

Assignment 3

Introduction to OpenGL (2)

Ming Chuang

T.A CS – 357 / Computer Graphics

04/07/2009

Outline

- Display List
- Shadow
- Stencil Buffer
- Luxo Jr. Lamp
- Miscellaneous

Display List

- To accelerate the rendering, OpenGL allows you to “record” a set of commands into a display list.

Display List

- To accelerate the rendering, OpenGL allows you to “record” a set of commands into a display list.
- All commands in the lists will be precompiled and, more importantly, will be stored in the graphics card memory if possible.

Example code (conti.)

```
#define USE_DISPLAY_LIST 1
#define res 4
#define radius 0.3
#define len (radius*2)/res

void redraw2()
{
    static int listID;
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glRotatef( 0.3 , 1 , 3 , 0);

    #if USE_DISPLAY_LIST
        if( listID!=0 ) glCallList(listID);
        else
        {
            listID = glGenLists(1);
            glNewList( listID , GL_COMPILE_AND_EXECUTE );
    #endif
            glLineWidth(1);
            glColor3f(1.0f,0.0f,0.0f);
            for(int x=0;x<res;x++)
                for(int y=0;y<res;y++)
                    for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    #if USE_DISPLAY_LIST
        glEndList();
    }
    #endif

    updateFrameRate();
    glutSwapBuffers();
}
```

Example code (conti.)

```
#define USE_DISPLAY_LIST 1
#define res 4
#define radius 0.3
#define len (radius*2)/res

void redraw2()
{
    static int listID;
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glRotatef( 0.3 , 1 , 3 , 0);

    #if USE_DISPLAY_LIST
    if( listID!=0 ) glCallList(listID);
    else
    {
        listID = glGenLists(1);
        glNewList( listID , GL_COMPILE_AND_EXECUTE );
    #endif

        glLineWidth(1);
        glColor3f(1.0f,0.0f,0.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
    #if USE_DISPLAY_LIST
        glEndList();
    }
    #endif

    updateFrameRate();
    glutSwapBuffers();
}
```

Initialize an empty list and retrieve its handle (list ID).

listID = glGenLists(1);

Example code (conti.)

```
#define USE_DISPLAY_LIST 1
#define res 4
#define radius 0.3
#define len (radius*2)/res

void redraw2()
{
    static int listID;
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glRotatef( 0.3 , 1 , 3 , 0);

    #if USE_DISPLAY_LIST
    if( listID!=0 ) glCallList(listID);
    else
    {
        listID = glGenLists(1);
        glNewList( listID , GL_COMPILE_AND_EXECUTE );
    #endif
        glLineWidth(1);
        glColor3f(1.0f,0.0f,0.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
    #if USE_DISPLAY_LIST
        glEndList();
    #endif

    updateFrameRate();
    glutSwapBuffers();
}
```

Record the commands in-between.
(drawing things as usual)

listID = glGenLists(1);
glNewList(listID , GL_COMPILE_AND_EXECUTE);

glEndList();

Example code (conti.)

```
#define USE_DISPLAY_LIST 1
#define res 4
#define radius 0.3
#define len (radius*2)/res

void redraw2()
{
    static int listID;
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glRotatef( 0.3 , 1 , 3 , 0);

    #if USE_DISPLAY_LIST
    if( listID!=0 ) glCallList(listID);
    else
    {
        listID = glGenLists(1);
        glNewList( listID , GL_COMPILE_AND_EXECUTE );
    #endif

        glLineWidth(1);
        glColor3f(1.0f,0.0f,0.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
    #if USE_DISPLAY_LIST
        glEndList();
    }
    #endif

    updateFrameRate();
    glutSwapBuffers();
}
```

