【OpenGL による 3D 描画の基礎知識】

今回は３D 描画に JOGL を使用する。

※JOGL…C++でかかれている OpenGL を Java 化したもので、Sun Microsystems が開発を行っている

- OpenGL は 3D 描画ライブラリとしては抽象度が低く、ハードウエアに近いライブラリ。これに対し抽象度が高いライブラリの代表格が Java 3D。
- Java3D はインストールが必要、JOGL は自分でパス設定する。また、資料の数では OpenGL が多いので、勉強しやすい。
- OpenGL はもともと C++でかかれているので、JOGL のソースは独特の書き方になる。
- Java 3D は内部的に OpenGL などを使っているので、OpenGL がわかっていれば Java 3D への敷居は低いと思われる。

＜パネルの宣言＞

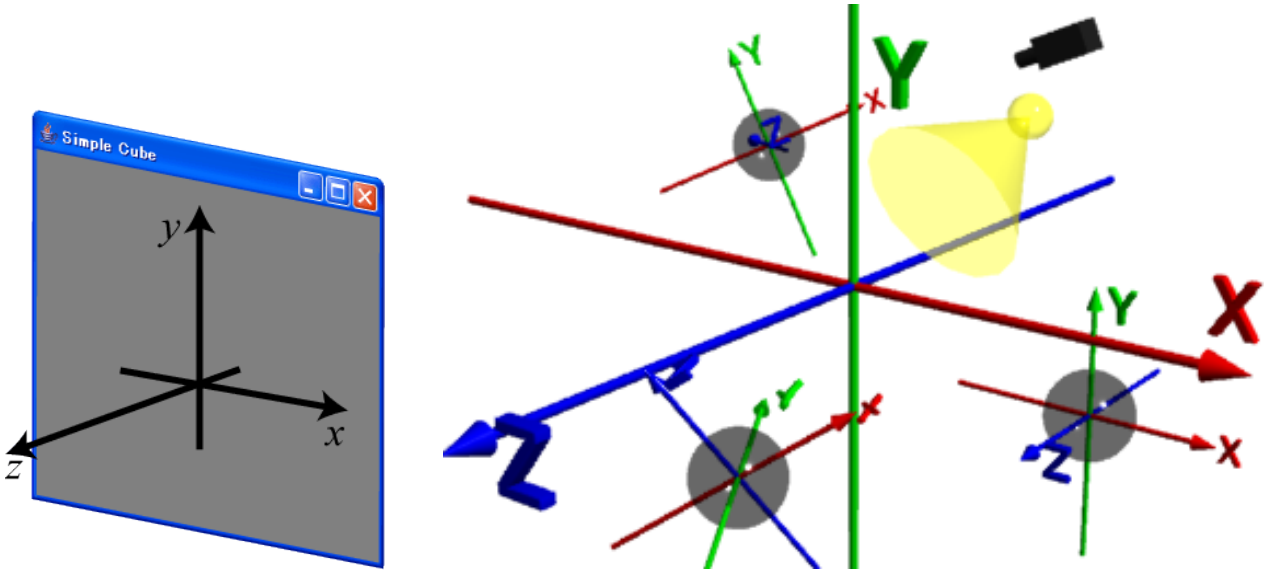
```
import javax.media.opengl.GLEventListener;
import javax.media.opengl.GLJPanel;
public class WaterPanel extends GLJPanel implements GLEventListener {・・・・・}
```

＜GLEventListener にあるメソッド＞

メソッド	説明
init	初期化時にコールされるメソッド
reshape	表示領域が変更されたときにコールされるメソッド
display	描画を行うときにコールされるメソッド
displayChanged	表示の切り替えが発生した場合にコールされるメソッド

init メソッドは起動時に 1 度だけコールされ, reshape はフレームのサイズを変更したときなどにコールされます。描画を行うには display メソッドに記述します。

＜座標の概念＞



左図：OpenGL の座標は上図のようになる。特に、y 座標が上向きのところが
２D の場合と大きく異なる。

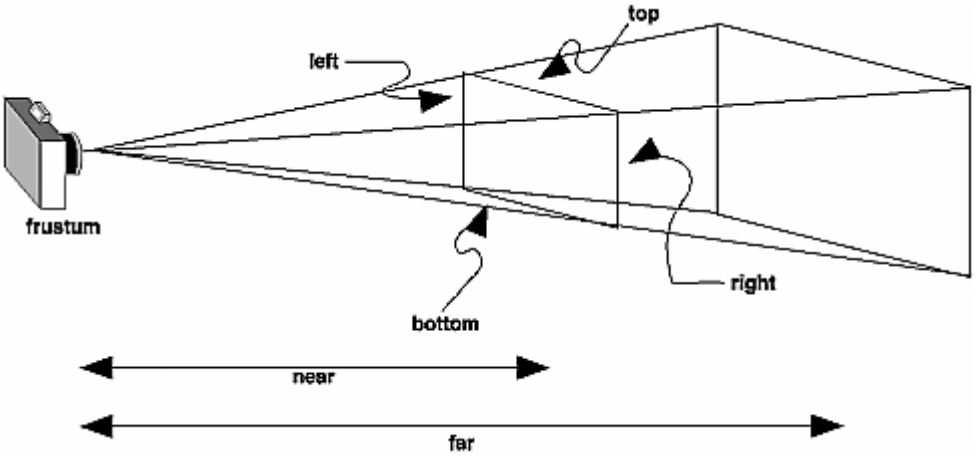
右図：ワールド座標、各物体のローカル座標（Object 座標）、光源の位置、視点

