Homework #1

Displays, Image Processing,

Affine Transformations, Hierarchical modeling, Projections

Assigned: Monday, May 5

Due: **Wednesday**, May 14
at the beginning of class

Directions: Please provide short written answers to the following questions, using this page as a cover sheet. Feel free to discuss the problems with classmates, but please ***answer the questions on your own.***

Name:____

Problem 1: Display Devices (9 points)

In order to allow more time for transmitting and displaying each pixel, the American broadcast television standard (NTSC) uses an "interlaced" type of refresh, in which each video frame is broken into two "fields," each containing one-half of the picture. The two fields are "interlaced" in the sense that each field contains every other scan line: all odd-numbered scan lines are displayed in the first field, and all even-numbered scan lines are displayed in the second.

The purpose of an interlaced scan is to place some new information in all areas of the screen at a high enough rate to avoid flicker, while allowing the hardware more time for accessing and displaying each pixel.

1. (1 point) If the video controller displays each field in 1/60th of a second, what is the overall frame rate for displaying the entire screen?

2. (2 points each) An interlaced refresh works well as long as adjacent scan lines display similar information. In which parts, if any, of the following images would you expect to see flicker on an interlaced display (and briefly mention why):

   - A single-pixel-wide horizontal white line on a black background?

   - A single pixel wide vertical white line on a black background?

   - A checkerboard of black and white, where each black or white square is a single pixel?

3. (2 points) On interlaced displays, there is a very noticeable artifact when objects on the screen are moving fast. What is this artifact, and how will it appear if the video is of a white box moving horizontally on a black background?

Problem 2: Image processing (26 points)

In this problem, you will consider several convolution filtering operations and their behaviors. *For each sub-problem, justify your answer.*

(a) (4 points) The image you're editing is too dark, and you decide you need to amplify the value of each pixel by a factor of 2. Suggest a convolution filter that will double the value at each pixel of the image without changing it in any other way. (Technically, after scaling pixel values, they could be out of range; assume that any needed clamping will be taken care of later, after filtering).

(b) (4 points) While taking a photograph with your digital camera, you fail to hold the camera steady, and it translates downward while the shutter is open. You discover this later when you see that horizontal edges, in particular, have been blurred a bit (an effect called "motion blur"). You decide to filter the image so that horizontal edges are sharpened, but vertical edges are unchanged. Suggest a single convolution filter that does this.

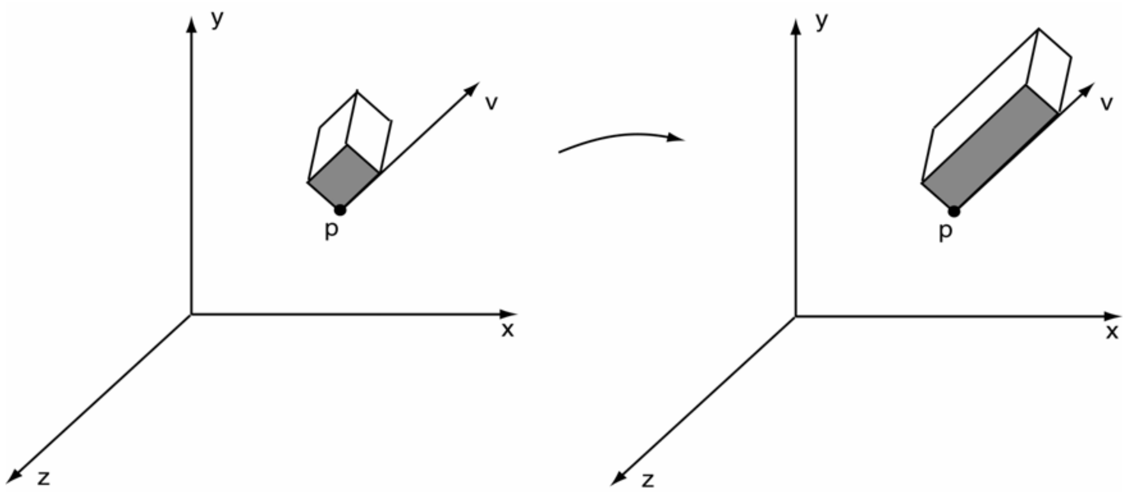
(c) (4 points) After thinking a little more about the previous picture, you decide that motion blur is cool, and you want to apply it to another image. In this case, though, you want to simulate the effect of a camera translating diagonally along the x=y direction while the shutter is open. Suggest a convolution filter that would accomplish some diagonal blurring along that direction by averaging across *m* pixels.