If the list has already been created, simply invoke it by the call list ID.

```

void drawGridUnit(GLfloat x, GLfloat y, GLfloat z, GLfloat len)
{
    glBegin(GL_LINES);
        glVertex3f(x,    y,    z    ); glVertex3f(x,    y+len, z    );
        glVertex3f(x,    y+len, z    ); glVertex3f(x+len, y+len, z    );
        glVertex3f(x+len, y+len, z    ); glVertex3f(x+len, y    , z    );
        glVertex3f(x+len, y    , z    ); glVertex3f(x    , y    , z    );

        glVertex3f(x,    y,    z+len); glVertex3f(x,    y+len, z+len);
        glVertex3f(x,    y+len, z+len); glVertex3f(x+len, y+len, z+len);
        glVertex3f(x+len, y+len, z+len); glVertex3f(x+len, y    , z+len);
        glVertex3f(x+len, y    , z+len); glVertex3f(x    , y    , z+len);

        glVertex3f(x,    y,    z+len); glVertex3f(x,    y,    z    );
        glVertex3f(x,    y+len, z+len); glVertex3f(x,    y+len, z    );
        glVertex3f(x+len, y+len, z+len); glVertex3f(x+len, y+len, z    );
        glVertex3f(x+len, y    , z+len); glVertex3f(x+len, y    , z    );
    glEnd();
}

```

```

void updateFrameRate()
{
    char temp[16];
    double t;
    static double frameRate, frameCountStart=GetTime();
    static int frameCount;
    frameCount++;
    if(frameCount==100){
        frameCount=0;
        t=frameCountStart;
        frameCountStart=GetTime();
        frameRate=100/(frameCountStart-t);
    }
    sprintf(temp, "%.1f fs", frameRate);
    if(frameRate!=0) glutSetWindowTitle(temp);
}

```

```

void drawGridUnit(GLfloat x, GLfloat y, GLfloat z, GLfloat len)
{
    glBegin(GL_LINES);
        glVertex3f(x,    y,    z    ); glVertex3f(x,    y+len, z    );
        glVertex3f(x,    y+len, z    ); glVertex3f(x+len, y+len, z    );
        glVertex3f(x+len, y+len, z    ); glVertex3f(x+len, y    , z    );
        glVertex3f(x+len, y    , z    ); glVertex3f(x    , y    , z    );

        glVertex3f(x,    y,    z+len); glVertex3f(x,    y+len, z+len);
        glVertex3f(x,    y+len, z+len); glVertex3f(x+len, y+len, z+len);
        glVertex3f(x+len, y+len, z+len); glVertex3f(x+len, y    , z+len);
        glVertex3f(x+len, y    , z+len); glVertex3f(x    , y    , z+len);

        glVertex3f(x,    y,    z+len); glVertex3f(x,    y,    z    );
        glVertex3f(x,    y+len, z+len); glVertex3f(x,    y+len, z    );
        glVertex3f(x+len, y+len, z+len); glVertex3f(x+len, y+len, z    );
        glVertex3f(x+len, y    , z+len); glVertex3f(x+len, y    , z    );
    glEnd();
}

```

```

void updateFrameRate()
{
    char temp[16];
    double t;
    static double frameRate, frameCountStart=GetTime();
    static int frameCount;
    frameCount++;
    if(frameCount==100){
        frameCount=0;
        t=frameCountStart;
        frameCountStart=GetTime();
        frameRate=100/(frameCountStart-t);
    }
    sprintf(temp, "%.1f fs", frameRate);
    if(frameRate!=0) glutSetWindowTitle(temp);
}

```

~ DEMO ~

Shadow (2 approaches)

- Direct Rendering
- Shadow Volume

Shadow (2 approaches)

□ Direct Rendering:

- Idea: splatting/flattening the 3D objects onto the ground.
 - Given the plane equation and the light position/direction, we can compute the transformation matrix that projects all objects down to the plane (see the redbook for details)
 - Apply the matrix and then draw objects as usual (using black color).

Shadow (2 approaches)

❑ Direct Rendering:

- Idea: splatting/flattening the 3D objects onto the ground.
 - Given the plane equation and the light position/direction, we can compute the transformation matrix that projects all objects down to the plane (see the redbook for details)
 - Apply the matrix and then draw objects as usual (using black color).
- Difficulties and drawbacks:
 - Only work for planes.
 - Need to know where the plane is to compute its equation.

