CS559 Computer Graphics – Spring 2016

Midterm Exam – Tuesday March 15th 2015

Time: 2 hrs

Name	
University ID	

Part #1	
Part #2	
Part #3	
Part #4	
Part #5	
TOTAL	

1. [$7 \times 4\% = 28\%$] MULTIPLE CHOICE SECTION. Circle or underline the correct answer (or answers). You do not need to provide a justification for your answer(s).

   (1) In order to construct the camera transform (e.g. via the TWGL `lookAt` method) we typically provide as inputs:

   - the eye location (a point denoted by $\mathbf{e}$)
   - the target that the camera is pointed at (this is also called the "lookAt" point, and we will denote it as $\mathbf{l}$), and
   - the "up" vector (denoted by $\mathbf{t}$).

   Using this information, a camera coordinate system is constructed, with $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ as its axis vectors.

   Which of the following statements about this construction are true?

   Write the letters of ALL correct answers here: $\boxed{}$

   (a) The "gaze" vector $\mathbf{g}$ points from $\mathbf{e}$ to $\mathbf{l}$, and is oriented exactly in the opposite direction of the coordinate axis vector $\mathbf{w}$.

   (b) The "up" vector $\mathbf{t}$ will be perpendicular to both coordinate axis vectors $\mathbf{u}$ and $\mathbf{v}$.

   (c) The "up" vector $\mathbf{t}$ is perpendicular to coordinate axis vector $\mathbf{w}$.

   (2) Consider the camera coordinate system with axis vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$, constructed as described in Question (1) above. You may assume that all three of these vectors are normalized, i.e. have unit length.
   Which of the following statements about this construction are true?

   Write the letters of ALL correct answers here: $\boxed{}$

   (a) The vector $\mathbf{v}$ is equal to the cross product $\mathbf{w} \times \mathbf{u}$.
   (Hint: use the right hand rule to check if the sign is correct).

   (b) The target point (denoted by $\mathbf{l}$, as defined in Question 1) will have the representation $(0, 0, \alpha)$ in camera coordinates, where $\alpha > 0$.

   (c) The vector $\mathbf{v}$ has exactly the same direction and orientation as the "up" vector $\mathbf{t}$ (i.e. they are parallel and point in the same direction), although their lengths might be different.

(3) Consider a point P on the same plane as a triangle ABC. We know that in this case, we can write

$$P = \alpha A + \beta B + \gamma C$$

where α, β and γ are the *barycentric coordinates* of P relative to the triangle vertices A,B and C.
Which of the following statements are true?

Write the letters of ALL correct answers here: $\boxed{}$

(a) If $\alpha = 0$ and $\beta = \gamma = 0.5$, then P must lie somewhere along the line segment BC.

(b) If $\alpha = 0.5$, $\beta = 1.5$ and $\gamma = -1$, then P is outside the triangle ABC (although still on the same plane).

(c) $\alpha = \beta = \gamma = 1$ cannot be valid barycentric coordinates, for a point P on the same plane as triangle ABC.

(4) The simple lighting model we discussed in class included three terms: (a) an ambient lighting component, (b) a diffuse reflection component, and (c) a specular reflection component.
Which of the following statements, regarding this model, are correct?

Write the letters of ALL correct answers here: $\boxed{}$

(a) The diffuse reflection depends on the surface normal and on the direction of the incoming light, but not on the placement of the camera relative to the object.

(b) The ambient component only depends on the surface normal.(c) Although we could implement this lighting model on the vertex shader, we would generally expect to achieve better detail by implementing it on the fragment shader, especially in models with relatively coarse triangle faces, and when we have significant amounts of specular reflection.

(5) In class, we discussed several transforms that are combined together in the process of drawing three-dimensional scenes. In no particular order, these transforms include:

- The projection transform T_{proj}.
- The modeling transform T_{model}.
- The viewport transform T_{vp}.
- The camera transform T_{camera}.