(d) (4 points) Describe a non-constant image that, when convolved with your diagonal blur filter from (c), would be unchanged by the filter. (You may ignore the boundaries.)

(e) (10 points) Suppose you pad the boundary of an image in some way that allows you to compute output values for every pixel being filtered by a convolution filter. For an image of dimensions *n* x *n* and a filter of dimensions *m* x *m*, how many output pixels will be influenced by input pixels "hallucinated" beyond the boundary of the image? For simplicity, assume that *m* is odd. However, *m* and *n* may otherwise have arbitrary positive values.

Problem 3. 3D affine transformations (32 points)

The basic scaling matrix discussed in lecture scales only with respect to the x, y, and/or z axes. Using the basic translation, scaling, and rotation matrices, specify how to build a transformation matrix that scales along any ray in 3D space. This new transformation is determined by the ray origin $\mathbf{p} = (p_x, p_y, p_z)$ and direction vector $\mathbf{v} = (v_x, v_y, v_z)$, and the amount of scaling s_v. For clarity, a diagram has been provided, showing a box being scaled with respect to a given ray. Your answer should work for *any* ray, not just the case shown in the picture.



You can use any of the following standard matrices (from lecture) as building blocks: canonical axis rotations (a.k.a., Euler angle rotations) $R_x(\alpha)$, $R_y(\beta)$, $R_z(\gamma)$, scales $S(s_x,s_y,s_z)$, and translations $T(t_x,t_y,t_z)$. You don't need to write out the entries of the 4x4 matrices. It is sufficient to use the symbols given above, supplied with the appropriate arguments. All scale factors are strictly positive. You must compute the angles of any rotations required. Note that you may require inverse trigonometric functions, and you should assume that $\cos^{-1}(x)$ outputs a range of $[0..\pi]$, and that $\sin^{-1}(x)$ and $\tan^{-1}(x)$ each outputs a range of $[-\pi/2..\pi/2]$.

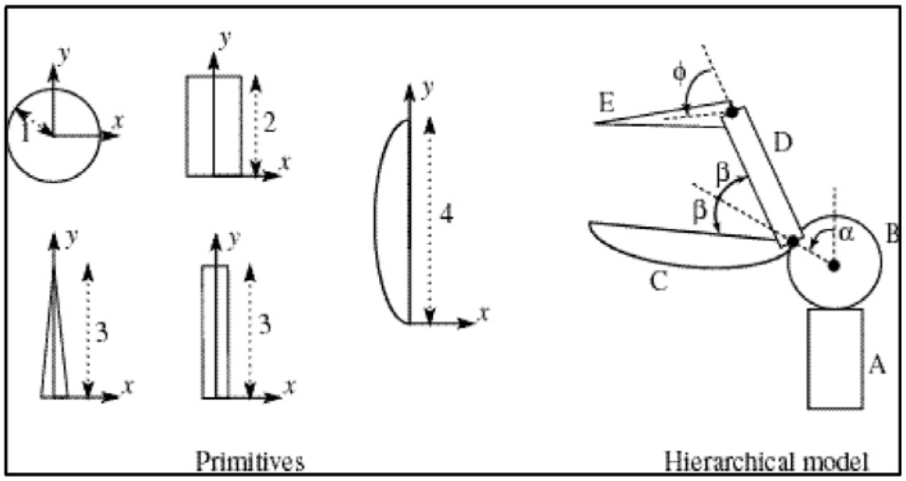
There are many possible solutions to this problem. To constrain the space of answers, and to give you a solution hint: you must cause the $\mathbf{v}$ direction to align with the y-axis at some stage of your solution.

Show your work, using words and drawings as needed to support your answer.

Problem 4. Hierarchical modeling (18 points)

Suppose you want to create the hierarchical model shown below. The model is comprised of five parts, labeled **A**, **B**, **C**, **D,** and **E**, and each part is drawn as one of five primitives given below (they are already scaled to the correct sizes). The following transformations are available to you:

- R(θ) – rotate by θ degrees (counter clockwise)
- T(a, b) – translate by $[a \ \ b]^T$



(a) (15 points) Construct a tree to specify the hierarchical model where the nodes of the tree should be labeled **A**, **B**, **C**, **D**, and **E**. Along each edge, write out the product of matrices that should be performed when traversing that edge. Insert numerical values (i.e., for primitive sizes) where available.

(b) (3 points) Write out the full transformation expression to be applied to the part labeled **E**.

