

Homework #2 and #3
Due Friday, October 12th and Friday, October 19th

- 1.
- a. Show that the following sequences commute:
 - i. A rotation and a uniform scaling
 - ii. Two rotations about the same axis
 - iii. Two translations
 - b. Show that the following does **not** commute:
 - i. A rotation and a translation

a. (part i)

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r \cos\theta & -s \sin\theta & 0 \\ r \sin\theta & s \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Note: Keep only 1 variable instead of 2 (replace either the 'r' with 's' or the 's' with 'r') in the uniform scaling matrix above, before checking that the sequence commutes.

a. (part ii)

Use the angle sum identities.

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta \cdot \cos\phi - \sin\theta \cdot \sin\phi & -(\cos\theta \cdot \sin\phi + \sin\theta \cdot \cos\phi) & 0 \\ \sin\theta \cdot \cos\phi + \cos\theta \cdot \sin\phi & -\sin\theta \cdot \sin\phi + \cos\theta \cdot \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta + \phi) & -\sin(\theta + \phi) & 0 \\ \sin(\theta + \phi) & \cos(\theta + \phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a. (part iii)

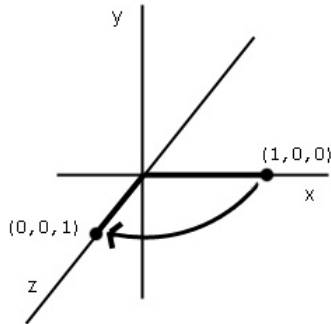
$$\begin{bmatrix} 1 & 0 & \Delta x_1 \\ 0 & 1 & \Delta y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \Delta x_2 \\ 0 & 1 & \Delta y_2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x_1 + \Delta x_2 \\ 0 & 1 & \Delta x_1 + \Delta y_2 \\ 0 & 0 & 1 \end{bmatrix}$$

b. (part i)

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & \Delta x \\ \sin\theta & \cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & \Delta x(\cos\theta - \sin\theta) \\ \sin\theta & \cos\theta & \Delta y(\cos\theta + \sin\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

2. Suppose we have a line extending from (0,0,0) to (1,0,0). We apply a single rotation to move one endpoint of the line to (0,0,1). Which vector was used as the axis of rotation?



Rotation Axis = $[0, k, 0]$, for $k \neq 0$

3. Write down the lighting equation that expresses lighting at a point in terms of ambient, diffuse, and specular components. Explain what the terms mean.

$$I = I_a K_a + I_d K_d (N \cdot L) + I_s K_s (N \cdot H)^n$$

Ambient : approximation to indirect light, or environment light

Diffuse: reflection is equal in all directions, Lambertian/matte surfaces

Specular: reflection is only in a particular direction, such as mirror reflection

4. Given a polygonal mesh as (x, y, z) vertices connected by edges and faces, how would you calculate each face's normal and the normal at each vertex?

Face normal: normalize (made to have unit length) the cross product of any two consecutive edge vectors.

Vertex normal: normalize the sum of all the associated face normals that share the same vertex

- 5.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \\ 1 \\ 1 \end{bmatrix}$$

- a. What is the homogenous 4-vector defined by the above multiplication?
 b. What is the real coordinate 3D space that the 4-vector represents?

(part a)

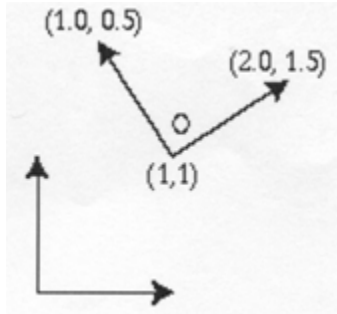
$$\begin{bmatrix} 6 \\ 4 \\ 1 \\ 2 \end{bmatrix}$$

(part b)

$$\begin{bmatrix} 3 \\ 2 \\ .5 \end{bmatrix}$$

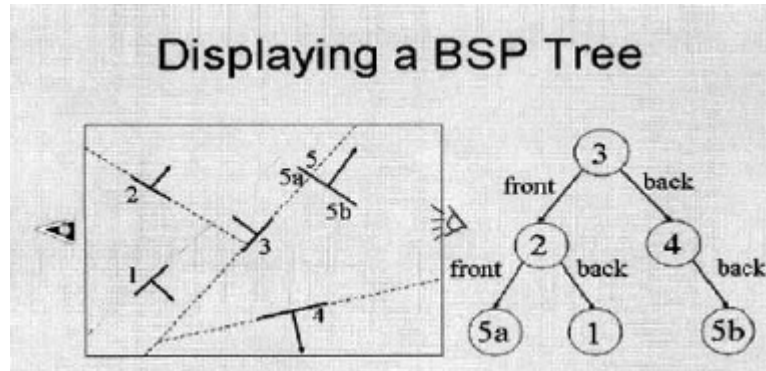
6.

