IGME 740 Spring 21 - Midterm Exam

Name: Trenton Plager

1. (6 points) Explain the difference between **vector graphics** and **raster graphics**.

Vector graphics are based on vectors and are defined by geometrical primitives and math. Therefore, they are scalable, but they cannot be displayed. Raster graphics are made of pixels. Vector graphics cannot be displayed until they are converted to raster graphics through rasterization.

2. (4 points) When a program assigns a total of 8 bits to a pixel, that pixel can display up to _____ colors.

A. 32 B. 64 C. 256 D. 1024 E. None of them

3. (7 points) Given a pixel represented as a triple of float color values `(0.3, 0.5, 0.8)`, convert this pixel to byte color values. Note that the range of a byte color value is `[0, 255]`, and the range of a float color value is `[0.0, 1.0]`. Round your answer up to the nearest whole number.

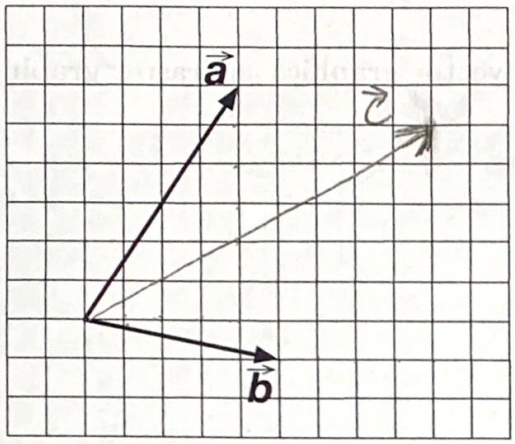
$\frac{0.3}{1.0} = \frac{r}{255}$ $\frac{0.5}{1.0} = \frac{g}{255}$ $\frac{0.8}{1.0} = \frac{b}{255}$

$r = 77$ $g = 128$ $b = 204$

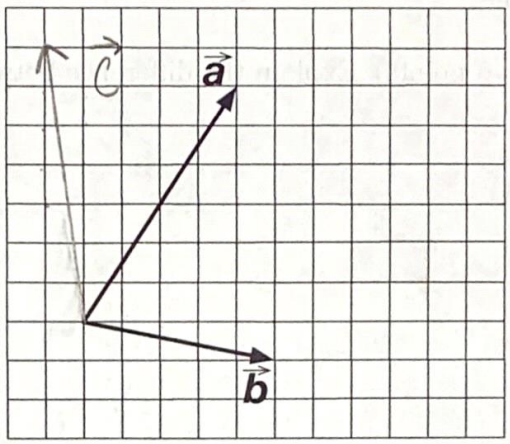
(77, 128, 204)

4. (6 points) Given two vectors $\vec{a}$ and $\vec{b}$ as shown in the figures below, draw the vector $\vec{c} = \vec{a} + \vec{b}$ in the left figure, and draw the vector $\vec{c} = \vec{a} - \vec{b}$ in the right figure.

$a = (4,6)$
$b = (5,-1)$



(9, 5)

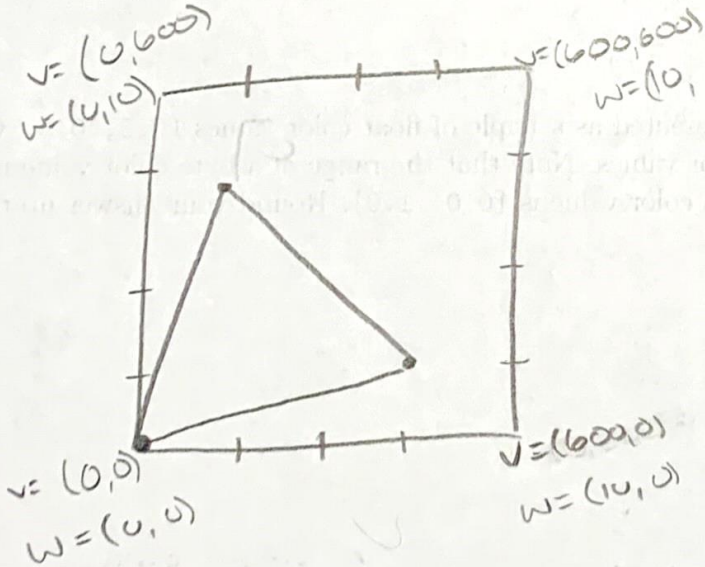


(-1, 7)

5. (7 points) Given the following OpenGL code statements:

```
gluOrtho2D (0, 10, 0, 10);
glViewport (0, 0, 600, 600);
```

Assume that a 2D triangle has been displayed on the OpenGL's raster window, where the triangle's vertices have been rasterized to the pixel locations: `(0,0)`, `(450,150)`, and `(90,400)`. Compute original coordinates of the triangle's vertices.



