授業のおおまかな予定

4/05 ガインダンス
4/12
4/19
4/26
5/10
5/17

2次元グラフィックス
- 四角形，円形，直線の描画
- カラー画像
- 座標変換

5/24 休講
5/27(月) 6限 補講(OD2)
5/31 (3201教室集合) ← **中間試験**(小テスト)
6/07
6/14
6/21

3次元グラフィックス
- 物体形状の表現
- シェーディング
- 隠面消去
- OpenGL
- CG作品

6/28 休講
7/01(月) 6限 補講(OD2)
7/05 休講
7/12
7/19
試験期間中 ← **期末試験**

提出締切：8月初旬
詳細は後日発表

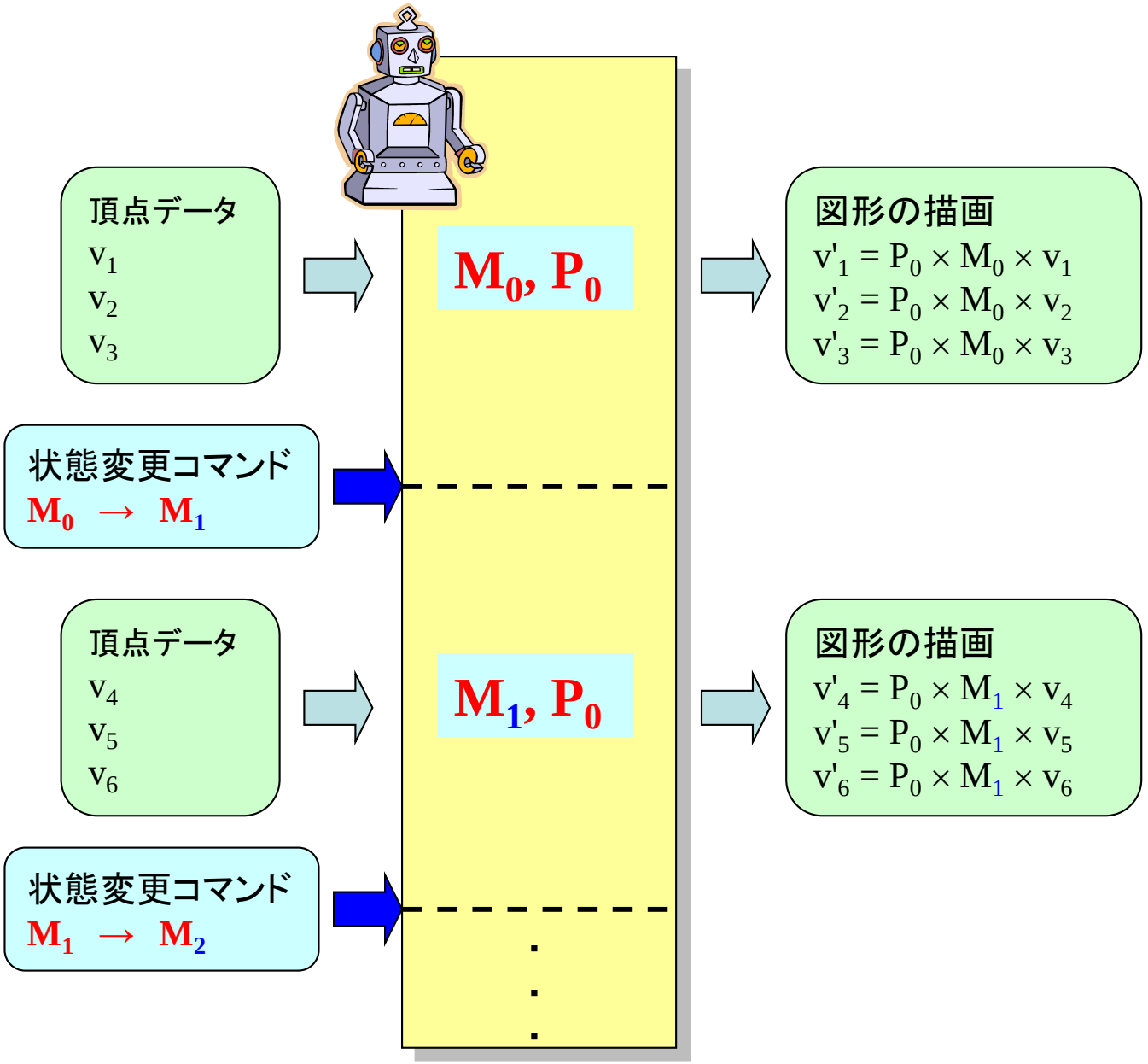
レポート	10%
出席	10%
修了作品	20%
中間試験	20%
期末試験	40%