Example Code (1/2)

```
void redraw3()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    static double degree;
    degree+=0.1;

    glPushMatrix();
    {
        glRotatef(degree, 0, 1, 0);
        glLineWidth(2);
        glColor3f(1.0f,0.0f,1.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    }
    glPopMatrix();
}
```

Example Code (1/2)

```
void redraw3()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    static double degree;
    degree+=0.1;

    glPushMatrix();
    {
        glRotatef(degree, 0, 1, 0);
        glLineWidth(2);
        glColor3f(1.0f,0.0f,1.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    }
    glPopMatrix();
}
```

Rendering the grid as the previous example



```
        glRotatef(degree, 0, 1, 0);
        glLineWidth(2);
        glColor3f(1.0f,0.0f,1.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    }
```

Example Code (2/2)

```
GLfloat shadowMatrix[16];
getShadowMatrix(shadowMatrix);

glPushMatrix();
{
    glMultMatrixf(shadowMatrix);
    glRotatef(degree, 0, 1, 0);
    glLineWidth(2);
    glColor3f(0.0f,0.0f,0.0f);
    for(int x=0;x<res;x++)
        for(int y=0;y<res;y++)
            for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
}
glPopMatrix();

glutSwapBuffers();
}
```

Example Code (2/2)

```
GLfloat shadowMatrix[16];  
getShadowMatrix(shadowMatrix);
```

```
glPushMatrix();  
{
```

```
    glMultMatrixf(shadowMatrix);
```

```
    glRotatef(degree, 0, 1, 0);
```

```
    glLineWidth(2);
```

```
    glColor3f(0.0f,0.0f,0.0f);
```

```
    for(int x=0;x<res;x++)
```

```
        for(int y=0;y<res;y++)
```

```
            for(int z=0;z<res;z++)
```

```
                {
```

```
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
```

```
                }
```

```
    }
```

```
    glPopMatrix();
```

```
    glutSwapBuffers();
```

```
}
```

Compute the matrix that projects all coordinates onto the plane (your job) and apply it to the current matrix

Example Code (2/2)

```
GLfloat shadowMatrix[16];
getShadowMatrix(shadowMatrix);

glPushMatrix();
{
    glMultMatrixf(shadowMatrix);
    glRotatef(degree, 0, 1, 0);
    glLineWidth(2);
    glColor3f(0.0f,0.0f,0.0f);
    for(int x=0;x<res;x++)
        for(int y=0;y<res;y++)
            for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
}
glPopMatrix();

glutSwapBuffers();
}
```

Rendering again

Example Code (2/2)

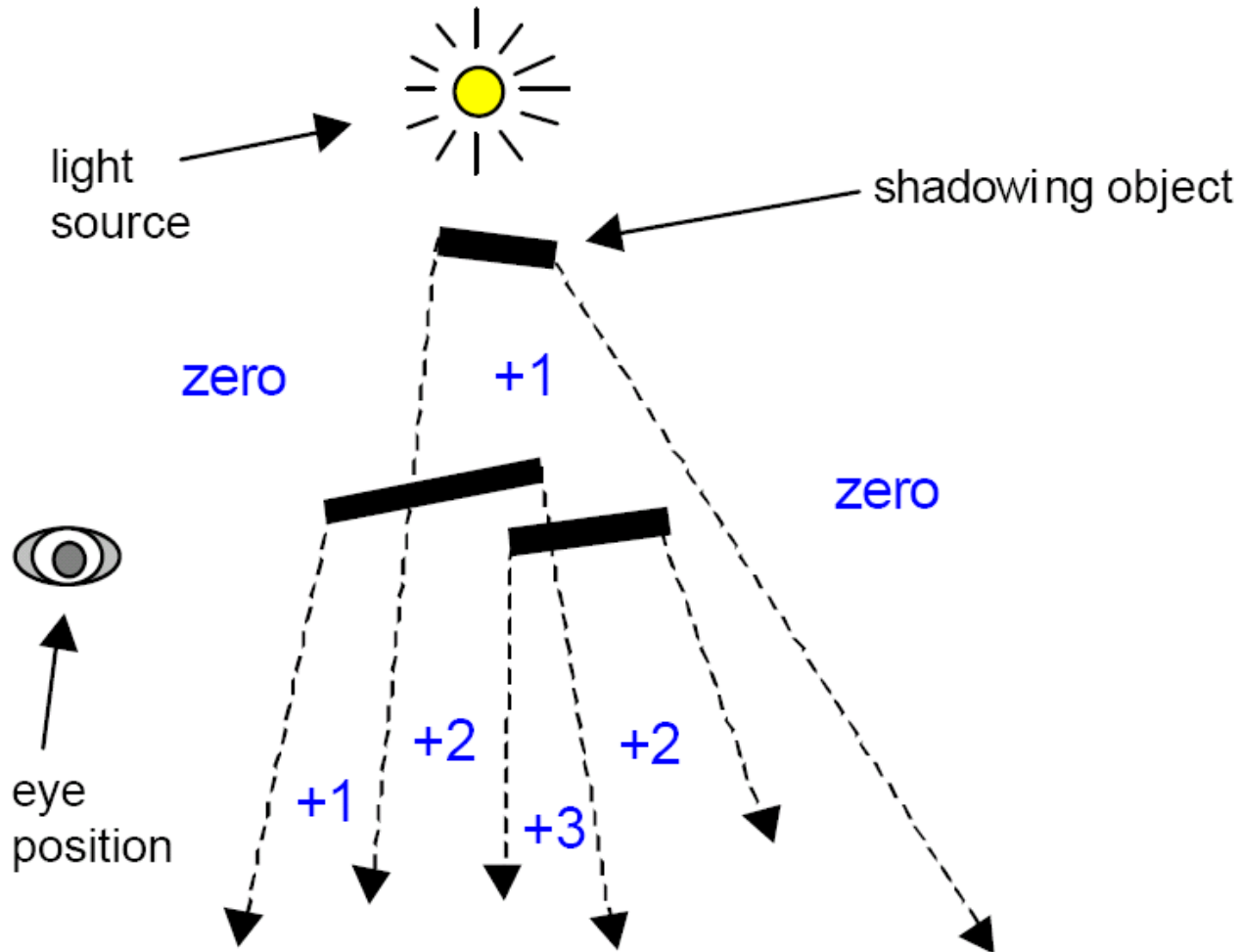
```
GLfloat shadowMatrix[16];
getShadowMatrix(shadowMatrix);

glPushMatrix();
{
    glMultMatrixf(shadowMatrix);
    glRotatef(degree, 0, 1, 0);
    glLineWidth(2);
    glColor3f(0.0f,0.0f,0.0f);
    for(int x=0;x<res;x++)
        for(int y=0;y<res;y++)
            for(int z=0;z<res;z++)
                {
                    drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                }
}
glPopMatrix();

glutSwapBuffers();
}
```

