10/13/2020

The Midterm is delayed.

It will be posted Early Tuesday 10/20 in the Morning (maybe Monday late night)
It will be due on Tuesday (10/20) at 11:55 pm.
It will cover the same material planned for this week midterm.

No Class meeting on Tuesday 10/20.

Up to and including the material that is available in Slides CG-1 to CG13
That have been covered in class.

Material “skipped” will not be included.

The Midterm be given with emphasis on The OGL side.

CG 10 ignore 18, 20 and above

CG11 ignore:
3, 4 6, 10 7 15 and above

CG12
Need to know the basics excluding slides 20 and above

CG13 up to slide 19

Plan for today

Review slide sets CG-12 CG-13

CG-10 CG-11 go to skipped material.

There will be a problem-solving assignment related to CG-10 to CG-13.
This material (and some of the questions) are discussed in TRACS/Resources/Shoaff-Material/Math.pdf.

There was material skipped which you still have to study.

If we need to perform transformations A, B, C,.. in sequence we can
Concatenate and generate M=A*B*C

This will save “time”
Might reduce artifacts

C is the first to be applied to vertices (post multiplication)

Assume V is vertex $[Vx, Vy, Vz, 1]^T$ a 4x1
A, B, and C are 4x4

M=A*B*C → M*V means A*B*C*V = (A*(B*(C*V)))

The order is important A*B is not necessarily equal to B*A
Covered in slides 18, 19 20 not for the midterm.

20 is discussing general rotation

21 to rotate (scale) around a fixed-point P’

Translate so that P’ is in the origin, Rotate, Translate back

$M=T()*R()*T^{-1}$
Applying M to a vertex V means that T^{-1} is applied first
So we actually need to do
$M=T^{-1}*R*T$

Skip 22, 23, 24.

CG-13

In S2 you had to apply glFrustum on the projection matrix.

Trial and error with different paprameters.

Select the projection matrix
Load identity
glFrustum (l, r, b, t, n, f).
Alternatively, gluProjection () macro using glFrustum and is more intuitive

CTM is one matrix representing P*M (assuming that the order in the slide is correct and that the vertices are first post multiplied by M then by P)

Model view

Slide 5 operations that can be done on P or M. Some make more sense for M.

C ← M (load M) and C←C*M is the only OGL 3+ instructions.

Vertices and vectors are represented as $[x, y, z, w]^T$ (4x1 matrix)
Transformations are represented as 4x4 matrices.

Additional Operations on the CTM,

Slide 12.

You can store the CTM in matrices. And manage a data
structure that holds these matrices.

For example you can manage them in FIFO
Or in your own LIFO
For LIFO you can use OGL push and pop.

glPushMatrix()
Pushes the matrix selected (e.g., M or P) into the OGL LIFO (stack)
glPopMatrix()
Pops the matrix

Consider light sources (require definition vectors and objects)
Any transformation applied to M
Affects all the vertices and all the vectors.

Consider a scene with light sources and
objects and moving camera.

1) We want the light sources and the objects to move tighter
2) We want the light sources and the objects to move in different direction
3) We want to move light sources and keep objects fixed
4) We want to move objects and keep light sources fixed

Move means transformed.

We may need to push and pop matrices in different orders for different options.

In the assignment.

You have to rate object A by 45 degrees.

And B by 25 degrees.

Assume:
glRaotate (45) for A
glRotate (25) for B

B is rotated by 70

So what to do?

Load identity back to square 0.
glRotate(A,45)
Load identity
glRotate(B,25)

Assume any current CTM CTM1

Push
Apply the A transformation.
Pop (back in CTM1)
Push
apply the B transformation
pop (back to CTM1)

slides 15 and above provide an example of an interactive program
that uses mouse interaction to rotate a cube.
Not for the midterm but can use as a reference for assignments or projects.

For the midterm, you need to know how to interact
with the KBD and the mouse.
Skip slide 20 and above.

Moving the Camera.

No functions in OGL

Camera is at (0, 0, 0, 1) cannot move.
Pointing to the -Z direction.

To get the effect of moving the camera we move objects.

In my setting to see pictures behind me. I would
translate them to be in front of me and rotate.
This can be done by Model view transformations or using gluLookat
Which does model view transformations as in S2.

Slide 13, Read the contents of a (transformation) matrix.

Applying transformations to vectors and vertices
Hence, [4x4]*[4x1] → transformed [4x1]

The CTM (M*P) it is a 4x4 matrix
The OpenGL functions from CG13 (used S2)
(1) glTranslate, (2) glRotate, (3) glScale
And the OpenGL projection transformations
(4) glOrtho and (5) glFrustum
Generate a [4x4] matrix
(maybe as simple as the identity matrix)

And this 4x4 matrix is multiplied by M (1, 2, and 3) or by P (4, 5)
[4x4]*[4x4] → 4x4 new P, New M –> New CTM
The CTM [4x4] is applied to each vertex on the scene before rendering
[4x4]*[4x1] for each vertex and this yields a new [4x1].