レポート
≒ C言語プログラム＋α

受講するためには
→「基礎プログラミング」程度のC言語の知識が必須

OpenGL状態マシンの動作

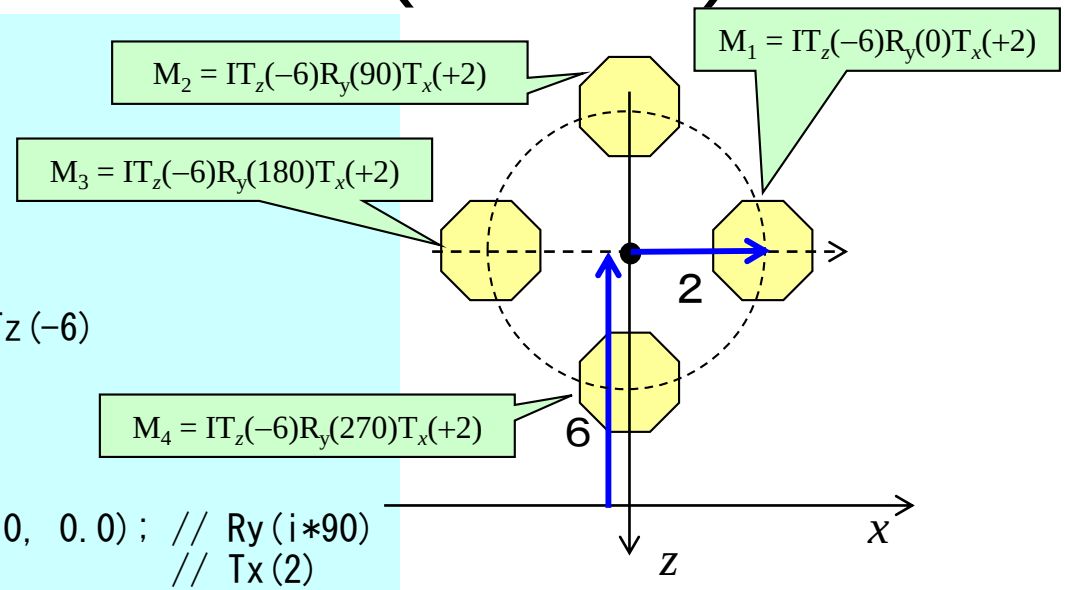
復習

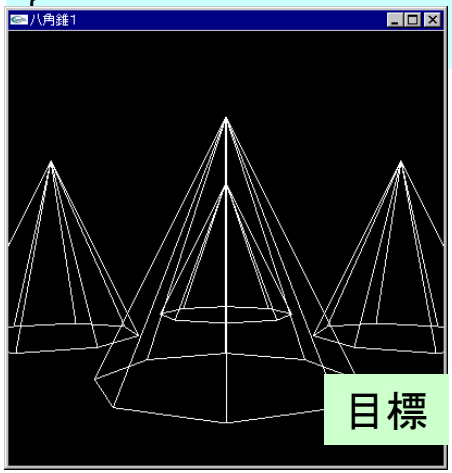


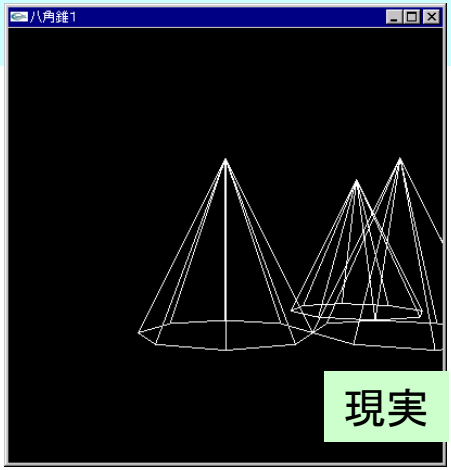
幾何変換行列の操作(失敗例)