＜視点（カメラ)の設定＞

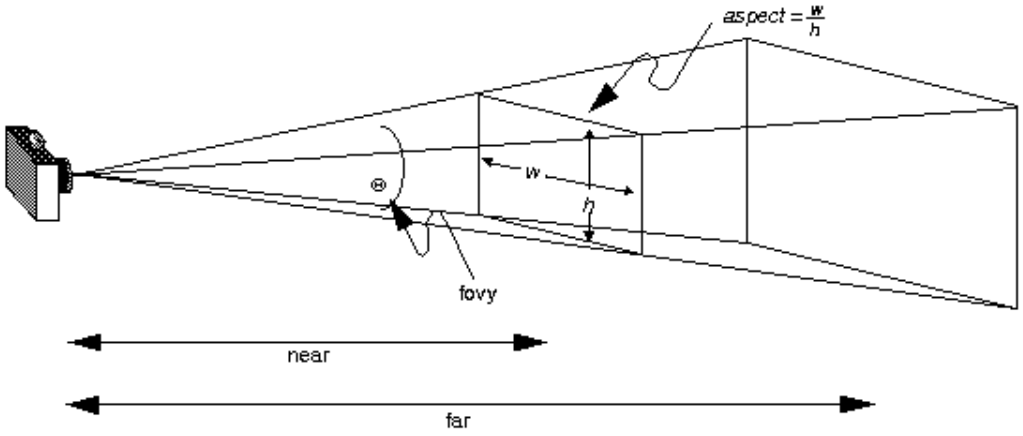
1. カメラを世界座標系のどこに置き、どちらを向かせるか
2. カメラの画角やどこまでの距離を写すか
画像認識や計測の世界では、一（1）をカメラの外部パラメータ、(2) をカメラの内部パラメータと言う。
OpenGL では、(1) の情報を modelview 行列で、(2) の情報を projection 行列で保持している。
※modelview 行列、projection 行列については後述

(1)の情報は、
glulookat(視点の位置x,y,z, 視界の中心位置の参照点座標x,y,z, 視界の上方向のベクトルx,y,z)
関数で設定できる。これは、実質的には物体の平行移動や拡大・縮小、回転で実現可能なので、modelview 行列にもつことになる。

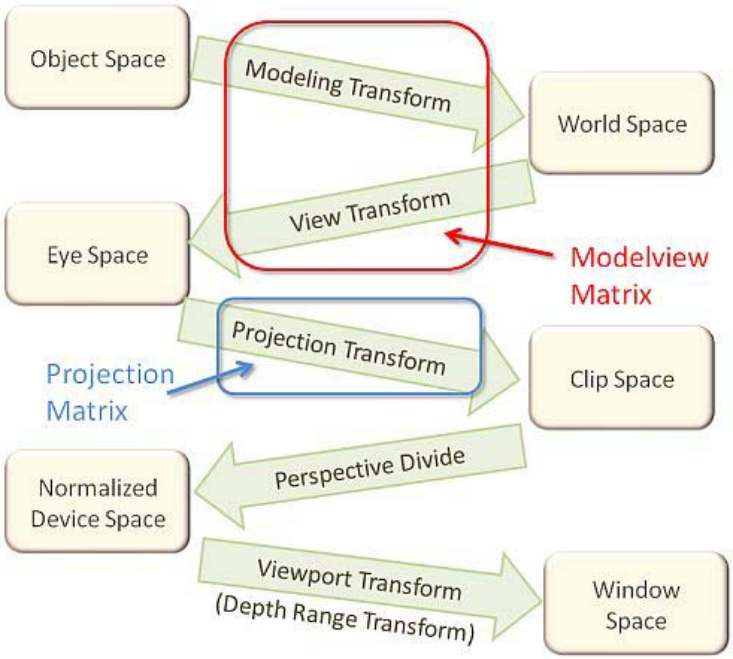
(2) の情報を設定する。情報は projection 行列に保持されることになるである。
遠近法を使って３D のものを２D に投影する方法は２通りの関数がある。
A. glFrustum(left, right, bottom, top, near, far)