~ DEMO ~

□ Shadow Volume (needs the stencil buffer):



Stencil Buffer (mirror as example)

- Intuition: to tell OpenGL “where to draw” (stenciling)
- To draw a mirror, a 2-pass rendering is required.
- How to do it?

Stencil Buffer (mirror as example)

- Intuition: to tell OpenGL “where to draw” (stenciling)
- To draw a mirror, a 2-pass rendering is required.
- How to do it? (general steps)
 - 1. draw the Stencil Buffer to define where is the mirror
 - 2. draw the world “inside” the mirror (flipping the space first)
 - 3. draw the world outside the mirror as usual

Example Code (1/3)

```
void redraw4()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( 0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    // step 0. draw the contour of the mirror)
    float Y_min=0.1, Y_max=0.3, Z_min=-0.5, Z_max=0.5;
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,0);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();

    // step 1. mark the mirror area.....|
    glColorMask(false,false,false,false);
    glDepthMask(false);
    glClearStencil(0);
    glClear( GL_STENCIL_BUFFER_BIT );
    glEnable(GL_STENCIL_TEST);
    glStencilFunc(GL_ALWAYS, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    glBegin(GL_QUADS);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();
    //.....
```

Example Code (1/3)

```
void redraw4()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( 0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    // step 0. draw the contour of the mirror)
    float Y_min=0.1, Y_max=0.3, Z_min=-0.5, Z_max=0.5;
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,0);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();

    // step 1. mark the mirror area .....|
    glColorMask(false,false,false,false);
    glDepthMask(false);
    glClearStencil(0);
    glClear( GL_STENCIL_BUFFER_BIT );
    glEnable(GL_STENCIL_TEST);
    glStencilFunc(GL_ALWAYS, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    glBegin(GL_QUADS);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();
    // .....
```

Turn off the depth test and disable drawing into the color buffer, since we just want to mark the mirror area in the stencil buffer.

