CSC 471 midterm 1 – Spring 2017

Name:____

READ ME FIRST

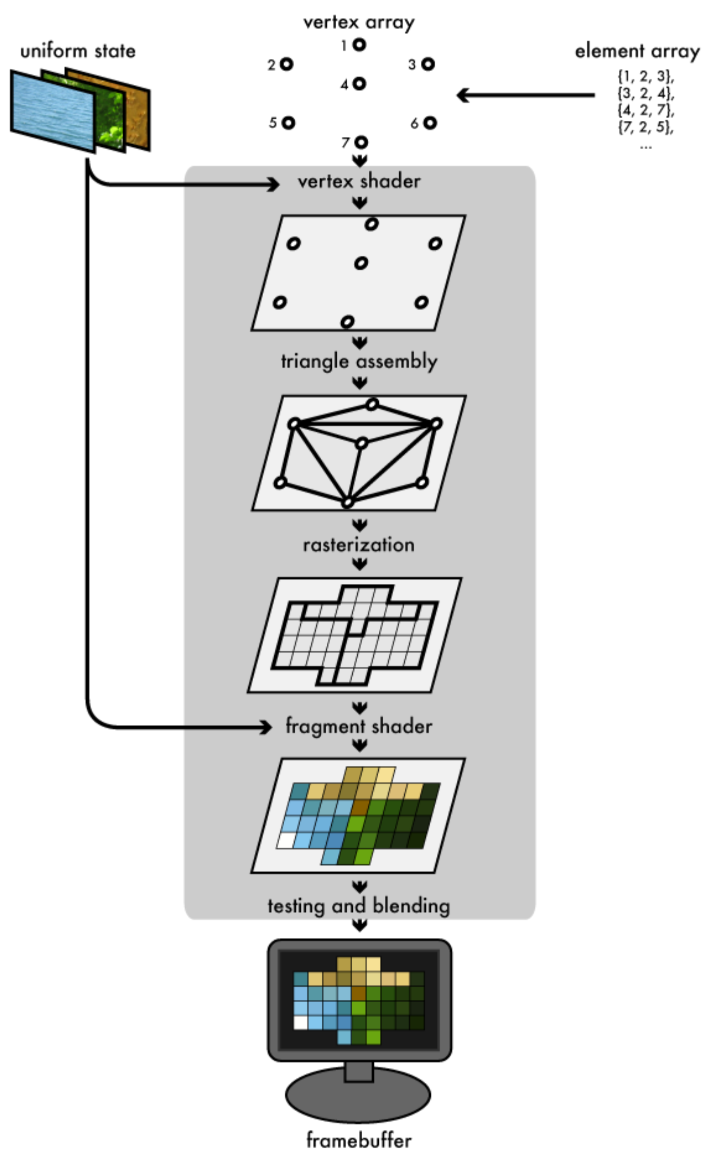
- Work individually! You may use a calculator
- Don't spend too much time on any one problem. This exam should take 80 minutes.
- Be neat
- Show how you got your answers!
- When in doubt, write down your assumptions
- You are allowed to use a calculator

1	15 pts	Short answer	
2	10 pts	Vectors	
extra credit	2 pts		
3	30 pts	2D transform matrices	
4	15 pts	Transforms	
5	20 pts	More Transforms	
6	10 pts	Rasterization	
	100 pts	Grand total	

1) **Short answer/ true & false questions (20 pts)**

a) (1 pt) In a very general sense, the GPU can be viewed as a SIMD machine that allows a program to run the same ‘vertex shader’ program on multiple different vertices in parallel and then run a ‘fragment shader’ program on multiple fragments in parallel, thus speeding up the process of rendering computer graphics
True **False**

(b-f) Refer to the following figure and fill in the missing information – short answers (2 pts each):



(b) Describe what data is typically transferred from the CPU to the GPU for graphics applications:

c) Describe what computational task a vertex shader typically does?

d) Describe in your own words, what happens during rasterization:

e) Describe what computational task a fragment shader typically does?