B. gluPerspective(fovy, aspect, near, far)・・・左右対称に投影する



＜座標変換の概念＞



◆Object Space(オブジェクト座標系) or Model Space
オブジェクト座標系は、描画物基準の座標系。
例えばティーポットがあったら、その中心に原点があるような座標系。
物体基準なので、物体毎に座標系があるということになる。
◆World Space(ワールド座標系)
ワールド座標系は、空間全体の座標系。
どこが「上」かとか、３次元空間における物体の位置がわかる座標系。
例えば、物体をオブジェクト座標系からワールド座標系に変換すると、
　物体Ａはワールド座標系の原点から(1,2,3)だけ平行移動した位置に物体が配置され,
　物体Ｂはワールド座標系の原点から(-2,7,0)だけ平行移動した位置に物体が配置される、
といった感じ。
◆Eye Space(視点座標系) or View Space
視点座標系は、視点の位置が原点となる座標系。
つまり、この座標系に変換するというは、
物体が平行移動などの処理をしているということである。
◆Clip Space(クリップ座標系)
クリップ空間は、いわゆる表示範囲のこと。
※Projection Matrix
「Eye Space」から「Clip Space」への変換は投影変換とよばれるもので、
4×4 の行列を演算することで変換できる。
この時の行列が「Projection Matrix(プロジェクション行列)」よばれる。
◆Normalized Device Space(正規デバイス座標系)
クリップ座標系は(x,y,z,w)の同次座標系。
この要素を「w」で割った結果の座標系が「正規デバイス座標系」
ここまでの処理で、(-1, -1, -1)～(1, 1, 1)の範囲に座標データが収まるようになる。
◆Window Space(ウィンドウ座標系)
ウィンドウ内のピクセルの単位での座標系。
※左「上」が原点の場合や、左「下」が原点の場合などがある。
OpenGL は左「下」を原点にしている。
ここへの変換が「ビューポート変換」とよばれるものである。

＜Modelview 行列＞

◆Modelview Matrix

「Object Space」から「Eye Space」へ変換する時は、
「モデリング変換」と「ビューイング変換」の２回の変換が行われている。
この変換はそれぞれ 4×4 の行列を演算する事になるが、
一般にこれらはまとめてしまって、「ModelView Matrix(モデルビュー行列)」
を使用している。

◆Modelview 行列理解のための座標変換基礎

【拡大縮小】

X/Y/Z 軸方向に拡大縮小する（(Sx, Sy, Sz)で指定)。1.0 のときは等倍。

$$[X'\ Y'\ Z'\ W'] = [X\ Y\ Z\ 1]\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

X' = X * Sx;
Y' = Y * Sy;
Z' = Z * Sz;

【X 軸中心に回転】

X 軸を中心に θ x 回転させる。

$$[X'\ Y'\ Z'\ W'] = [X\ Y\ Z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta x & \sin\theta x & 0 \\ 0 & -\sin\theta x & \cos\theta x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

X' = X;
Y' = Y * cosθx - Z * sinθx;
Z' = Y * sinθx + Z * cosθx;

【Y 軸中心に回転】

Y 軸を中心に θ y 回転させる。

$$[X'\ Y'\ Z'\ W'] = [X\ Y\ Z\ 1]\begin{bmatrix} \cos\theta y & 0 & -\sin\theta y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta y & 0 & \cos\theta y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

X' = X * cosθy + Z * sinθy;
Y' = Y;
Z' = -X * sinθy + Z * cosθy;

【Z 軸中心に回転】

Z 軸を中心に θ z 回転させる。

$$[X'\ Y'\ Z'\ W'] = [X\ Y\ Z\ 1]\begin{bmatrix} \cos\theta z & \sin\theta z & 0 & 0 \\ -\sin\theta z & \cos\theta z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

X' = X * cosθz - Y * sinθz;
Y' = X * sinθz + Y * cosθz;
Z' = Z;

【平行移動】
移動後のワールド座標位置を(Px, Py, Pz)とする。

$$[X'\ Y'\ Z'\ W'] = [X\ Y\ Z\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Px & Py & Pz & 1 \end{bmatrix}$$

$$X' = X + Px;$$
$$Y' = Y + Py;$$
$$Z' = Z + Pz;$$

＜光と素材＞
光は以下の 4 種類に分けてあてる。環境光が強すぎると影ができにくい。また、鏡面反射がつよいとテカテカした質感になる。

鏡面反射	光の入射角度により，反射角度が一意的に決まる反射光 ガラスや金属など
拡散反射	入射角度によらず，反射があらゆる方向に散乱する反射光 プラスティックなど
環境光反射	いわゆる地あかり 方向をもたず全体的に照らす環境光に対する反射光
放射	物体から放射される光

物体の反射率も上記の 4 種類ある。光が当たっていても、物質が反射しなければ物質の色は見えない。また、それに加えて光が当たった部分のハイライトの大きさを表す光輝パラメータも指定できる。
今回のサンプルでは、光は白色光とし、鏡面反射率・拡散反射率を適当に小さめにしたうえで、環境光の反射率を各物体ごとに大きくかえ、赤や青といった色を表現する。今回は放射は使わない。
つまり、環境光の反射率は色を決定するものと考えてよい。他の鏡面反射・拡散反射はライトアップされた感じをだすためのものとなる。

鏡面・反射・環境光の質感との関係は以下のサイトにわかりやすくまとめてあるので、参照すること。
http://www.komoto.org/opengl/sample07.html