Example Code (1/3)

```
void redraw4()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( 0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    // step 0. draw the contour of the mirror)
    float Y_min=0.1, Y_max=0.3, Z_min=-0.5, Z_max=0.5;
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,0);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();

    // step 1. mark the mirror area. ....
    glColorMask(false,false,false,false);
    glDepthMask(false);
    glClearStencil(0);
    glClear( GL_STENCIL_BUFFER_BIT );
    glEnable(GL_STENCIL_TEST);
    glStencilFunc(GL_ALWAYS, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    glBegin(GL_QUADS);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();
    // .....
```

Clear the buffer.

Enable the buffer

Let the test always pass.

Mark the buffer when the test passes

Example Code (1/3)

```
void redraw4()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( 0.1, 0.1, 0.1,
              0.0, 0.0, 0.0,
              -1.0, 2.0, -1.0 );

    // step 0. draw the contour of the mirror)
    float Y_min=0.1, Y_max=0.3, Z_min=-0.5, Z_max=0.5;
    glBegin(GL_LINE_LOOP);
        glColor3f(0,0,0);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();

    // step 1. mark the mirror area. ....|
    glColorMask(false,false,false,false);
    glDepthMask(false);
    glClearStencil(0);
    glClear( GL_STENCIL_BUFFER_BIT );
    glEnable(GL_STENCIL_TEST);
    glStencilFunc(GL_ALWAYS, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    glBegin(GL_QUADS);
        glVertex3f(0, Y_max, Z_max);
        glVertex3f(0, Y_max, Z_min);
        glVertex3f(0, Y_min, Z_min);
        glVertex3f(0, Y_min, Z_max);
    glEnd();
    // .....
```

In this example, we draw a rectangle mirror on the Y-Z plane

Example Code (2/3)

```
// step 2. draw the reflected world.....
glDepthMask(true);
glColorMask(true,true,true,true);
glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glPushMatrix();
{
    glScalef(-1.0f, 1.0f, 1.0f);
    redraw4_sub();
}
glPopMatrix();
glDisable(GL_STENCIL_TEST);
// .....

// step 3. draw the original world.....
redraw4_sub();
// .....

glEnable(GL_LINE_STIPPLE);
glLineStipple(3, 0xAAAA);
glLineWidth(0.1);
glColor3f(0.5,0.5,0.5);
glBegin(GL_LINES);
    glVertex3f(-1, 0, 0); glVertex3f( 1, 0, 0);
    glVertex3f( 0,-1, 0); glVertex3f( 0, 1, 0);
    glVertex3f( 0, 0,-1); glVertex3f( 0, 0, 1);
glEnd();
glDisable(GL_LINE_STIPPLE);

glutSwapBuffers();
}
```

Example Code (2/3)

```
// step 2. draw the reflected world.....
glDepthMask(true);
glColorMask(true,true,true,true);
glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glPushMatrix();
{
    glScalef(-1.0f, 1.0f, 1.0f);
    redraw4_sub();
}
glPopMatrix();
glDisable(GL_STENCIL_TEST);
//.....

// step 3. draw the original world.....
redraw4_sub();
//.....
```

```
glEnable(GL_LINE_STIPPLE);
glLineStipple(3, 0xAAAA);
glLineWidth(0.1);
glColor3f(0.5,0.5,0.5);
glBegin(GL_LINES);
    glVertex3f(-1, 0, 0); glVertex3f( 1, 0, 0);
    glVertex3f( 0,-1, 0); glVertex3f( 0, 1, 0);
    glVertex3f( 0, 0,-1); glVertex3f( 0, 0, 1);
glEnd();
glDisable(GL_LINE_STIPPLE);

glutSwapBuffers();
}
```

Re-enable the depth test and drawing

Example Code (2/3)

```
// step 2. draw the reflected world.....
glDepthMask(true);
glColorMask(true,true,true,true);
glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glPushMatrix();
{
    glScalef(-1.0f, 1.0f, 1.0f);
    redraw4_sub();
}
glPopMatrix();
glDisable(GL_STENCIL_TEST);
//.....

// step 3. draw the original world.....
redraw4_sub();
//.....
```

```
glEnable(GL_LINE_STIPPLE);
glLineStipple(3, 0xAAAA);
glLineWidth(0.1);
glColor3f(0.5,0.5,0.5);
glBegin(GL_LINES);
    glVertex3f(-1, 0, 0); glVertex3f( 1, 0, 0);
    glVertex3f( 0,-1, 0); glVertex3f( 0, 1, 0);
    glVertex3f( 0, 0,-1); glVertex3f( 0, 0, 1);
glEnd();
glDisable(GL_LINE_STIPPLE);

glutSwapBuffers();
}
```

Only draw things when the tests pass.
Read-only mode .

