Hierarchical modeling

Estelle Duveau - estelle.duveau@inria.fr

MoSIG1, Introduction to Computer Graphics, 04/03/2009

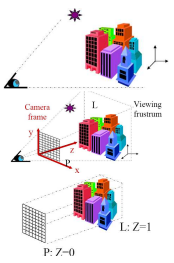
Planning

- **Lecture :** introduction to OpenGL
  Lab : first steps in OpenGL and modeling - 25/02/2009
- **Lecture/Lab :** transformations and hierarchical modeling - 04/03/2009
- **Lecture :** lights and materials in OpenGL - 11/03/2009
- **Lab :** lights and materials in OpenGL - 18/03/2009
- **Lecture :** textures in OpenGL
  Lab : textures in OpenGL - 25/03/2009
- **Lab :** procedural animation - 01/04/2009
- **Lab :** physical animation : particle systems - 08/04/2009
- **Lab :** physical animation : collisions - 22/04/2009

Plan

Plan

Graphics Pipeline

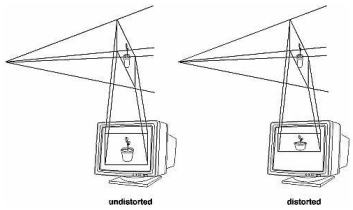


2. Build the scene from instances of models placed in a world frame (modeling transformation)
3. Convert to camera frame (culling, frustrum)
4. Convert to screen frame (projection)

Viewport

Determine how large you want the final image to be :

```
glViewport(x, y, width, height)
```



Representation of transformations

- 4x4 **matrices**
- Affine transformations : scale, rotation, translation
- Composition of transformations ⇒ Multiplication of matrices
- Not commutative

Plan

Plan

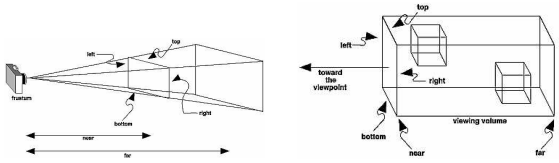
Matrices in OpenGL

- Predefined matrices :
  `GL_MODELVIEW`, `GL_PROJECTION`, `GL_TEXTURE`, ...
  → Current matrix defined with `glMatrixMode(...)`
  ⇒ M_O
- Load :
  `glLoadIdentity()` `glLoadMatrixf(M)`,
  `glLoadTransposeMatrixd(N)`
  ⇒ $M_O = I$, $M_O = M$, $M_O = N^T$
- Arithmetic :
  `glMultMatrixd(P)`, `glMultTransposeMatrixf(Q)`
  ⇒ $M_O = M_0 P$, $M_O = M_0 Q^T$
- Transformations :
  `glTranslatef(1.0,2.0,-1.5)`,
  `glRotated(90.0,0.0,1.0,0.0)`,
  `glScalef(2.0,-0.5,1.0)`

Plan

PROJECTION matrix

- Mode : `GL_PROJECTION`
- Perspective :
  `glFrustrum(left,right,bottom,top, near,far)`
- Orthographic :
  `glOrtho(left,right,bottom,top, near,far)`



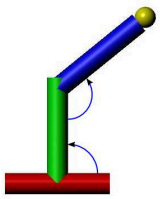
Plan

MODELVIEW matrix

- Mode : `GL_MODELVIEW`
- To position and orient a model
- Most common way to modify it :
  - Translation : `glTranslatef(1.0,2.0,-1.5)`
  - Rotation : `glRotated(90.0,0.0,1.0,0.0)`
  - Scale : `glScalef(2.0,-0.5,1.0)`
- Multiple use of objects

Plan

Principle

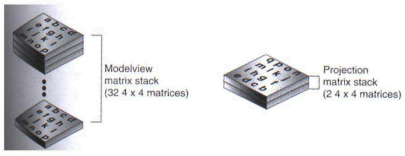


- Transformation of one object with respect to another instead of absolute transformation in world reference frame
- **Modeling :** easier location in space
- **Animation :** no need to animate all objects at all time steps

Plan

Matrix stack

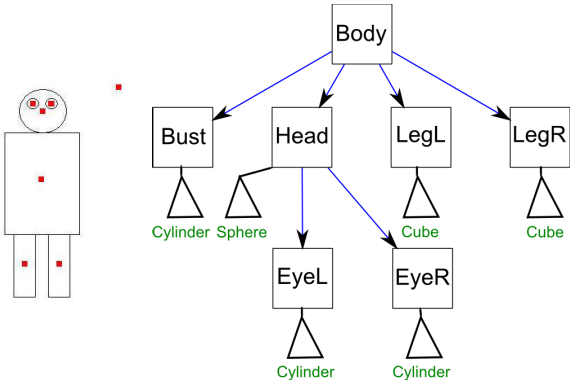
All operations done on the current matrix, but need to manipulate several matrices
⇒ two stacks of matrices (one for MODELVIEW, one for PROJECTION)



- The current matrix is the one on top of the stack
- `glPushMatrix()`, `glPopMatrix()`

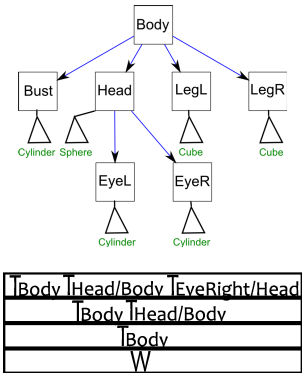
Plan

Graph scene - Example 1/2



Graph scene - Example 2/2

```
glPushMatrix()
glLoadMatrix(T_Body)
glPushMatrix()
glMultMatrix(T_Bust/Body)
drawCylinder()
glPopMatrix()
glPushMatrix()
glMultMatrix(T_Head/Body)
drawSphere()
glPushMatrix()
glMultMatrix(T_EyeLeft/Head)
drawCylinder()
glPopMatrix()
glPushMatrix()
glMultMatrix(T_EyeRight/Head)
drawCylinder()
glPopMatrix()
glPopMatrix()
glPopMatrix()
```



Remarks

- `glMultMatrix()` usually is `glTranslate()`, `glScale()` or/and `glRotate()`
- $Coordinates_{world} = T_{Body}\, T_{Head/Body}\, T_{EyeRight/Head}\, Coordinates_{object}$
- Example :

`drawSquare()`

$$\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}$$

`glScalef(2.0,1.0)` `drawSquare()`

$$\begin{pmatrix}2\\1\\1\end{pmatrix} = \begin{pmatrix}2&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}$$

Plan

Conclusion :

- Done :
  - Transformations in OpenGL
  - Hierarchical modeling = composition of transformations
- Highlights :
  - `GL_MODELVIEW`, `GL_PROJECTION`
  - Graph scene
- To do :
  - Lab session : model a robot
  - Lights, materials, effects, buffers...next week!