CSCI 4631 / 5631 Midterm Exam Name Edward Garcia

101/110 = 92% + 15 → 107

1) The GLUT main event loop is essentially an infinite loop – explain what happens during the execution of this loop **(8 points)**

The GLUT main event loop occurs after initialization code and at least one subscribed event (the display event).

GLUT main event loops runs various events related to keyboard, mouse, display, idle, and other events infinitely until told to break (exit(0)).

what if no callback has been registered for that event?

– 1

2) GLSL is the language that allows us to reprogram our shaders. What two types of shaders can be reprogrammed, and what sort of data in the pipeline does each work on? (6 points)

- Vertex Shader

  Attributes of a vertex such as color, size, etc. "Attribute" Variables

- Fragment Shader

  Attributes of a potential pixel. "Varying" Variable

Both can use "Uniform"

3) True or False **(2 points each)**

F A) Points are defined in vector spaces (No location in vector space)

F B) Vectors have location and direction (Magnitude not location)

T C) You should not explicitly execute your display callback in your code (should use GLUTPostRedisp

T D) An affine space with an origin in called a **frame**

4) Matching **(3 points each):**

~~A) Rotation about the X axis~~

~~B) Rotation about the Y axis~~

~~C) Rotation~~ about the Z axis

~~D) Translation~~

~~E) Scaling~~

~~F) Reflection about the Z axis~~

~~G) Shear in X~~

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

G $\begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

C $\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

A $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

D $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

E $\begin{bmatrix} Bx & 0 & 0 & 0 \\ 0 & By & 0 & 0 \\ 0 & 0 & Bz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

B $\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

✓

5) **Rigid body** transformations and **affine** transformations **(5 points each)**:

A) What properties are guaranteed not to vary in **rigid body** transformations?

Angles and Lengths
(implying parrellel"icity?")

B) What kinds of transformations can be combined to produce **rigid body** transformations?

Translations and Rotations

C) What properties are guaranteed not to vary in **affine** transformation?

Parrallel Property of Lines.

Angles and Lengths may change.

D) What kinds of transformations can be combined to produce **affine** transformations?

Translation, Rotatitions, Scale, Sheer

6) You're given a primitive whose centroid is at point P {10.0, 3.0, 6.0, 1}. You want to rotate this primitive about a vector starting at the centroid that is parallel to the world X axis, through an angle of 45°, then scale the primitive by ½ in X, Y, and Z.

A) What set of transformations must you perform to accomplish this in a stepwise manner? Use **shorthand** matrix notation. **(5 points)**

$$M = T R S T^{-1} = (T(R(S(T^{-1}))))$$

but what specifically is this T? $T_{(-10.0, -3.0, -6.0)}$

ie. Translate to origin, scale, rotate, and translate back.

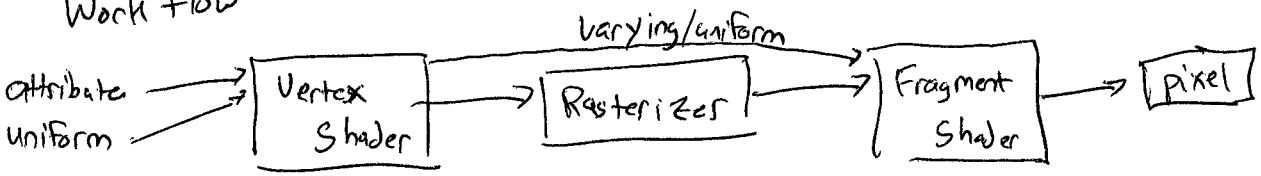
B) Is the composite transformation rigid body? Affine? Why or why not? **(5 points)**

(RigidBody) No, scaling does not perserve angles and lengths.

(Affine) Yes, Parralel lines are perserved.

7) In GLSL, there are 3 types of variable qualifiers.... **attribute**, **varying**, and **uniform**. What do these mean? (9 points)

Work Flow



attribute (i.e. IN) - goes into a vertex shader and be changed once per vertex.

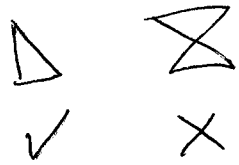
varying (i.e. OUT/IN) - goes into a Fragment shader
vertex fragment

uniform (i.e INOUT) - is a variable passed by APP (not shown) and is used by both vertex shader and fragment shader. Cannot be changed by shader.

8) OpenGL will be sure to render a polygon correctly if it is both *simple* and *convex*.

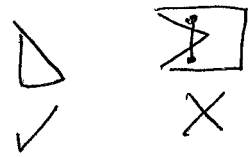
What does it mean for a polygon to be "simple"? (**2 points**)

✓ Polygon's Lines don't cross.



What does it mean for a polygon to be "convex"? (**2 points**)

✓ All possibles points in polygon produces a line entirely in polygon.