Example Code (2/3)

```
// step 2. draw the reflected world.....
glDepthMask(true);
glColorMask(true,true,true,true);
glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glPushMatrix();
{
    glScalef(-1.0f, 1.0f, 1.0f);
    redraw4_sub();
}
glPopMatrix();
glDisable(GL_STENCIL_TEST);
//.....

// step 3. draw the original world.....
redraw4_sub();
//.....
```

```
glEnable(GL_LINE_STIPPLE);
glLineStipple(3, 0xAAAA);
glLineWidth(0.1);
glColor3f(0.5,0.5,0.5);
glBegin(GL_LINES);
    glVertex3f(-1, 0, 0); glVertex3f( 1, 0, 0);
    glVertex3f( 0,-1, 0); glVertex3f( 0, 1, 0);
    glVertex3f( 0, 0,-1); glVertex3f( 0, 0, 1);
glEnd();
glDisable(GL_LINE_STIPPLE);

glutSwapBuffers();
}
```

Reflect the space around the Y-Z plane.
Then draw the things as usual

Example Code (2/3)

```
// step 2. draw the reflected world.....
glDepthMask(true);
glColorMask(true,true,true,true);
glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glPushMatrix();
{
    glScalef(-1.0f, 1.0f, 1.0f);
    redraw4_sub();
}
glPopMatrix();
glDisable(GL_STENCIL_TEST);
//.....
```

After disabling the stencil test and going back to the original space, draw the things again (second pass).

```
// step 3. draw the original world.....
redraw4_sub();
//.....
```

```
glEnable(GL_LINE_STIPPLE);
glLineStipple(3, 0xAAAA);
glLineWidth(0.1);
glColor3f(0.5,0.5,0.5);
glBegin(GL_LINES);
    glVertex3f(-1, 0, 0); glVertex3f( 1, 0, 0);
    glVertex3f( 0,-1, 0); glVertex3f( 0, 1, 0);
    glVertex3f( 0, 0,-1); glVertex3f( 0, 0, 1);
glEnd();
glDisable(GL_LINE_STIPPLE);

glutSwapBuffers();
}
```

Example Code (3/3)

```
//drawing the grid as previous examples
void redraw4_sub()
{
    static double degree;
    degree+=0.1;
    glPushMatrix();
    {
        glTranslatef(0.5,0.0,0);
        glRotatef(degree, 0, 1, 0);
        glLineWidth(2);
        glColor3f(1.0f,0.0f,1.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    }
    glPopMatrix();
}
```

Example Code (3/3)

```
//drawing the grid as previous examples
void redraw4_sub()
{
    static double degree;
    degree+=0.1;
    glPushMatrix();
    {
        glTranslatef(0.5,0.0,0);
        glRotatef(degree, 0, 1, 0);
        glLineWidth(2);
        glColor3f(1.0f,0.0f,1.0f);
        for(int x=0;x<res;x++)
            for(int y=0;y<res;y++)
                for(int z=0;z<res;z++)
                    {
                        drawGridUnit(x*len-radius, y*len-radius, z*len-radius, len);
                    }
    }
    glPopMatrix();
}
```

~ DEMO ~

Luxo Jr. Lamp

- Revisit the matrix manipulation:

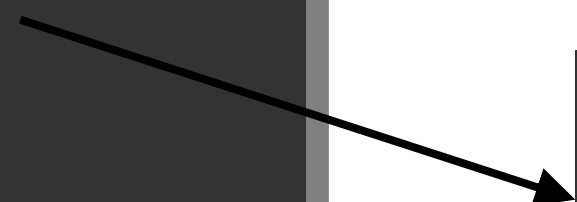
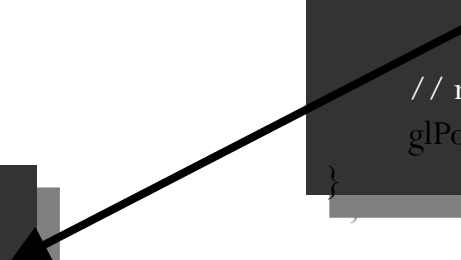
```
// save the current state
glPushMatrix();
    //do whatever you want to transform...
    glMultMatrixf(.....);
    glScalef(.....);
    .....
// recover the state
glPopMatrix();
```

- Essentially it is a robot arm problem.
- Same as the concept of the scene graph. We propagate the transformation matrix at each joint.

