10/06/2020

The midterm will be posted on TRACS on October 15 or "real early" on October 16.
It will be due on October 16, 2020 at 11:55 pm.

It will be open material (books, notes, internet etc.) with emphasis on demonstrating understating of questions and solutions; showing all the details of the solutions.

The midterm will cover everything up to Open GL Transformations as in S1, and S2. Covered in slides CG1 to CG13 with emphasis on material covered more thoroughly from these slides. There are a few slide sets not covered in class but included.

Open GL Programming and Plotting – CG1 – CG6 + material on plotting form class
CG07, 08, 09 assigned to you for your own study (only partially covered in class)
Geometry slides CG10
Representation CG11
Transformation CG12
OGL Transformation CG13

Assignment S2 will be due on October 11, 2020. Solutions published promptly so that they are available for the midterm.

Only the circle has to be rendered using the parametric equation.

Interested in Rotation, Translation, Scaling via matrix operations
In homogeneous coordinates.
Mainly multiplication of a 4x4 transformation matrix by the 4x1
representation of a vertex.

Affine transformations applied to rigid bodies.
Preserve lines ➔ can be applied to vertices and preserve
strait line edges between vertices (not necessarily angles)

The Current Transformation Matrix CTM

Open GL has several basic matrices.
Two that are of interest to us are the Projection (P) and the Model View (M) matrices.

Each vertex on the scene is post multiplied by P and M before been rendered.

Consider Vertex $V_0 = [x_0, y_0, z_0]^T$ (normalized point in homogeneous coordinates)

Before rendering, V_0 is being transformed according to the current P and the current M

We do $P \times V_0$ and $M \times V_0$ before rendering the vertex V_0.

Slide 5: Every vertex is going through transformation ($P \times V_0$ and $M \times V_0$)
Before "getting" to the frame buffer.

To avoid the need to multiply every vertex twice (By M then by P)
OGL maintains one current transformation matrix (CTM) CG13-4
CG13 – 8 The CTM is $M \times P$ updated by OGL "on the fly."

In CG12 we mainly consider transformations on M.
Translation scaling and Rotation.

The open GL Model-View transformation functions (glTranslate(), glScale(), glRotate())
Result in: CTM ← CTM*T, CTM← CTM*S, CTM ← CTM*R
OGL produces the appropriate 4x4 matrix (using your function parameters)
Post multiplying by M thereby post multiplying by the CTM

The open GL projections transformation result in:
CTM ← CTM*[Parallel projection matrix], CTM*[Perspective projection matrix]
glOrtho() – produces a parallel projection matrix based on L, R, B, T, N, F; given by
the programmer and post multiply it by P thereby post multiplying by the CTM

glFrustum() – produces a perspective projection matrix based on L, R, B, T, N, F; given by
the programmer and post multiply it by P thereby post multiplying by the CTM

One more important function is the load identity which basically resets the
CTM to I (default) meaning that vertices are not changed by the CTM

Note the for the S2 assignment you may want to consider using the
glLoadIdentity() between the rendering and transformation of different objects (CG-13-9).

Alternatively you can push and pop the M or the P (going from one version of the CTM to another in LIFO). Additionally, you can store the CTM in a matrix then load it.
To go from one state of transformation to another then back.

Consider:
Rotate O1
Translate O2

Can be done via:
Load identity
Rotate O1
flush
Load identity
Translate O2
Flush

Or:
Load identity
… perform any transformation
Push
Rotate O1
Pop

Push
Translate O2
Pop

The push and pop save and retrieve the CTM transformation in LIFO manner.

Translation

CG12-7-10 The matrix is given in slide 10
In a previous class session we have multiplied a specific
translation matrix by a specific vertex.

Assume translation by [dx, dy, dz] with translation matrix T as in 10.
How do we obtain the inverse matrix T^{-1} from T?
Use linear Algebra methods (e.g., using the cofactor matrices)
Use the same matrix with [-dx, -dy, -dz]
Inverse translation: if T translate by D then T^{-1} should translates by -D.

Rotation (of vertices).
Slide 11

Given a vertex at location (x, y) where the vector from origin to (x, y)
Is at an angle of ϕ with the X axis.
We rotate the vertex by an angle of θ and obtain a new vertex at
location (x', y')
The vector from the origin to (x', y') is at an angle of $(\phi + \theta)$.

The distance to the origin is fixed (R). The angle WRT to the X axis
is changed from ϕ to $\phi + \theta$
x, y are given by $r \times cos(\phi), r \times sin(\phi)$
x' y'; are $r \times cos(\phi + \theta), r \times sin(\phi + \theta)$
Result in slide 12, producing the matrix of slide 13.

Inverse rotation by θ – get the matrix with $-\theta$.
For inverse scaling by S_x, S_y, S_z, use: $1/S_x, 1/S_y, 1/S_z$