Lecture 4

Interactive 3D Control

(optional material, not in exam)

Lecture outline:

1. Interactive Control Concept
    - Mouse
    - Modelview Matrix: TRS
2. Interactive Translation relative to screen
3. Interactive Rotation: Rolling Ball

Interactive Control Concept

Device: Mouse

Important: Position! Position! Position!

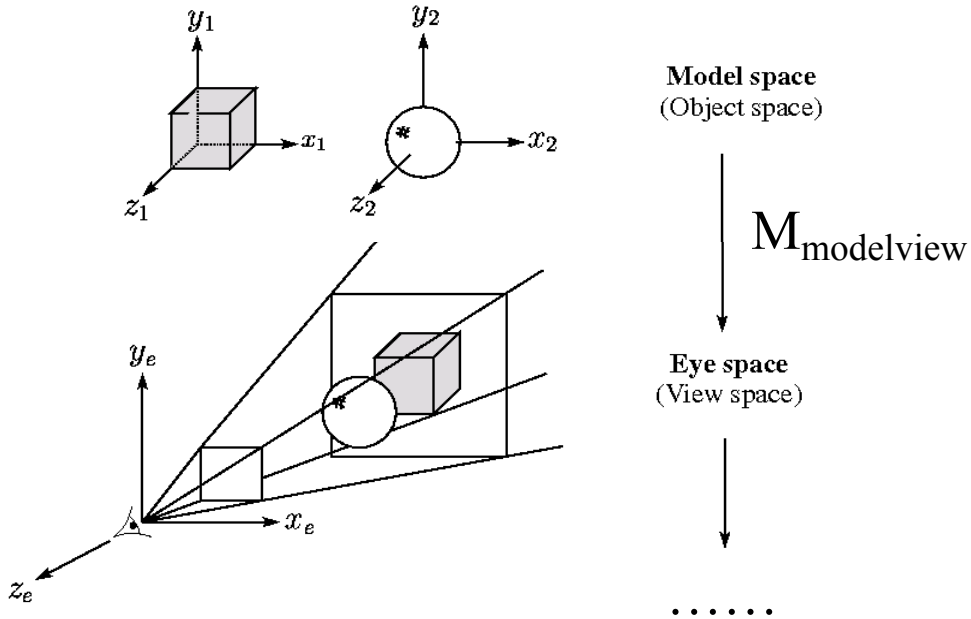
Mouse In 3D control:

1. It is a 2D input device for 3D viewing control

Interactive Control Concept

Previous lecture: 4x4 Modelview matrix

- Store the transformation from world to eye space
- Eye at the eye space origin



Interactive Control Concept

Approach:

By modifying the 4x4 Modelview matrix incrementally accordingly to the mouse motion, we can control our 3D viewing interactively

Note: we are modifying the base Modelview matrix. When we draw the scene with other transformations, the objects are drawn relative to this modelview matrix.

Interactive Control Concept

Normal Modelview Transformation Equation:

Assume the Modelview matrix is affine. In particular, it has translation, rotation, and scaling only. If so, we can write the Modelview transformation like this:

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix}$$

Note: if the matrix has T and R only, the 3x3 submatrix is orthonormal

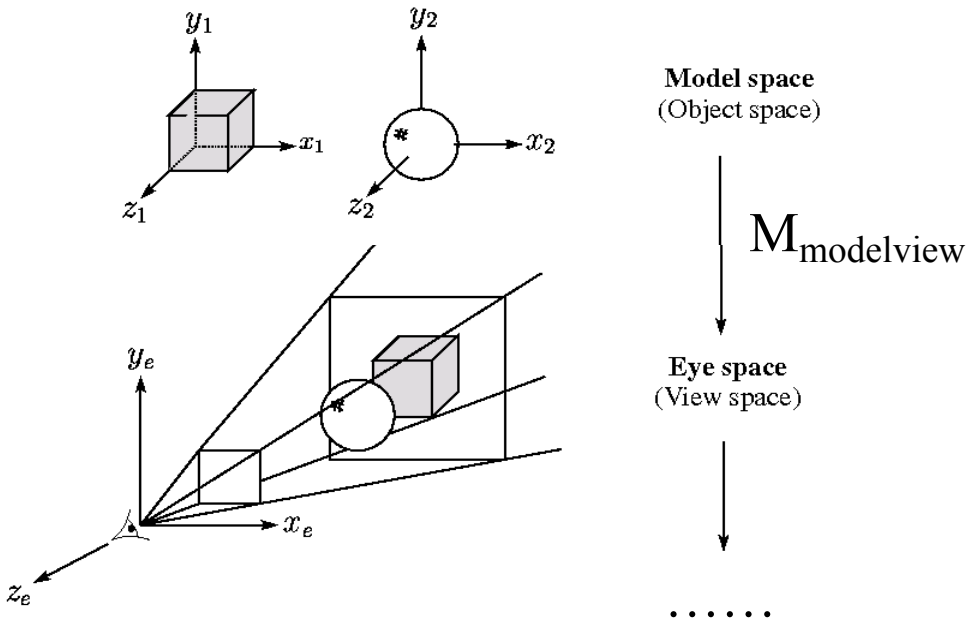
Interactive Translation

Left-Multiply a translation matrix to the existing modelview

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix}$$

Multiply a translation matrix here

We can have screen-aligned translation!