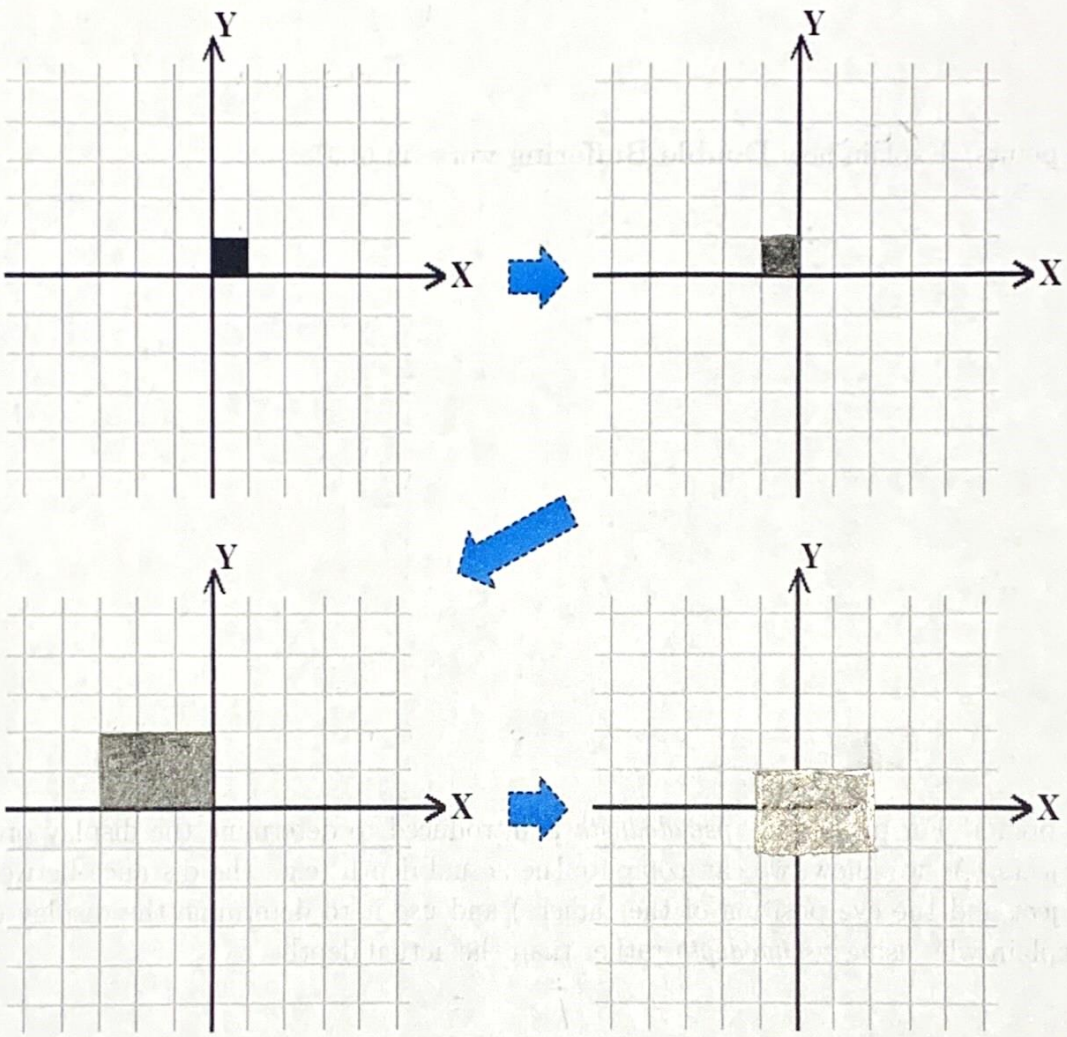
$6.\overline{6}$

Original Coords. = $(0,0), (7.5, 2.5), (1.5, 6.67)$

6. (9 points) A polygon is defined with the points $(0,0)$, $(1,0)$, $(1,1)$ and $(0,1)$. The shape of the polygon is shown in the first figure. In the rest of three figures, draw the shapes obtained after transforming the polygon with the provided matrices. (Note: the order of matrix multiplication to transform the polygon is **right-to-left**.)

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} cos(90°) & -sin(90°) & 0 \\ sin(90°) & cos(90°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

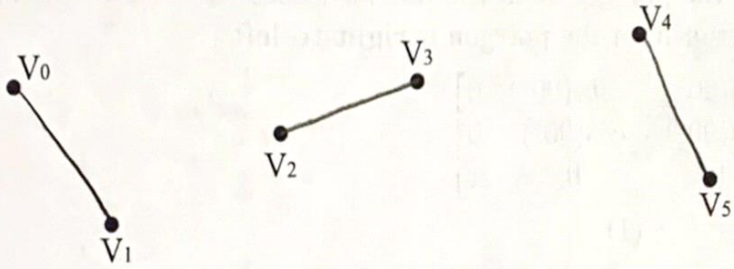
(3) (2) (1)



7. (6 points) Describe in one sentence what the command `glPushMatrix()` does.

glPushMatrix pushes the current transformation matrix to the matrix stack, essentially saving it so that a new one can be edited and then popped off, returning to the pushed one.

