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

1. Transformations
2. OpenGL transformations
   OpenGL matrices
   Projection
   Modelview
3. Hierarchical modeling
   Matrix stack
   Graph scene
4. Conclusion

Graphics Pipeline

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

Viewport

Determine how large you want the final image to be :
\[ \text{glViewport}(x, y, width, height) \]
Hierarchical modeling
Transformations
OpenGL transformations
Open GL
Transform
Projection
Modelview
Hierarchical modeling
Matrix stack
Graph scene
Conclusion

Representation of transformations

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

Plan

1. Transformations
2. OpenGL transformations
   - OpenGL matrices
   - Projection
   - Modelview
3. Hierarchical modeling
   - Matrix stack
   - Graph scene
4. Conclusion

Matrices in OpenGL

- Predefined matrices:
  - GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE, ...
  → Current matrix defined with glMatrixMode(...) ⇒ \( M_0 \)
- Load:
  - glLoadIdentity(), glLoadMatrixf(M)
  - glLoadTransposeMatrixd(N)
    ⇒ \( M_0 = I, M_0 = M, M_0 = N^T \)
- Arithmetic:
  - glMultMatrixd(P), glMultTransposeMatrixf(Q)
    ⇒ \( M_0 = M_0P, M_0 = M_0Q^T \)
- Transformations:
  - glTranslatef(1.0, 2.0, -1.5),
  - glRotated(90.0, 0.0, 1.0, 0.0),
  - glScalef(2.0, -0.5, 1.0)

PROJECTION matrix

- Mode: GL_PROJECTION
- Perspective:
  - glFrustum(left, right, bottom, top, near, far)
- Orthographic:
  - glOrtho(left, right, bottom, top, near, far)
1. Transformations

2. OpenGL transformations
   - OpenGL matrices
   - Projection
   - Modelview

3. Hierarchical modeling
   - Matrix stack
   - Graph scene

4. Conclusion

---

**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

---

**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

---

**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()`
1 Plan

1 Transformations

2 OpenGL transformations
OpenGL matrices
Projection
Modelview

3 Hierarchical modeling
Matrix stack
Graph scene

4 Conclusion

Hierarchical modeling
Transformations
OpenGL transformations
OpenGL matrices
Projection
Modelview

Graph scene - Example 1/2

Remarks

• `glMultMatrix()` usually is `glTranslate()`, `glScale()` or `glRotate()`

• Coordinates_{world} = T_{Body} T_{Head / Body} T_{EyeRight / Head} Coordinates_{object}

• Example:

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

```
    \begin{pmatrix}
        1 & 0 & 0 & 1 \\
        0 & 1 & 0 & 1 \\
        0 & 0 & 1 & 1 \\
    \end{pmatrix} \times \begin{pmatrix}
        1 & 0 & 0 & 1 \\
        2 & 0 & 0 & 1 \\
        0 & 1 & 0 & 1 \\
        0 & 0 & 1 & 1 \\
    \end{pmatrix} = \begin{pmatrix}
        1 & 0 & 0 & 1 \\
        2 & 0 & 0 & 1 \\
        0 & 1 & 0 & 1 \\
        0 & 0 & 1 & 1 \\
    \end{pmatrix}
```

Plan

1 Transformations

2 OpenGL transformations
OpenGL matrices
Projection
Modelview

3 Hierarchical modeling
Matrix stack
Graph scene

4 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!