University of California San Diego
Department of Computer Science
CSE167: Introduction to Computer Graphics
Spring Quarter 2016
Midterm Examination #1
Tuesday, April 26th, 2016
Instructor: Dr. Jürgen P. Schulze

Name:________________________________

Your answers must include all steps of your derivations, or points will be deducted.

This is closed book exam. You may not use electronic devices, notes, textbooks or other written materials.

Good luck!

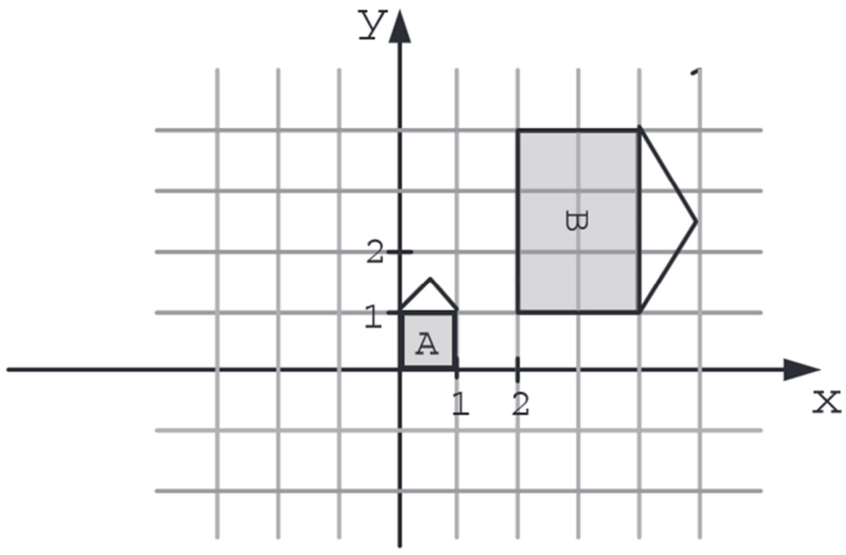
Do not write below this line

Exercise	**Max.**	**Points**
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
Total	80	

1. Affine Transformations (10 Points)

Find the 3x3 homogeneous matrix that transforms the 2D vertices of object **A** to the corresponding vertices of object **B** in the figure below. Express the matrix as a composition of elementary transformations such as translation, scaling or rotation, and calculate the resulting 3x3 matrix for the composite operation.

Note that for rotations which are multiples of 90 degrees, the sin and cos terms in the rotation matrix simplify to 1, -1, and 0, depending on the angle. If you cannot remember what the rotation matrix you need looks like, try to derive it based on this knowledge.



2. Projection (10 Points)

a) A vertex (point) is drawn at the origin of the object coordinate system.

It is viewed through a camera whose inverse transformation matrix is:

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

The object that the vertex is part of is drawn with transformation matrix:

$$\begin{bmatrix} 0 & -1 & 0 & 5 \\ 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This simple projective transformation matrix is used:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Where does the point appear in screen coordinates? Give the x/y position. (6 points)

b) For each of the following statements, mark them with a P if it is true of a perspective projection, O if it is true of an orthographic projection, B if it is true of both orthographic and projective, and N if it is true of neither. (Use P or O only if the property is true of only one type of transformation). (4 points)

__ Straight lines are mapped to straight lines

__ Distances are preserved

__ Angles are preserved

__ Far away objects appear the same size as closer ones

3. Graphics Pipeline (10 points)

a. Camera Space
b. Image Space
c. Inner Space
d. Canonical View Volume Space
e. Object Space
f. Outer Space
g. Projective Space
h. World Space

Select the correct spaces (=coordinate systems) from above and list them in the order that we expect each vertex to go through in the traditional transformation process, which transforms vertex coordinates to pixel coordinates:

Fill in letters here: 1. ____ 2. _____ 3. _____ 4. _____ 5. _____

4. Camera Matrix (10 points)

Throughout the quarter, we have been using the GLM math library. It conveniently provides a function called **lookAt()** which takes three *vec3* variables as parameters for the center of projection, look at point, and up vector.

a. Complete the three equations below used to calculate the three coordinate axes of a camera (6 points):

$$\widehat{x_{axis}} =$$

$$\widehat{y_{axis}} =$$

$$\widehat{z_{axis}} =$$

b. Given the camera matrix below, fill in the two matrices used to calculate the inverse camera matrix (4 points):

$$C = \begin{bmatrix} 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C^{-1} = \begin{bmatrix} , & , & , & \\ , & , & , & \\ , & , & , & \\ , & , & , & \end{bmatrix} * \begin{bmatrix} , & , & , & \\ , & , & , & \\ , & , & , & \\ , & , & , & \end{bmatrix}$$