8. (6 points) Given a sequence of vertices $\{v_0, v_1, \cdots, v_5\}$, as shown in the figure below, draw the lines constructed from those vertices when using the drawing mode of `GL_LINES`.



9. (6 points) Explain how **Double Buffering** works in `GLUT`.

When using Double Buffering, there are 2 color buffers being utilized. At any given time, one is being displayed while the other is being written to. Once rasterization is finished on the buffer being displayed, the buffers are swapped and the new drawing is now displayed while a new drawing is being written. This is useful for processes like animation.

10. (6 points) For projection, *pseudodepth* is introduced to determine the display order of objects. As we know, we can compute the actual depth (e.g., the distance between an object and the eye position of the camera) and use it to determine the display order. Explain why using *pseudodepth* rather than the actual depth.

Pseudodepth is much less computationally expensive than computing the actual depth because it doesn't involve squares and square roots. It also maintains visual quality well until far distances from the near plane, but at this distance the detail and visual fidelity is much less important.

11. (10 points) Given a camera defined in the world space, the camera's eye position is defined as $eye = (2, 0, 0)$, and its lookAt position os defined as $lookAt = (4, 1, 2)$. Compute the three axes, $\vec{u}$, $\vec{v}$,and $\vec{n}$, of the camera coordinate system. (Assuming the up vector is y-axis. $\vec{u}$, $\vec{v}$ and $\vec{n}$ must be normalized vectors.)

$$\vec{n} = eye - lookAt$$

$$\vec{n} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ -2 \end{bmatrix} \qquad \hat{n} = \begin{bmatrix} -2/3 \\ -1/3 \\ -2/3 \end{bmatrix}$$

$$\vec{u} = up \times n$$

$$\vec{u} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} -2 \\ -1 \\ -2 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix} \qquad \hat{u} = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$

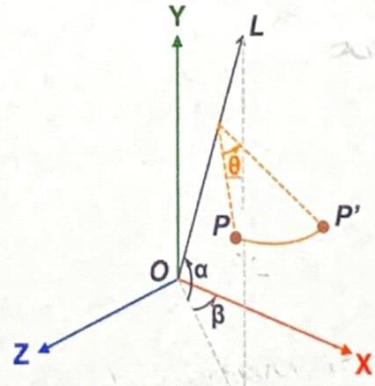
$$\vec{v} = n \times u$$

$$\vec{v} = \begin{bmatrix} -2 \\ -1 \\ -2 \end{bmatrix} \times \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ 8 \\ -2 \end{bmatrix} \qquad \hat{v} = \begin{bmatrix} -1/3\sqrt{2} \\ 2\sqrt{2}/3 \\ -1/3\sqrt{2} \end{bmatrix}$$

12. (9 points) The figure below shows the rotation of point P by θ degrees about the arbitrary vector L. Write the transformation sequence that rotates P to P', and briefly explain how you find out the answer. You don't need to write 4×4 matrices, you only need to use T, R, S symbols as needed to represent translation, rotation and scaling, respectively.

Symbols should look like these examples:

- $T_{(\Delta x, \Delta y, \Delta z)}$ – translate Δx, Δy, Δz along the axis x, y, z, respectively.
- R^z_α – rotate α about the axis z.
- S^y_d –scale d on the axis y.

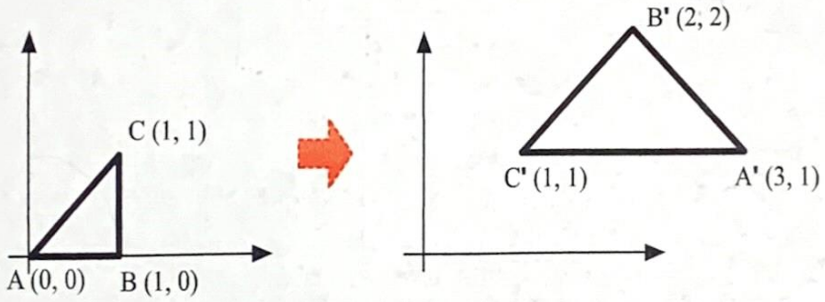


R^L_θ. Because it is rotating about the vector L, the answer should include only rotation since rotation can be defined as rotating some amount of degrees about an axis (or, in this case vector). Since it is not rotating about a specific point, no translation is involved.

13. (6 points) GLUT uses callback functions. Explain when the reshape callback is called.

The reshape callback is called when the window is reshaped by resizing, exposing, or minimizing it.

14. (12 points) Given a triangle ΔABC shown on the left of the figure, we want to transform it into the triangle $\Delta A'B'C'$ on the right. Write a sequence of 3×3 homogeneous transformation matrices that represent all steps of the transformation, and compute the composed matrix.



$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(135) & -\sin(135) & 0 \\ \sin(135) & \cos(135) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \sqrt{2} & 0 & 3 \\ 0 & \sqrt{2} & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(135) & -\sin(135) & 0 \\ \sin(135) & \cos(135) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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