f) What are the value ranges of the geometry data at the end of the graphic’s pipeline, i.e. {x, y}?

g) (4 pts) Assume in your game the circle defined by:

$$f(x,y) = (x - x_c)^2 + (y - y_c)^2 - r^2$$

with {xc, yc} = {-1, 5} and a radius of 2.5, is shielded from the highly contagious zombie virus. If you place your trusty steed at point {1, 6} are they safe from contamination? (show your work with math):

2) Vectors (10 pts)

Given the following vectors: $\mathbf{v}^T$ =[7, 9, 3] and $\mathbf{u}^T$ = [7, 11, 3] Compute**:**

1) **(2 pts) v+u=**

2) **(2 pts) v·u=**

3) **(2 pts) If w = v–u,** What is the length of the vector **w?**

4) **(4 pts)** Write the normalized form of **w (from the part 3)** (i.e. write **w** as a unit length vector).

5) **(2 pt extra credit):** draw the vector -1***w** (accurately depicting length (ratio) and direction) as some part of a creature (make it clear which part of the creature is the vector) – you may define the units (i.e. inches, feet, etc.)

3) 2D transform matrices (30 pts)

Given the following 2D transform matrices:

$$m_0 = \begin{bmatrix} .707 & -.707 & 0 \\ .707 & .707 & 0 \\ 0 & 0 & 1 \end{bmatrix} m_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} m_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} m_3 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a) Name what type of 2D transformation is associated with each matrix and say something about the magnitude of the transform for x or y. **(4 pts total)**
m0:

m1:

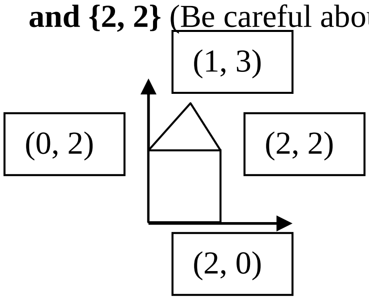
m2:

m3:

b) If these are 2D transforms, why are they 3x3 matrices? (Write 1-2 sentences) **(2 pts)**

c) Carefully compute m0*m2 (that is write out the composite matrix) (4 pts):

d))**(13 pts total)**
(4 pts) **Draw** the result of applying the composite matrix (from part (c) – i.e. m0*m2) to the following figure (draw the entire house transformed). (3 pts each) **Include coordinate labels for your completed drawing for the updated points {0, 2}, {1, 3} and {2, 2}** (Be careful about how you represent the 2D points as vectors of length 3

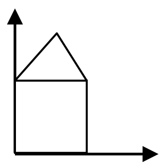


(3 pts) {0, 2} :

(3 pts) {1, 3} :

(3 pts) {2, 2} :

e) Now, only **draw** the result of applying two transforms: m1*m3 to the same figure (feel free to compute the composite matrix if that helps you, but it is not required). Be sure that your drawing includes a representation of the axes to clarify the house' exact final position: **(7 pts)**



4) Transforms (15 pts)

Assuming you have the following functions:

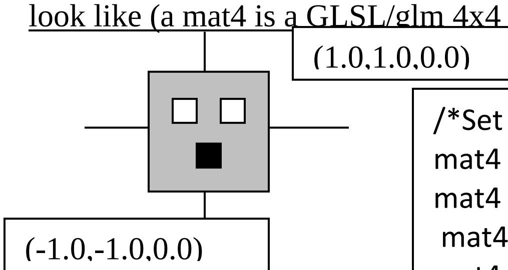
mat4 scale(float sx, float sy, float sz) {... } : returns a scale matrix

mat4 rotate(float angle, float ax, float ay, float ax) {...} : returns a rotation matrix by the given angle and axis [ax, ay, az]

mat4 translate(float tx, float ty, float tz) {...} : returns a translation matrix

