Problem 1: Transformations (2008fa)

Express the homogeneous 3D transformation defined by the matrix

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

as a sequence of transformations in the following ways:

1. A rotation followed by a translation.
2. A translation followed by a rotation.

Express the 2D linear transformation defined by the matrix

\[
\begin{bmatrix}
2 & 1 \\
1 & 2 \\
\end{bmatrix}
\]

as a sequence of transformations in the following ways:

3. A rotation followed by a nonuniform scale followed by a rotation.
4. A shear along the x axis followed by a nonuniform scale followed by a shear along the y axis.

Hint: For the 2D transformations, sometimes it might be easier to draw a picture and see how the given matrix transforms points in the plane (the points of the unit box for example)
Problem 2: Ray tracing 1 (2008fa)

![Images of ray tracing examples](image)

Figure 2: Four images from a ray tracer. The left image is correct, and the other three were produced by introducing single-statement bugs into the program.

Look at each of the three images in Figure ?? that were produced by a Ray I ray tracer with various bugs. For each one:

(a) Could it have been caused by a problem with ray generation?

(b) Could it have been caused by a problem with ray intersection?

(c) Could it have been caused by a problem with shading computations?

For each “yes” answer, back it up with an example of an error that would cause the observed symptoms. There is no right or wrong explanation; only plausible and implausible ones. But when there is a clearly plausible cause, very far-fetched explanations (which are roughly equivalent to a “no” answer) won't make full credit.

Shadow computations count as part of shading. Computing surface normals counts as part of ray intersection.

Use “yes”, “possibly” and “not likely” as answers and give plausible explanation/example for each one of them.
The following pseudocode intersects a ray with some geometric shape.

```
function surfaceIntersect(Ray r)
    Point3 p = r.origin;
    Vector3 d = r.direction;
    p.x = 2 * p.x;
    d.x = 2 * d.x;
    [r1, r2] = quadraticRoots(d.x*d.x + d.y*d.y, 
                              2*(p.x*d.x + p.y*d.y), 
                              p.x*p.x + p.y*p.y - 1);
    t1 = -p.z/d.z;
    t2 = (1 - p.z)/d.z;
    tmin = max(min(r1, r2), min(t1, t2));
    tmax = min(max(r1, r2), max(t1, t2));
    if (tmin < tmax) {
        if (tmin > 0) return tmin;
        if (tmax > 0) return tmax;
    }
    return INFINITY;
```

The function `quadraticRoots` returns the two roots of a quadratic with the given coefficients, if there are two roots.

1. What is the shape? Give a detailed and precise definition, including all dimensions.

2. When there are no roots, what values could `quadraticRoots` return for `r1` and `r2` that will make this code work correctly without changes?

3. Give an example of a hit with `d.z == 0` and explain what values the variables in this function take on and how the return value arises.
Problem 4: Transformation matrices (2008fa)

Classify each of the following 2D homogeneous matrices as follows: (a) rotation, (b) mirror reflection, (c) uniform scale, (d) nonuniform scale, (e) translation, (f) shear, or (g) combination.

Earlier categories take precedence over later categories if more than one applies. There is not a one-to-one mapping between the matrices and the categories.

1. \[
\begin{bmatrix}
1 & -1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

2. \[
\begin{bmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

3. \[
\begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

4. \[
\begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

5. \[
\begin{bmatrix}
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

6. \[
\begin{bmatrix}
2 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

7. \[
\begin{bmatrix}
1 & -1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
**Problem 2: Planar Shadows (15 pts)**

Consider a directional light source (unit direction, $u$), and the planar shadow created by literally projecting each mesh vertex position, $p$, to its shadow point $\hat{p}$ on the 3D plane specified (in homogeneous coordinates) by $c^T \hat{p} = 0$ where $c \in \mathbb{R}^4$. Derive a formula for the 4x4 projection matrix, $A$, that maps a homogeneous object point, $p = (x, y, z, 1)^T$, to its shadow point, $\hat{p} = Ap$. (*Hint: Consider the ray $p + tu$.*
Problem 6: View Frustum Culling (2009fa)

“View frustum culling” is a technique to avoid drawing (or cull) geometry which is outside the view frustum. To assist with culling, assume that each object has a bounding sphere with object-frame center position, $c_o = (c_x, c_y, c_z, 1)^T$, and radius $R_o$. Imagine that you know you have an $[l, r] \times [b, t] \times [f, n]$ orthographic viewing volume, and you know each of the matrices $(M_{vp}, M_{orth}, M_{cam}, M_m)$ used to construct the orthographic view transformation which maps points from object space to screen space:

$$p_s = \begin{pmatrix} x_s \\ y_s \\ z_c \\ 1 \end{pmatrix} = M_{vp} M_{orth} M_{cam} M_m \begin{pmatrix} x_o \\ y_o \\ z_o \\ 1 \end{pmatrix}.$$  

Derive a simple mathematical test to determine if an object is safely “off screen.”

Assume that $M_m$ is a rigid-body transformation
Problem 7: Camera matrix (2003fa)

Write the 4x4 matrix for this affine transformation:

3. (7 pts) A viewing transformation for a camera at the position (0, 3, 4) looking at the origin with up vector (0, 1, 0). It is OK to use a matrix inverse in your answer.

Hint: Try solving the problem, without explicitly computing cross products, but just considering the right-hand rule.