Interactive Translation

OGL Implementation (Note: we need Left-Multiplication)

```
// Mouse XY Translation mapped to screen XY Translation
tx = 0.01 * dx ;
ty = 0.01 * dy ;
glGetFloatv ( GL_MODELVIEW_MATRIX , mat ) ;
glLoadIdentity () ;
glTranslated ( tx , ty , 0.0 ) ;
glMultMatrixf ( mat ) ;
```

Note: dx and dy are changes in mouse coordinates when dragging

Interactive Rotation

Left-Multiply a rotation matrix to the existing modelview ???

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix}$$

Multiply a rotation matrix here?

Interactive Rotation

Not really!!! We need to use fixed point rule!

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix}$$

1. Undo the Translation (T_x,T_y,T_z) in Modelview matrix
2. Multiply a rotation matrix here?
3. Redo the Translation (T_x,T_y,T_z)

Interactive Rotation

Furthermore, how to construct the rotation matrix?

we need a method to map 2D mouse motion to 3D rotation!

Interactive Rotation

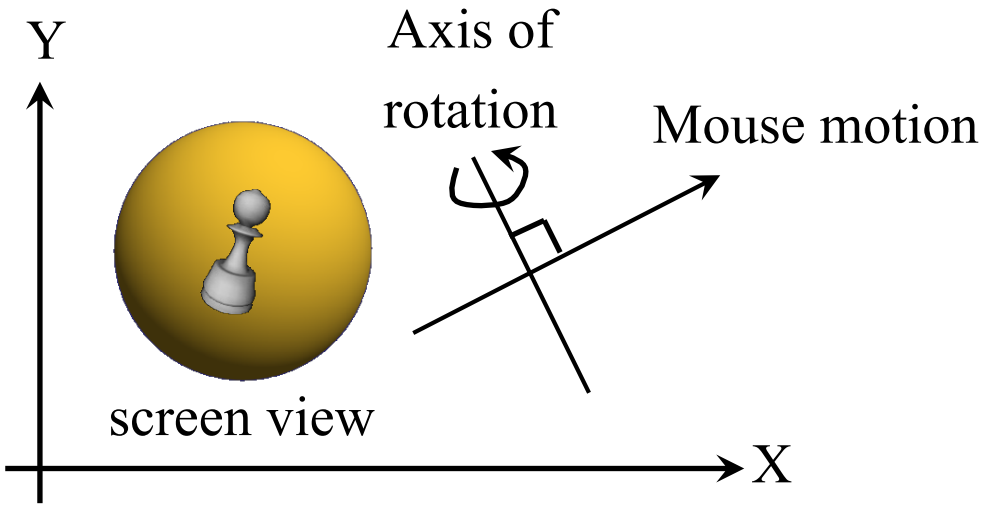
One popular method is the Rolling Ball.

Idea: Imagine the object being studied is enclosed inside a Glass Ball of radius R

Interactive Rotation

Constructing the Rotation matrix:

1. Put the mouse motion vector in XY eye space: (dx , dy , 0)
2. Axis of rotation perpendicular to the motion vector: (-dy , dx , 0)
3. Angle of rotation relative to motion vector length: sqrt ($dx^2 + dy^2$)



Interactive Rotation

OGL Implementation (Note: Left-Multiplication with fixed-pt)

```
// Rotation
nx    = -dy ;
ny    =  dx ;
scale = sqrt ( nx * nx + ny * ny ) ;
if ( scale > 0.0 )
{
        glGetFloatv ( GL_MODELVIEW_MATRIX , mat ) ;
        glLoadIdentity () ;
        nx    = nx / scale ;
        ny    = ny / scale ;
        angle = scale * ROTSCALE ;

        glTranslated  (  mat[12] ,  mat[13] ,  mat[14]  ) ;
        glRotated     (  angle   ,  nx      ,  ny , 0.0 ) ;
        glTranslated  ( -mat[12] , -mat[13] , -mat[14]  ) ;
        glMultMatrixf ( mat ) ;
}
```

Note: Modelview Matrix in OGL is stored as column major and mat[12-14] tells us the current translation.

Summary

- Interactive 3D control by incremental update of the Modelview Matrix
- Left-Multiplication to the Modelview Matrix changes the eye space coordinate (relative to the screen)
- Rolling Ball is a context-free:
  - independent of the initial mouse coordinates