Imagine that we want to combine all these into a single transform T_{combined}, such that we can directly transform a point $\mathbf{x}_{\text{object}}$ in the local coordinate system of an object, to its final coordinates on the display window as $\mathbf{x}_{\text{display}} = T_{\text{combined}} \cdot \mathbf{x}_{\text{object}}$.
(you probably did this for your Programming Assignments 3 and 4)
What is the expression for this combined transform?
Note: Be careful; we are asking for the standard algebraic way of combining/multiplying transforms. TWGL *does it in a different order, for reasons we have discussed.*

Write the letter the ONE correct answer here: $\boxed{}$

(a) $T_{\text{combined}} = T_{\text{proj}} T_{\text{vp}} T_{\text{camera}} T_{\text{model}}$
[✓ (b) $T_{\text{combined}} = T_{\text{vp}} T_{\text{proj}} T_{\text{camera}} T_{\text{model}}$]
(c) $T_{\text{combined}} = T_{\text{proj}} T_{\text{vp}} T_{\text{model}} T_{\text{camera}}$
(d) $T_{\text{combined}} = T_{\text{vp}} T_{\text{proj}} T_{\text{model}} T_{\text{camera}}$

(6) Which of the following variable types can a GLSL **fragment shader** utilize?

Write the letters of ALL correct answers here: $\boxed{}$

[✓ (a) `attribute` variables.]
[✓ (b) `uniform` variables.]
(c) `const` variables.
[✓ (d) `varying` variables.]

(7) Which of the following variable types defined in a GLSL shader can be **written to** from the host JavaScript/WebGL program?

Write the letters of ALL correct answers here: $\boxed{}$

[✓ (a) `attribute` variables.]
[✓ (b) `uniform` variables.]
(c) `const` variables.
[✓ (d) `varying` variables.]

2. [28%] SHORT ANSWER SECTION. Answer each of the following questions in no more than 1-3 sentences.

   (1) [6%] In class, we have discussed both (a) the painter's algorithm and (b) the Z-buffer as possible solutions for the visibility problem. Give two reasons why the Z-buffer approach is generally considered to be the better approach of the two.

   (2) [6%] Can the following be done in the **V**ertex shader, in the **F**ragment shader, in **E**ither the vertex or fragment shader, or in **N**either of them? **(Write "V","F","E", or "N" next to each question below. You don't need to provide an explanation, unless you really feel that your answer comes with significant caveats.)**

   - Change the shape of the object being drawn, for example compressing it or stretching it along a given direction.
   - Compute lighting, based on a model that employs an ambient and a diffuse component.
   - Apply a viewport transformation.

(3) [10%] The different variable types in GLSL shaders include `attribute`, `uniform` and `varying`.

i. Give two examples of quantities that would be represented using an `attribute`.

ii. Give two example of quantities that are best represented with a `uniform`.

iii. Give one example of what kind of quantity you would use a `varying` for. Explain why this would be different than an attribute.

(4) [6%] In three-dimensional graphics programming, it is extremely common to use **homogeneous** representations for 3D vectors and transforms (i.e. 4-vectors, and 4×4 matrices, respectively). Describe two significant advantages that these homogeneous representations offer.

3. [14%] Complete the following two parts:

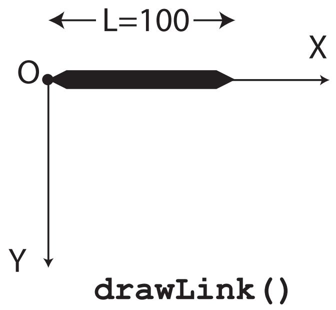
   (a) The following 4×4 matrix is the homogeneous representation of a perspective projection transform

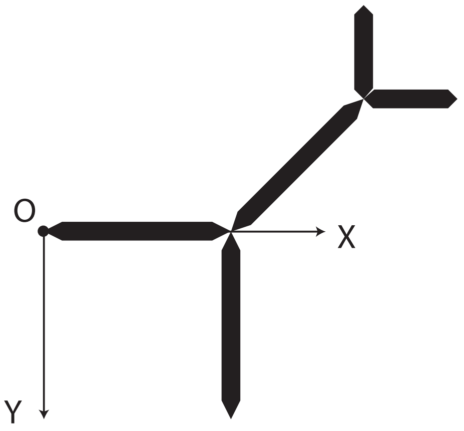
$$T_{\text{proj}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -3 & -4 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

   Apply this transform to the 3D vector $(1, 3, -2)$, and write the 3D vector that is the result of this transform.
   [Hint: You will need to convert the input vector to homogeneous representation, and do the necessary division to convert the final result back to a plain 3D vector, from the 4-number homogeneous form]

