division by the homogeneous coordinate. That is, show that

$$\begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

maps the points

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \text{ and } \begin{pmatrix} hx \\ hy \\ hz \\ h \end{pmatrix}$$

to the same points (for any h) when homogeneous division is taken into account.

- 9.) (3 pts) **OpenGL Context.** Answer each of the following questions. What is an OpenGL context? What does it do? Where do you get one? What options does it have? Where/when does GLUT secretly create one? Notice that GLUT allows you to have only *one*.
- 10.) (2 pts) **Glew.** Glew, the OpenGL Extension Wrangler Library, is used a few times in this course. What is it and why is it used?