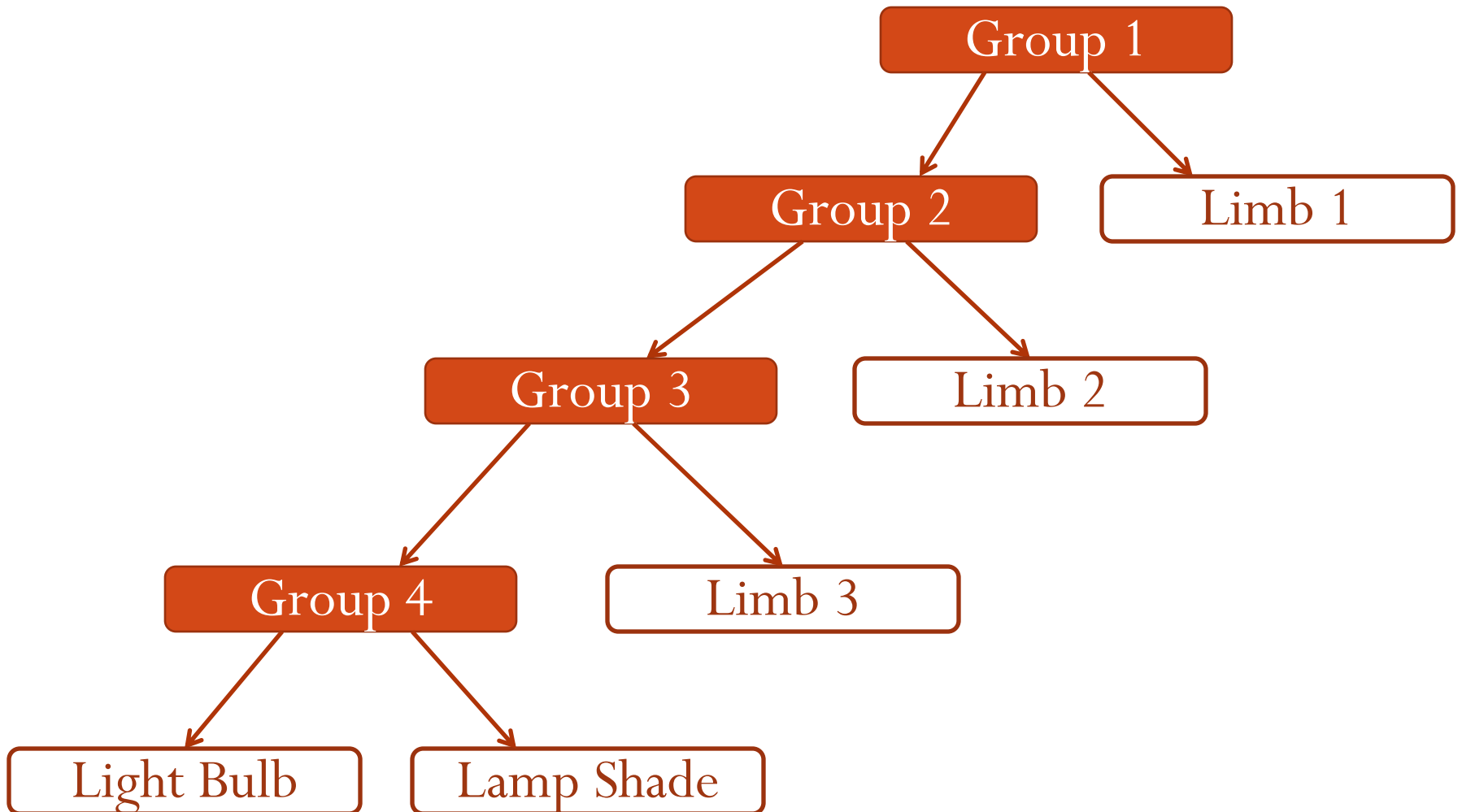
```
RayGroup::drawOpenGL(...) {  
  
    // save the current state  
    glPushMatrix();  
    // apply the transformation  
    glMultMatrixf(matrix);  
    // draw each children  
    [redacted]  
    // recover the state  
    glPopMatrix();  
}
```

```
RayGroup::drawOpenGL(...) {  
  
    // save the current state  
    glPushMatrix();  
    // apply the transformation  
    glMultMatrixf(matrix);  
    // draw each children  
    [redacted]  
    // recover the state  
    glPopMatrix();  
}
```

```
RayGroup::drawOpenGL(...) {  
  
    // save the current state  
    glPushMatrix();  
    // apply the transformation  
    glMultMatrixf(matrix);  
    // draw each children  
    [redacted]  
    // recover the state  
    glPopMatrix();  
}
```