*And assume the operator * is defined for matrix multiplication as expected*

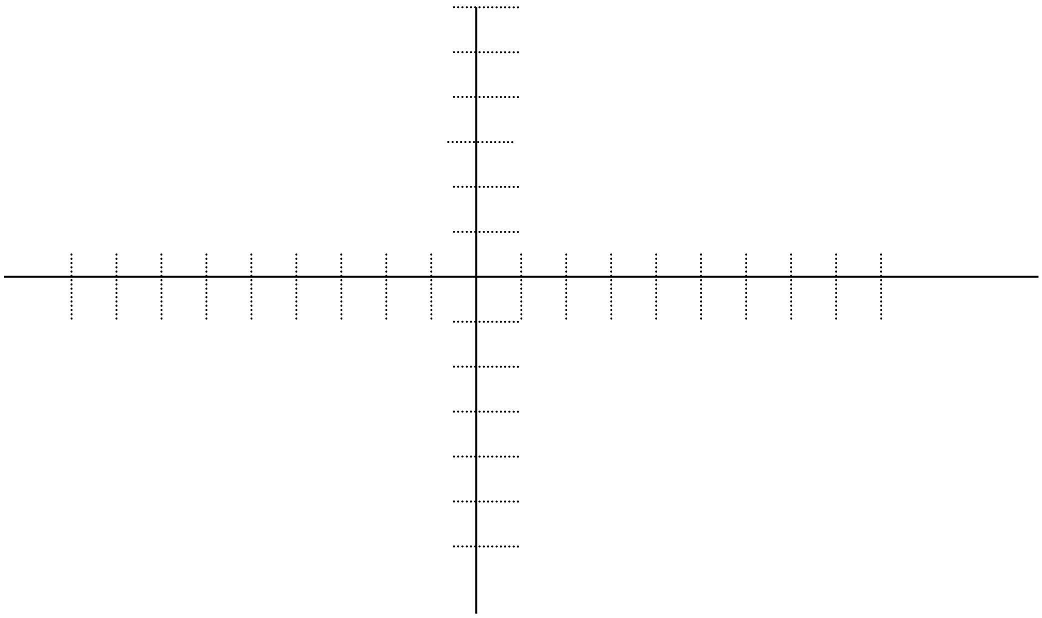
Carefully draw the result of the following OpenGL/GLSL code assuming that the DrawRobotFace() function draws the complete image below (i.e. one grey box with sides of length 2 with three small sub-boxes inside with sides of length 0.5: white eyes and a black mouth). Recall that rotations are specified as counter-clockwise. <u>**Carefully** read all the code below before drawing and be sure that it is clear what the final drawing will look like (a mat4 is a GLSL/glm 4x4 matrix – as expected):</u>




```
/*Set up the first matrix */
mat4 Scale = scale(2, 1, 1);
mat4 Trans = translate( -2, 0, 0);
 mat4 Rot = rotate( -45, 0, 0, 1);
mat4 Model = Trans*Rot*Scale;
/*send matrix to the vertex shader */
glUniformMatrix4fv(prog->getUniform("MV"), 1, GL_FALSE, Model);
 /* Draw */
 DrawRobotFace ();

/*Set up the second matrix */
mat4 Scale = scale( 1, 1, 1);
mat4 Trans = translate( 1, 1, 0);
 mat4 Rot = rotate( 45, 0, 0, 1);
 mat4 Model = Trans*Rot*Scale;
/*send matrix to the vertex shader */
glUniformMatrix4fv(prog->getUniform("MV"), 1, GL_FALSE, Model);
/* Draw */
DrawRobotFace ();
```