- a. Drawn below are two 2D coordinate systems shown as a set of axes. The coordinates of the origin and the tips of the x and y axes of the coordinate system **O** (the object coordinate system) are given. These coordinates are with respect to the world coordinate system.



- Derive a transformation matrix that transforms the object coordinate system to the world coordinate system. You may write this matrix as a product of other matrices if you wish. You must fill in the entries for each matrix that you give (i.e. it is not enough to just give an answer that says rotate by 10 degrees); however, each entry may be a number or a mathematical expression.
- b. Derive the transformation from the world coordinate system to the object coordinate system **O**.

- The diagram below shows a BSP tree. The numbered darker lines are polygons, and the arrows show the orientation of the polygon.

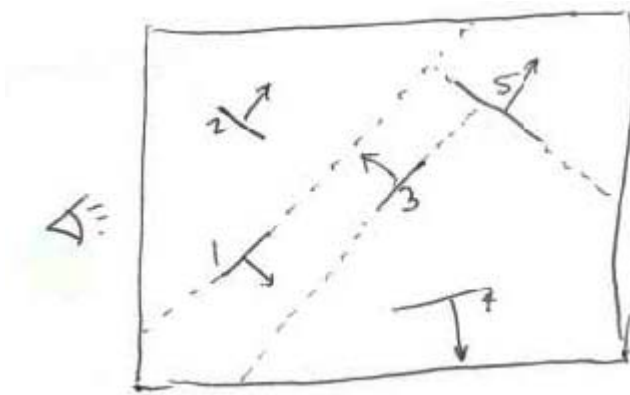


- Suppose we look at this scene from the right side, instead of the left. Use the BSP tree to determine the order in which triangles should be rendered.

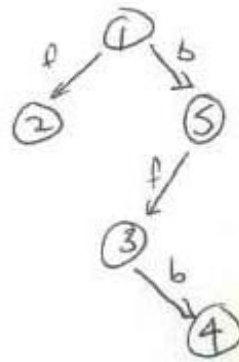
1, 2, 5a, 3, 5b, 4

Back to front from the new eyepoint (back and front labels in tree irrelevant).

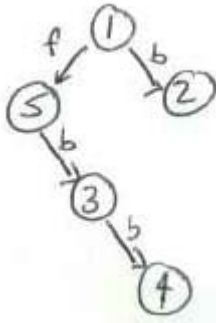
- Construct a valid BSP tree starting at triangle 1, instead of triangle 3 (starting from the left, as shown).



If labeling front to back relative to viewpoint:



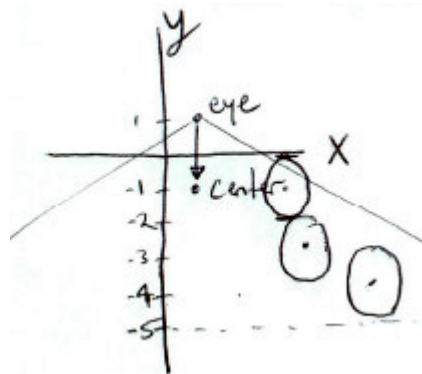
If labeling front to back relative to polygons (as above):



2. Suppose we set up a camera using `gluLookAt(eye=[1, 1, 3], center=[1, -1, 3], up=[1, 0, 0])`, and then render a scene with three spheres of radius 1, centered at $[4, -1, 4]$, $[5, -3, -2]$, and $[7, -4, 1]$.
 - a. What are the best (or “tightest”) distances for the near and far planes?

The eye is looking at the $-y$ direction. The eye is at $y = 1$. The closest sphere is at $y = 0$, and the furthest at $y = -5$.

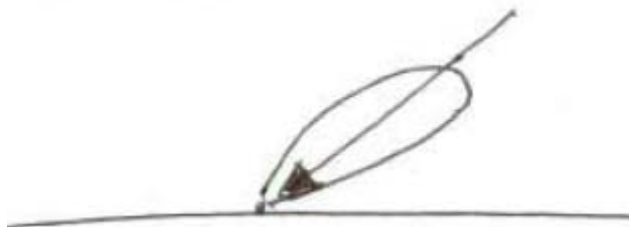
Thus, **near = 1** and **far = 6**.