   (b) Write an expression of the **viewport transform** that would convert a point from *normalized device coordinates (NDC)* (Reminder: In NDC, the coordinates of the visible window range from -1.0 to +1.0 along each axis) to pixel coordinates on a window of your screen, with dimensions 640×480 pixels. It is perfectly fine to write this transform as the product of two transformation matrices (no need to carry out the multiplication).
   [Note: The scaling of the Z-coordinate is not important, you can do as you please. Also, it is up to you if you want to "flip" the Y-axis ... either way is acceptable.]

4. [15%] We have implemented a function `drawLink()` that draws the pointed stick-shaped object shown on the left. Subsequently we used this function to draw the more complex shape shown on the right:





This was done via hierarchical modeling, as shown in the code below:

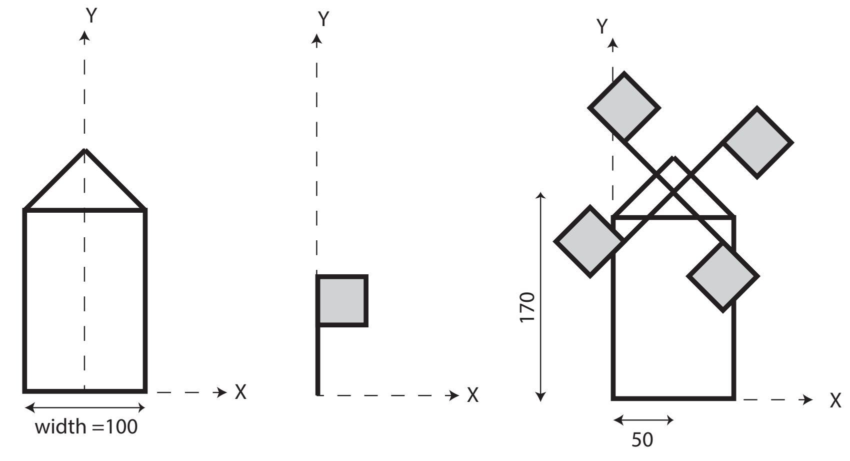
```
context.save();
drawLink();
context.translate(100,0);
context.save();             // Line "A"
context.rotate(-Math.PI/4);
drawLink();
context.translate(100,0);
context.save();
context.rotate(Math.PI/4);
context.scale(0.5,1);
drawLink();
context.restore();
context.save();
context.rotate(-Math.PI/4);
context.scale(0.5,1);       // Line "B"
drawLink();
context.restore();          // Line "C"
context.restore();
context.save();
context.rotate(Math.PI/2);
drawLink();
context.restore();
context.restore();
```

(a) Draw the shape that the code above would produce, if we removed lines marked “A” and “C”.

(b) Draw the shape that the code above would produce, if we removed just the line marked “B”.

Hints: Since the Y-axis is pointing down, positive angles will be oriented *clockwise*, and negative ones *counter-clockwise*. Also, a friendly reminder that `PI/4`=45 degrees.

5. [15%] We want to create a 2D model of a windmill, as illustrated on the rightmost drawing in the figure below.



DrawTower() **DrawSail()** Final Windmill Model

Let's assume that we have been given the a routine `DrawTower()` that draws the main building (tower) of the windmill, as seen on the drawing on the left, and another routine `DrawSail()` that draws just one of the windmill's sails, as seen in the middle.

Write a program, in pseudo-code, to draw the windmill using a hierarchical modeling approach. Your code can call the functions `DrawTower()` and `DrawSail()` when needed, and you can also use the Canvas-like `save()` and `restore()` functions to manipulate the transform stack. You can use functions like `translate(x,y)` and `rotate(angle)` to multiply the current (top-of-stack) transformation with the respective translation/rotation.

Notes/Hints:

- You will not need to scale either the tower or the sail(s); they are already at the size they appear in the model. You will just need to translate and rotate.
- The fan of the windmill is mounted at location (50,170), as shown. Note that the tower has been shifted so that one of its corners is now at the origin.
- Note that the fan has been rotated 45 degrees, away from the configuration that would have all sails be horizontal or vertical.