Problem 5: Perspective Projections (15 points)

Recall the matrix for perspective projection $P(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix}$

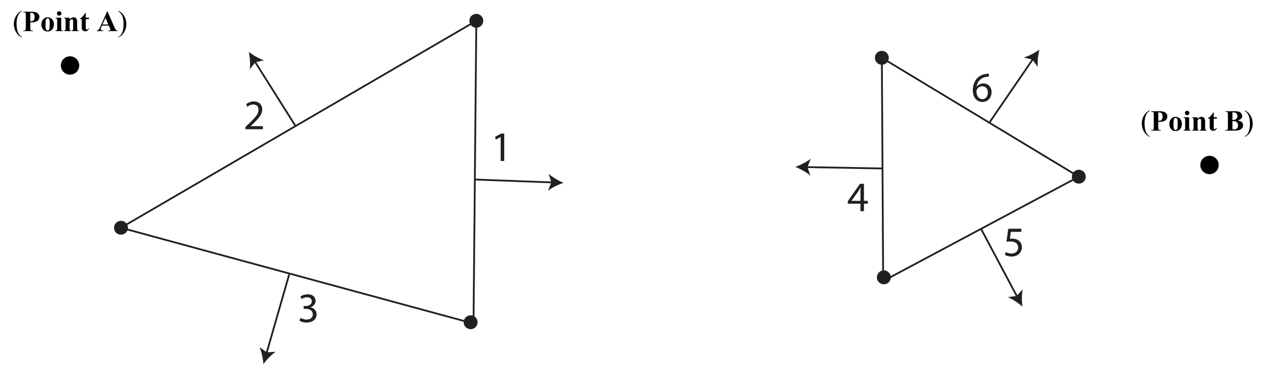
a. (9 points) The perspective projection matrix takes a set of points p_i in 3-space and maps them to points on the viewing plane $p_i' = P(d)p_i$. If we first apply a uniform scale by a factor s to all points in 3-space, before performing the projection indicated by this matrix $p_i' = P(d)S(s)p_i$, how will the projected points be different from the case where no scale is applied? Show your reasoning mathematically and explain the result geometrically (in words and/or with a sketch). ***Hint: it may be useful to think about what this looks like in two dimensions instead of three.***

b. (6 points) Using the perspective projection matrix above, we observed in class that any set of parallel lines will converge and intersect at a vanishing point. In fact, some parallel lines will ***not*** intersect, i.e., they will remain parallel, even after undergoing this perspective projection. Which lines are these? Geometrically speaking, what do all sets of these lines have in common?

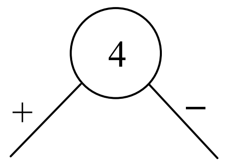
Problem 6. BSP trees (19 points)

Recall that Binary Space Partitioning (BSP) trees break the world up into a tree of positive and negative half-spaces. As spatial partitioning data structures they are very versatile and can be applied to aid in hidden surface removal, collision detection, and even robot motion planning.

Below is a world in two dimensions described by a set of numbered line segments. Note that each segment normal (shown as arrows) points into the positive (+) half-space of its segment.



(a) (4 points) Draw a diagram of a BSP tree that could be generated from this scene using segment #4 as the root (as illustrated below). Note: there are multiple valid answers.



(b) (3 points) Given the viewpoint marked **Point A** in the scene, traverse your BSP tree to list the polygons in the order they would be rendered for hidden surface removal so that no Z-buffer is required (i.e., using the "Painter's algorithm"). For you answer, you only need to provide the list of primitives in the order in which they will be drawn.

(c) (5 points) An alternative approach to hidden surface removal is Z-buffering. Suppose we are performing time-consuming shading at each pixel while Z-buffering. As noted in class, performing shading calculations *after* doing the Z comparison will save some work, but you can still end up overwriting pixel values many times. How can Z-buffering and BSP trees be combined to minimize the number of shading calculations needed to render a scene? Assuming that all objects (primitives) in an arbitrary scene are opaque and non-overlapping, would we ever shade any pixel more than once? Justify your answer.

(d) (3 points) Using the Z-buffer + BSP tree combination from (c) and given your BSP tree from (a), in what order would you draw the line segments for the viewpoint marked **Point B** above?

(e) (4 points) Suppose we render with "backface culling," so that we don't draw any polygons that are facing away from the viewer. How would you modify the BSP-based traversal and drawing algorithm? Explain and then write out the ordered list of primitives that would be drawn for part (b) of this problem and then for part (d), if we used backface culling.

Extra Credit: Rotations as Shear Transformations (10 points)

You are ***not*** required to do this problem! But extra points are good things, so we encourage you to give it a try.

In 2D, a rotation transformation by angle θ can be specified as a series of shear transformation matrices. Give these matrices, or if it can't be done, prove it.