- Your graph will look like an unbalanced tree:



- The idea is easy. The hard part is how to do it under the code skeleton
- You can develop your own solution.
- My solution was...

- The idea is easy. The hard part is how to do it under the code skeleton
- You can develop your own solution.
- My solution was.... (for reference only, not necessarily a good choice)
 - Set up flags to let us know when and which transformation matrix to apply (or you can say, at which level of the tree we are now) Each transformation matrix corresponds to a “joint”.
 - Need to modify the header files to accommodate the implementation.

Miscellaneous

- Moving the camera?
 - Play with Misha's executable to make sure you understand what you are supposed to implement. (use the middle button to rotate the scene)

Miscellaneous

- Moving the camera?
 - Play with Misha's executable to make sure you understand what you are supposed to implement. (use the middle button to rotate the scene)
 - Your task is to update the camera parameters (position, direction, and up), so the codebase can set up the camera properly by `RayCamera::drawOpenGL` before drawing each frame.

Miscellaneous

- Moving the camera?
 - Play with Misha's executable to make sure you understand what you are supposed to implement. (use the middle button to rotate the scene)
 - Your task is to update the camera parameters (position, direction, and up), so the codebase can set up the camera properly by `RayCamera::drawOpenGL` before drawing each frame.
 - For `RayCamera::rotateUP()/rotateRight()`, three different transformations are required:
 - 1. translating the space so the origin sits at 'center'
 - 2. rotating around the up/right vector
 - 3. translating back (opposite operation to 1.)

- Super-sampling?
 - See the red book as to how to use the accumulation buffer
 - Break into rayScene.cpp to modify RayScene::drawOpenGL()

- Super-sampling?
 - See the red book as to how to use the accumulation buffer
 - Break into rayScene.cpp to modify RayScene::drawOpenGL()
- Cone and Cylinder
 - Pay attention to whether it sits at the correct position with the correction orientation.

- Super-sampling?
 - See the red book as to how to use the accumulation buffer
 - Break into rayScene.cpp to modify RayScene::drawOpenGL()
- Cone and Cylinder
 - Pay attention to whether it sits at the correct position with the correction orientation.
- Box
 - Make sure each quad's front is facing the correct direction
 - Make sure the normals are assigned properly (by observing the specularities).

- Super-sampling?
 - See the red book as to how to use the accumulation buffer
 - Break into rayScene.cpp to modify RayScene::drawOpenGL()
- Cone and Cylinder
 - Pay attention to whether it sits at the correct position with the correction orientation.
- Box
 - Make sure each quad's front is facing the correct direction
 - Make sure the normals are assigned properly (by observing the specularities).
- Others?
 - Send me a mail if you are interested in doing more.

- Apart from sharing or copying codes, you are allowed to explore and exploit all resources you can find.

- Apart from sharing or copying codes, you are allowed to explore and exploit all resources you can find.
- Also, you are encouraged to add commandline arguments, and/or develop a more complex interface, to deal with the new functionalities.

- Apart from sharing or copying codes, you are allowed to explore and exploit all resources you can find.
- Also, you are encouraged to add commandline arguments, and/or develop a more complex interface, to deal with the new functionalities.
 - It is recommended to separate your code before and after implementing the 'room', since you may be going to hard code some features and could mess up what you have done.

- Apart from sharing or copying codes, you are allowed to explore and exploit all resources you can find.
- Also, you are encouraged to add commandline arguments, and/or develop a more complex interface, to deal with the new functionalities.
 - It is recommended to separate your code before and after implementing the 'room', since you may be going to hard code some features and could mess up what you have done.
- **Make sure they are all documented.**

Questions?