5. Lights (10 Points)

a) Name two differences between directional lights and point lights. (4 points)

b) Which two additional parameters do spot lights have compared to point lights? (4 points)

c) Which of the three types of light sources we covered in class is typically used to simulate sunlight? (2 points)

6. GLSL Shaders (10 points)

Given this vertex shader (file name shader.vert)

```
#version 330 core

layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;

uniform mat4 MVP;
uniform mat4 M; //M: Model Matrix
uniform mat4 V; //C^-1: Inverse Camera Matrix
uniform mat4 P; //P: Persp Projection Matrix
uniform mat3 N; //N: Normal Matrix created from Model Matrix

out vec4 fragPos;
out vec3 fragNormal;

void main()
{
   gl_Position = MVP * position;
   fragPos = ___________________;
   fragNormal = normalize(N * normal);
}
```

And this fragment shader (file name shader.frag)

```
#version 330 core

layout (location = 0) out vec4 color;

in vec4 fragPos;
in vec3 fragNormal;

uniform struct DirLight {
   bool on;
   vec3 dir;
} dirLight;

void main()
{
   vec4 color = vec4(0.0f); // Color is black when light is off
   if (dirLight.on)
   {
      float nDotL = ___________________________;
      vec3 material = vec3(0.72f, 0.43f, 0.47f); // Pink Material
      color = ______________________;
   }
}
```

The above two shaders are used for simplified directional lighting (purely diffuse and white). Answer the questions below.

a. The vertex shader is passing the fragment's position to the fragment shader through the variable `fragPos`, so that we can calculate our lighting in world space. What should go in the blank space on line **`shader.vert:18`**? (3 points)

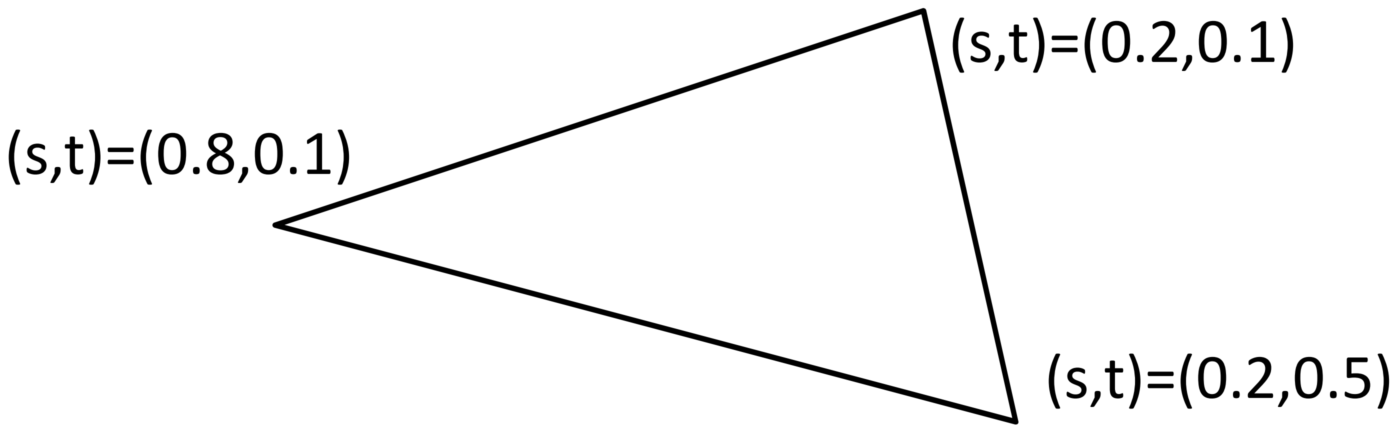
b. We've run into an issue and the lighting is not showing up. We suspect our `dirLight`'s initial direction, coming from screen right-to-left (i.e. `<-1, 0, 0>`), is not properly being assigned. If we want to debug and show this direction visually (we expect to see a flat red (i.e. `(1, 0, 0)`) on our object), what must go in the blank space on line **`shader.frag:20`**? (3 points)

c. Assuming we've now fixed the directional light's direction, we want to see the diffuse color of the directional light. What code must go in the lines **`shader.frag:18`** and **`20`** to see our result? (4 points)