What is the simplest polygon we can draw that is both simple and convex? (**2 points**)

✓ Triangle

9) Name three different OpenGL primitives and draw examples. (**3 points**)

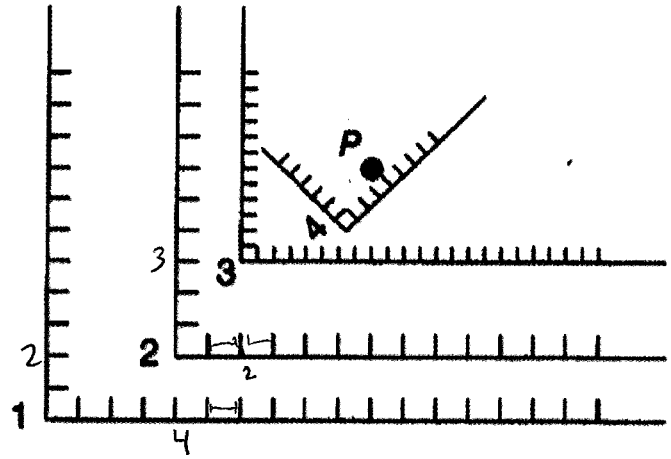
• Point

Line

triangle

✓

10) What transformation must we use…(show me the M in a homogenous coordinate transformation matrix) **(3 points each)**



A) to map from coordinate system 1 to get to coordinate system 2? $M_{2\leftarrow 1}$

$$M_{2\leftarrow 1} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

-1

B) From 2 to 3?

$$M_{3\leftarrow 2} = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 2 \\ 0 & 0.5 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

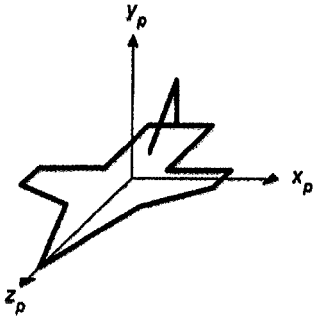
-2

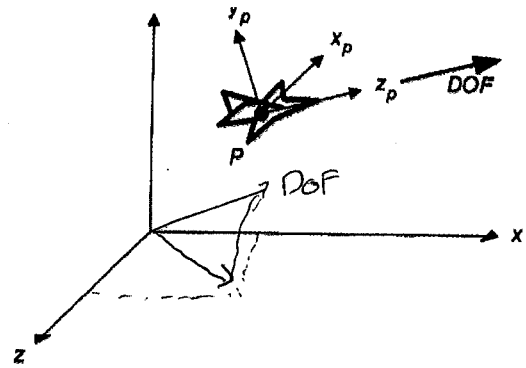
C) From 1 to 3?

$$M_{3\leftarrow 1} = \begin{bmatrix} 0.5 & 0 & 2 \\ 0 & 0.5 & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} = M_{3\leftarrow 2} M_{2\leftarrow 1} = \begin{bmatrix} 0.5 & 0 & 6 \\ 0 & 0.5 & 5 \\ 0 & 0 & 1 \end{bmatrix}$$

-1

Bonus 1 (Required for Grad Students): (10 points): I explained some properties of special orthogonal matrices that allow us to construct transformations quickly (and easily) as a change of coordinate systems. For example we've got a plane that we want to transform such that it is pointed in some direction DOF, and is not banked. How can we build a transformation matrix to accomplish this?





Because they are special orthogonal, the angles are additive, Hence we can do a rotation per axis

$$R_{DOF} = \cancel{\begin{bmatrix} \{DoF_x\} & \{DoF_y\} & \{D & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos DoF_x & \sin DoF_x & 0 \\ 0 & -\sin DoF_x & \cos DoF_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos m & \sin m & 0 & 0 \\ \sin m & \cos m & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot M_{R_z}$$

$\boxed{DoF_{\theta_x}}$ $\boxed{DoF_{\theta_z}}$

Then the transform for entire plane-frame is

$$M = T R_{DOF} S T^{-1}$$

-3

PS - Other properties of special orthogonal I missed, each vector in upper ~~right~~ left 3x3 must be a unit vector

Bonus 2 (10 Points Total – BONUS FOR EVERYONE!!! YIPPEEE!!!!)

A) (3 points) Why do we go to the trouble of using a homogenous coordinate representation?

Two reasons... reduce the type of operations that must be supported and the ability to concatenate all transforms into one data structure. 4x4 matrix.

✓ +3

B) (7 points) What are the properties of a special orthogonal transformation matrix?

$$\begin{bmatrix} a & b & c & d \\ e & F & S & g \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The topleft 3x3, each row must be a unit vector, (ie magnitude of 1)
- Each column vector must also be unit vectors.
- Each vector from above must be perpendicular to each other.

+7

✓

+10

Totally Random Bonus (ALSO FOR EVERYONE) (5 points).

What is the first stanza of the poem Jabberwocky from Lewis Carroll's "Through the Looking-Glass, and What Alice Found There"? (In the spirit of Jabberwocky, I may give you partial credit for a well-crafted nonsense poem of your own design)

My ~~[illegible]~~ knowledge of literature is bad
Which, on the outside is sad.
But my life is still fufilled
For I ~~can~~ have other areas which I am skilled.

+5

+5