Hierarchical modelling

- recap: composing transforms
- finish: rendering pipeline
- matrix hierarchies and stacks
- display lists

Composing transforms recap

Order matters
- 4x4 matrix multiplication not commutative!

Moving to origin
- Transformation of geometry into coordinate system where operation becomes simpler
- Perform operation
- Transform geometry back to original coordinate system

The Rendering Pipeline

Geometry | Model/View Transform | Lighting | Perspective Transform | Clipping
---|---|---|---|---

Scan Conversion | Texturing | Depth Test | Blending | Framebuffer

Discussion

Advantages of a pipeline structure
- Logical separation of the different components, modularity
- Easy to parallelize:
  - Earlier stages can already work on new data while later stages still work with previous data
  - Similar to pipelining in modern CPUs
  - But much more aggressive parallelization possible (special purpose hardware!)
  - Important for hardware implementations!
- Only local knowledge of the scene is necessary

Discussion

Disadvantages:
- Limited flexibility
- Some algorithms would require different ordering of pipeline stages
  - Hard to achieve while still preserving compatibility
- Only local knowledge of scene is available
  - Shadows
  - Global illumination
Matrix Operations in OpenGL

Direct matrix specification:
- load
  - glLoadIdentity
  - glLoadMatrix
- multiply
  - glMultMatrix

Transformations:
- glRotate*  
- glTranslate*  
- glScale*

Matrix Operations in OpenGL

gRotate* (θ, x, y, z)
θ : ccw rotation angle, looking along vector towards origin
x, y, z : vector along which rotation occurs

equivalent to glMultMatrix(A),  A =
\[
\begin{bmatrix}
x'(1 - \cos \theta) + \cos \theta & x'(1 - \cos \theta) - z \sin \theta & y'(1 - \cos \theta) + y \sin \theta & 0 \\
y'(1 - \cos \theta) + z \sin \theta & y'(1 - \cos \theta) + \cos \theta & y'(1 - \cos \theta) - x \sin \theta & 0 \\
z'(1 - \cos \theta) - y \sin \theta & z'(1 - \cos \theta) + x \sin \theta & z'(1 - \cos \theta) + \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Matrix Operations in OpenGL

2 Matrices:
- Model/view matrix M
- Projective matrix P

Example:
gMatrixMode( GL_MODELVIEW );
gLoadIdentity(); // M=I
gLRotatef( angle, x, y, z ); // M=M*R(θ)
gTranslatef( x, y, z ); // M= M*TR(α)
gMatrixMode( GL_PROJECTION );
gMatrixMode( GL_PROJECTION );

t r a n s f o r m a t i o n  H i e r a r c h i e s

Transformation Hierarchies

Scene graph
Transformation Hierarchies

Matrix Stack

```
glLoadIdentity();
glTranslatef(4, 1, 0);
glRotatef(45, 0, 0, 1);
glTranslatef(0, 2, 0);
glScalef(2, 1, 1);
glTranslatef(1, 0, 0);
glPopMatrix();
```

Composing Transformations

OpenGL example

Hierarchical Modeling: Matrix Stacks

Advantages:
- Matrix stacks make it feasible to have multiple copies of one object
- No need to compute inverse matrices all the time
- Avoids incremental changes to coordinate systems
  - Accumulation of numerical errors

Practical issues:
- In graphics hardware, depth of matrix stacks is limited
  - (typically 16 for model/view and about 4 for projective matrix)
Example: modularization

**Drawing a scaled square**

```c
void drawBlock(float k) {
    glPushMatrix();
    glScalef(k,k,k);
    glBegin(GL_LINE_LOOP);
    glVertex3f(0,0,0);
    glVertex3f(1,0,0);
    glVertex3f(1,1,0);
    glVertex3f(0,1,0);
    glEnd();
    glPopMatrix();
}
```

Example: applets

http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/scenegraphs.html

Display Lists

**Concept:**
- If multiple copies of an object are required, it can be compiled into a display list:
  ```c
  glNewList( listId, GL_COMPILE );
  glBegin( ...);
  ... // geometry goes here
  glEndList();
  // render two copies of geometry offset by 1 in z-direction:
  glCallList( listId );
  glTranslatef( 0.0, 0.0, 1.0 );
  glCallList( listId );
  ```