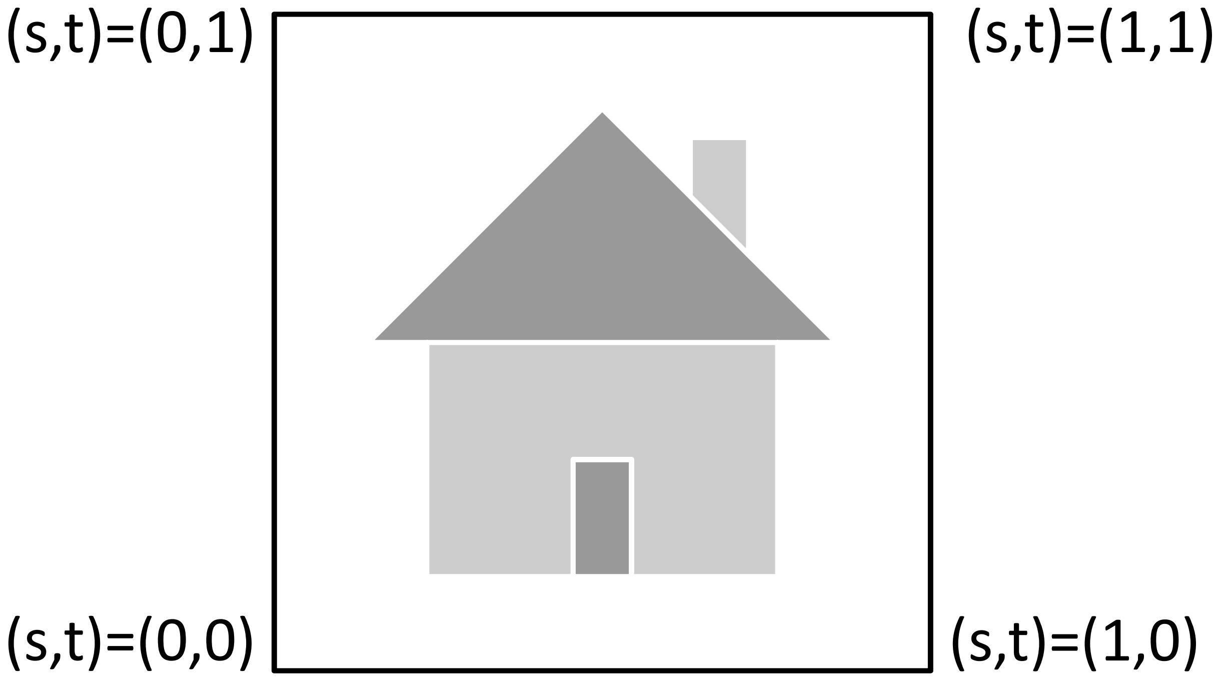
7. Texture mapping (10 points)

Part 1:

Given a triangle with associated texture coordinates (s,t) for its vertices

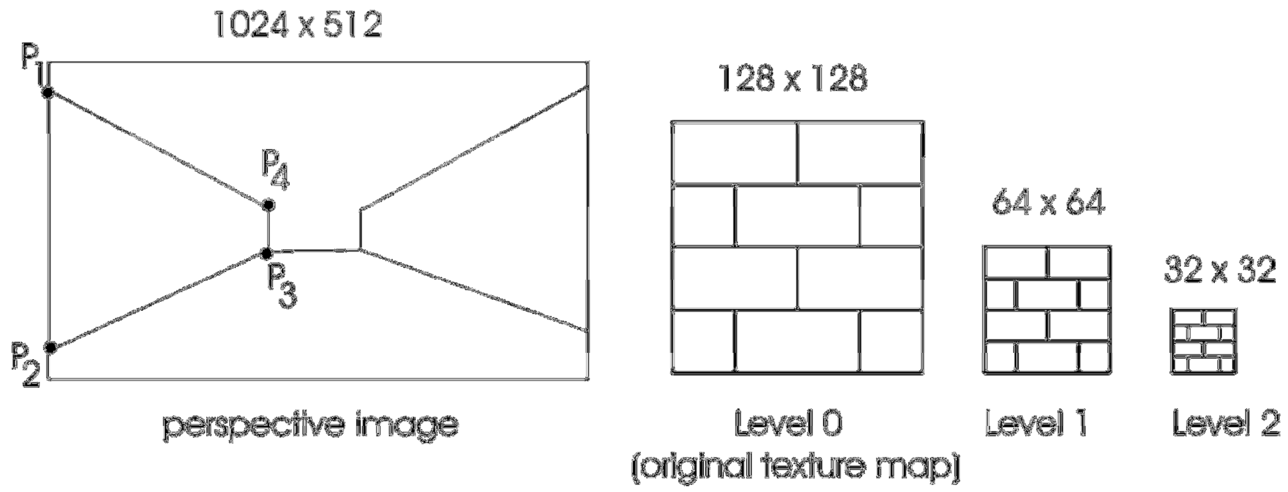


and a simple texture with its corner coordinates given:



a) Sketch **into the triangle above** what it will look like after texture mapping. (4 points)