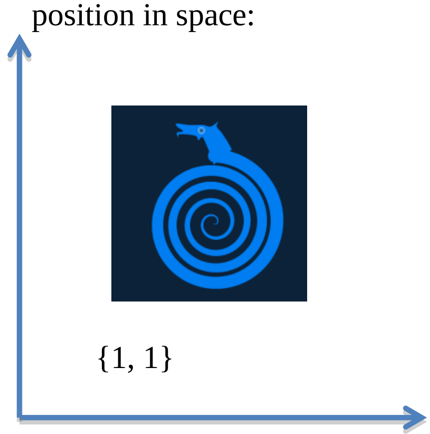
Complete your drawing on the next page

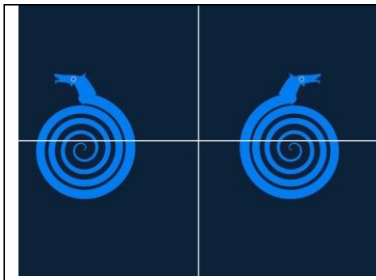
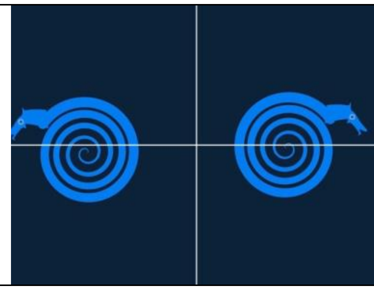
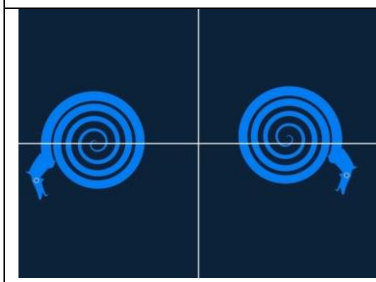
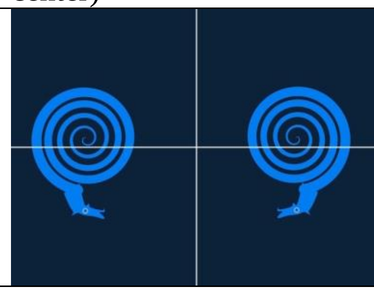


5) **More Transforms (20 pts) – *please write neatly***

Assuming that the DrawDragon() function draws the image below, that by default draws in a bounding box that ranges from a lower left corner of {1,1} and extends to an upper right corner of {2,2}. Recall that rotations are specified as counter-clockwise. **Write transform code, using a similar coding convention to what is used in question 4 that will result an animated scene (assume your code is within a loop – no need to write the loop).** The scene should include two dragons centered at {-1, 0} and {1, 0} each facing away from one another and each spinning around its center (the one on the right in a clockwise direction, with the one on the left spinning in a counter clockwise direction). Example frames from an implementation are included below for clarity.

Default draw position of the DrawDragon() – carefully note the dragon's default position in space:



	
1) The start of the scene – note the white lines represent the x and y axis	2) As time proceeds both dragons rotate in opposite directions (around their own center)
	
3) And continues to rotate	4) And continues to rotate…

Write any initialization code here:

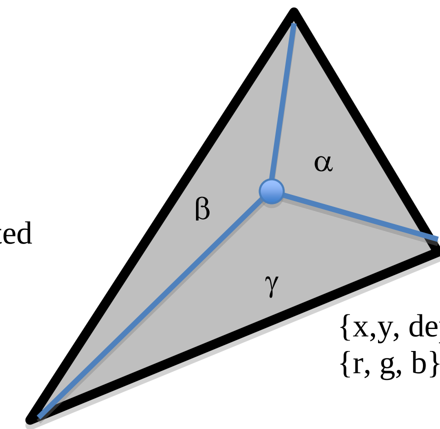
Write looped code here:

6) Rasterization (10 pts total):

If you have a triangle converted to window coordinates with the following coordinates, (including depths and colors) – **given the associated Barycentric coordinates** (ie do not compute them, use what is given):

{x,y, depth} = {215, 69, 3}
{r, g, b} = {0, 0, 1}

{x, y, z} = ?
{α, β, γ}=
{0.2, 0.2, 0.6}



{x,y, depth} = {243, 42, 6}
{r, g, b} = {0, 0, 1}

{x,y, depth} = {102, 13, 6}
{r, g, b} = {0, 0, 1}

a) (3 pts) What are the coordinates for the associated interpolated vertex?:

b) (3 pts) What is the interpolated color?:

d) (4 pts) Assuming the current value stored in the depth buffer/z-buffer for the associated pixel is 5.5, would the frame buffer/color buffer be updated with the new color? **Assuming the z values specified are distances measured from the camera – thus smaller values are closer to the camera.**