Display Lists

**Advantages:**
- More efficient than individual function calls for every vertex/attribute
- Can be cached on the graphics board (bandwidth!)
- Display lists exist across multiple frames
  - **Represent static objects in an interactive application**

Example: 36 Snowmen

http://www.lighthouse3d.com/opengl/displaylists/
efficiency issues
Snowmen: no lists

```c
// Draw 36 Snowmen
for(int i = -3; i < 3; i++)
  for(int j=-3; j < 3; j++) {
    glPushMatrix();
    gTranslate(i*10.0,0,j * 10.0);
    // Call the function to draw a snowman
drawSnowMan();
    glPopMatrix();
  }
```

Snowmen: no lists

```c
void drawSnowMan() {
  // Draw Eyes
  glPushMatrix();
  glColor3f(1.0f, 1.0f, 1.0f);
  glTranslatef(0.5f, -0.5f, 0.0f);
  glutSolidSphere(0.05f, 10, 10);
  glPopMatrix();
  // Draw Body
  glTranslatef(-0.1f, 0.0f, 0.0f);
  glutSolidSphere(0.75f, 20, 20);
  glPopMatrix();
  // Draw Head
  glTranslatef(0.0f, 1.0f, 0.0f);
  glutSolidSphere(0.25f, 20, 20);
  glPopMatrix();
}
```

Snowmen: no lists

```c
// Draw 36 Snowmen
for(int i = -3; i < 3; i++)
  for(int j=-3; j < 3; j++) {
    glPushMatrix();
    gTranslate(i*10.0,0,j * 10.0);
    // Call the function to draw a snowman
drawSnowMan();
    glPopMatrix();
  }
```

Display Lists

Example: 36 Snowmen
http://www.lighthouse3d.com/opengl/displaylists/

benchmarks of 36K polygons

- 55 FPS: no display list

Snowmen: display lists

```c
GLuint createDL() {
  glutInitDisplayLists(1);
  // start list
  glGenLists(1);
  // end list
  glEndList();
  return(1);
}
```

Snowmen: no lists

```c
// Draw 36 Snowmen
for(int i = -3; i < 3; i++)
  for(int j=-3; j < 3; j++) {
    glPushMatrix();
    gTranslate(i*10.0,0,j * 10.0);
    // Call the function to draw a snowman
    glCallList(DLid);
    glPopMatrix();
  }
```
Display Lists

Example: 36 Snowmen
http://www.lighthouse3d.com/opengl/displaylists/

benchmarks of 36K polygons

- 55 FPS: no display list
- 153 FPS: 1 snowman display list, called 36 times

Snowmen: one big list

GLuint createDL() {
    GLuint snowManDL;
    snowManDL = glGenLists(1);
    glNewList(snowManDL,GL_COMPILE);
    for(int i=-3; i < 3; i++) {
        for(int j=-3; j < 3; j++) {
            glBeginList(snowManDL);
            glTranslate(i*10.0,j*10.0);
            drawSnowMan();
            glEndList();
        }
    }
    return(snowManDL);
}

Display Lists

Example: 36 Snowmen
http://www.lighthouse3d.com/opengl/displaylists/

benchmarks of 36K polygons

- 55 FPS: no display list
- 153 FPS: 1 snowman display list, called 36 times
- 108 FPS: single 36 snowman display list

Snowmen: nested lists

GLuint createDL() {
    GLuint snowManDL,loopDL;
    snowManDL = glGenLists(1);
    loopDL = glGenLists(1);
    glNewList(snowManDL,GL_COMPILE);
    drawSnowMan();
    glEndList();
    glNewList(loopDL,GL_COMPILE);
    for(int i=-3; i < 3; i++) {
        for(int j=-3; j < 3; j++) {
            glBeginList(snowManDL);
            glTranslate(i*10.0,j*10.0);
            drawSnowMan();
            glEndList();
        }
    }
    return(loopDL);
}

Display Lists

Example: 36 Snowmen
http://www.lighthouse3d.com/opengl/displaylists/

benchmarks of 36K polygons

- 55 FPS: no display list
- 153 FPS: 1 snowman display list, called 36 times
- 108 FPS: single 36 snowman display list
- 153 FPS: nested display lists