Part 2:
Suppose we have a brick wall that forms the left-hand wall of a corridor in a maze game, as shown in the image below, and it is defined (in world coordinates) by points P_1, P_2, P_3, P_4. Assume that the brick wall is to be 16 bricks high and 200 bricks long.



b) Using the height of the brick wall as seen in the image, estimate (with derivation) how many texels of the original texture map a screen pixel covers, both for near points on the wall, i.e., on the edge P_1P_2, and at distant points on the wall, i.e., on the edge P_3P_4. (3 points)

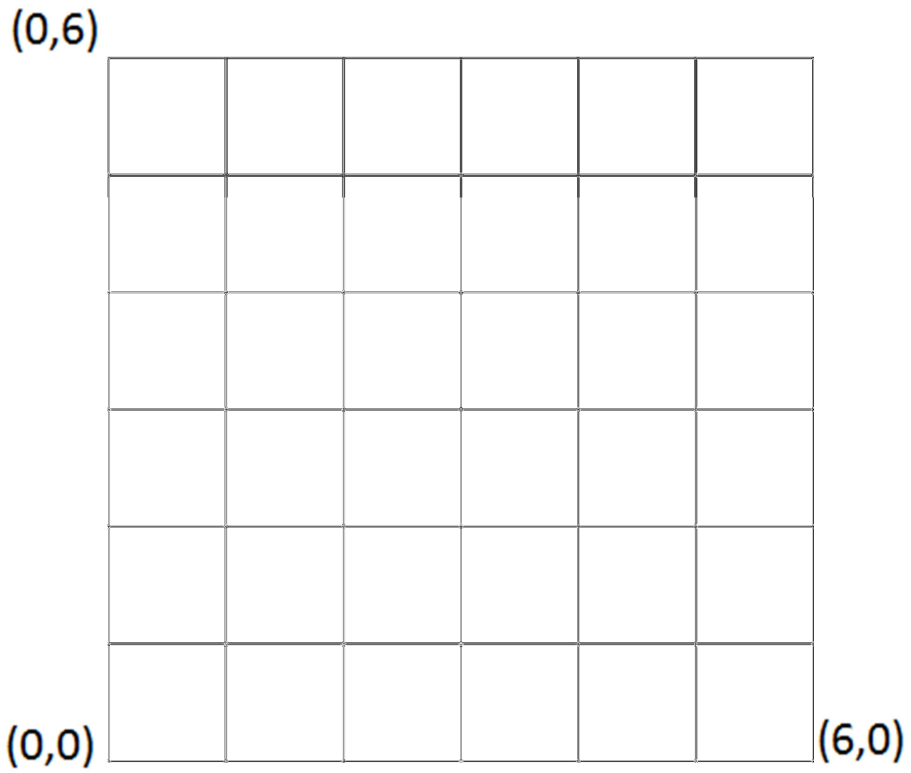
c) In the perspective image above, sketch approximately what regions of the wall on the left (defined by points P1, P2, P3 and P4) will use each of the levels of the mip-map image pyramid on the right (use nearest-mipmap interpolation, not trilinear mipmapping). (3 points)

8. Scene Graph (10 Points)

Transformations	**Primitives**
Translate(x,y) – translates by vector (x,y). Scale(sx,sy) – scale by factors sx and sy.	Circle – centered at the origin with radius 0.5 Square – centered at origin with side length 1

a) Draw the figure that this scene graph structure represents. (6 points, 1 for each shape). Hint: it's a robot.

$$\begin{tikzcd}
 & & & \text{Translate(3,3)} \arrow[dll] \arrow[dr] & & \\
 & \text{Scale(2,2)} \arrow[dl] \arrow[dd] & & & \text{Scale(1,2)} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Translate(0,1)} \arrow[d] & & \text{Translate(-1.5,0)} \arrow[d] & \text{Translate(1.5,0)} \arrow[d] & \text{Translate(0.5,-1)} \arrow[d] & \text{Translate(-0.5,-1)} \arrow[d] \\
\text{Circle} & \text{Square} & \text{Square} & \text{Square} & \text{Square} & \text{Square}
\end{tikzcd}$$



b) Modify the scene graph to put a disk in the robot's hand. Use a Circle primitive for the disk, and draw your modification on the current scene graph. Your modification must keep the disk in the robot's hand under any arm transformation. (4 points)