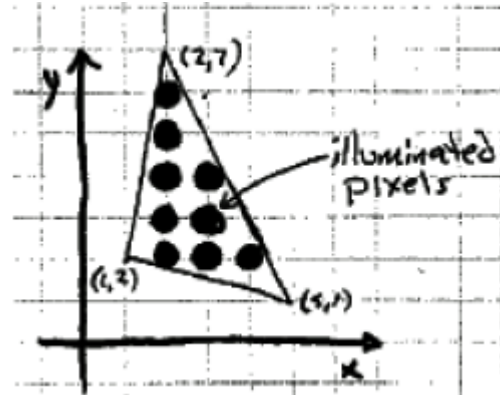
3. You are rendering a scene for the highway department (since they are the government, you have scored a \$1 million dollar contract to make one picture). They insist that your renderings include the shiny little reflective markers along the side of the road. It turns out these are made from *retroreflective* material, meaning that light is reflected mostly back in the direction of the light source.



Sketch a goniometric diagram for a retroreflective surface.

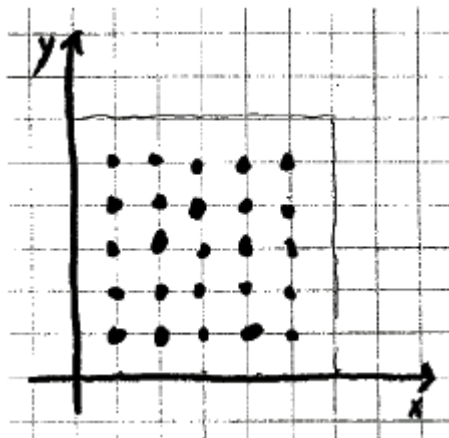


4. Suppose we have a graphics system that rasterizes polygons by turning on pixels that are strictly inside the bounds of the polygon. For example:



- a. Suppose we execute the following, which pixels will be turned on?

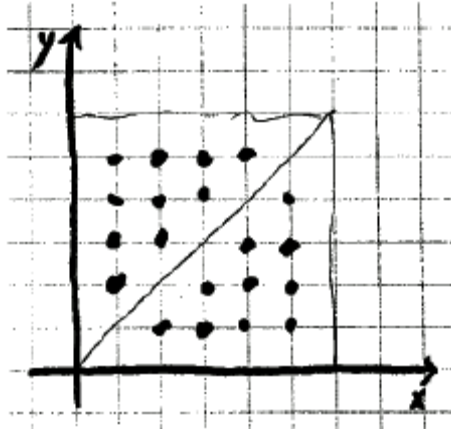
```
glBegin();
  glVertex(0, 0);
  glVertex(6, 0);
  glVertex(6, 6);
  glVertex(0, 6);
glEnd();
```



- b. Suppose that this system cannot rasterize quadrilaterals directly, and instead breaks them into triangles. What are the vertices of two triangles that would represent this quad?

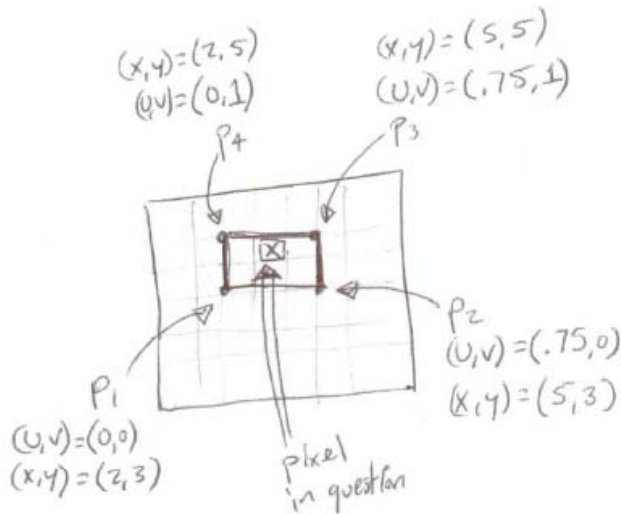
Tri 1: $(0, 0)$, $(6, 0)$, $(6, 6)$
 Tri 2: $(0, 0)$, $(6, 6)$, $(6, 0)$

- c. Suppose we now rasterize these two triangles, which pixels will be turned on?



- d. What is wrong with this rasterizer and how might we fix it?
- i. There is a line down the middle of pixels that were not turned on.
 - ii. We should rasterize the pixels on the top and left edges (inclusive) also. We should not turn on ALL the pixels on the edges or some will be drawn twice.

5. You are designing a graphics system that only supports texture maps that are 256x256. Obviously proper texture filtering will be required in some cases. As a filter, you select box filtering over the area covered by a pixel.



As a particular example of texture filtering, let's consider the pixel marked with an X in the diagram above. If $T(256, 256)$ references the texel color at u, v coordinates (1, 1) in the texture image, write an expression (math involving multiple texel values) for the color which should be copied into the frame buffer at pixel X.

Next, your boss tells you that you must support *summed area tables*. Write an expression for the color at X, assuming your texture is stored in this new format.