復習

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();                    // I
    glTranslatef(0.0, 0.0, -6.0);        // Tz(-6)

    int i;
    for (i = 0; i < 4; i++)
    {
        glRotatef(RotAngle + i*90, 0.0, 1.0, 0.0); // Ry(i*90)
        glTranslatef(+2.0, 0.0, 0.0);              // Tx(2)
        OctPyramid();
    }

    glFlush();
}
```







ループに入る前

$M = IT_z(-6)$

1回目のループ(i = 0)

$M_1 = IT_z(-6) \times R_y(0)T_x(+2)$

2回目のループ(i = 1)

$M_2 = M_1 \times R_y(90)T_x(+2)$

$= IT_z(-6)R_y(0)T_x(+2)R_y(90)T_x(+2)$

3回目のループ(i = 2)

$M_3 = M_2 \times R_y(180)T_x(+2)$

$= IT_z(-6)R_y(0)T_x(+2)R_y(90)T_x(+2)R_y(180)T_x(+2)$

・・・

OpenGLにおけるローカル座標系の扱い(1)

復習

Example11-3

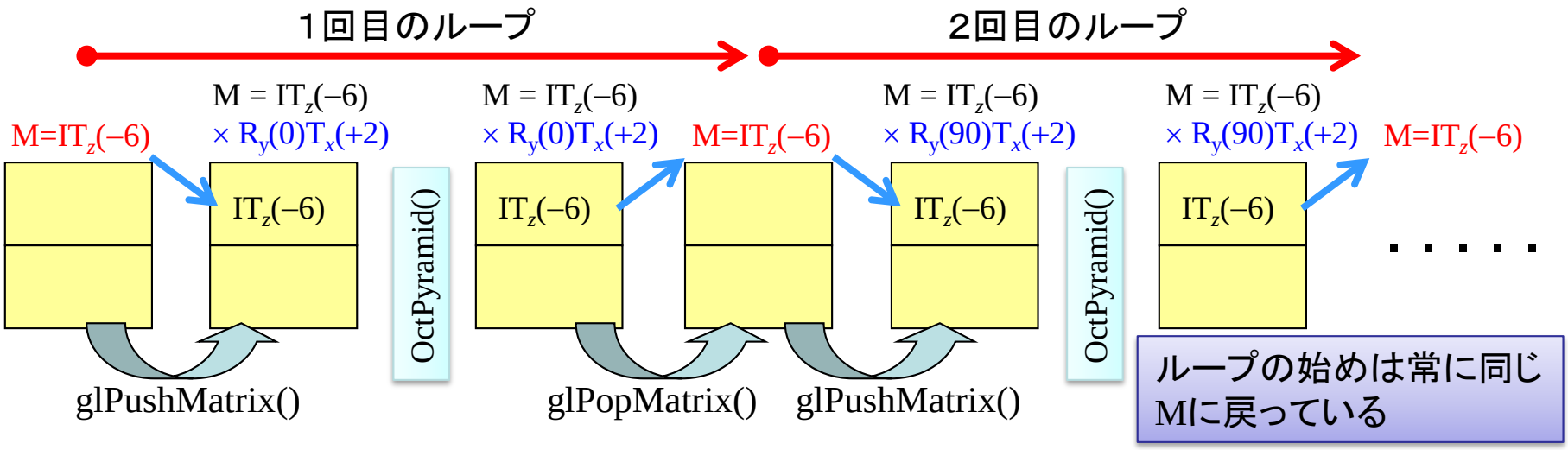
```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    int i;
    for (i = 0; i < 4; i++)
    {
        glPushMatrix();
        glRotatef(RotAngle + i*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        OctPyramid();
        glPopMatrix();
    }
    glFlush();
}
```

$M=IT_z(-6)$

変換行列の共通部分

$\times R_y(i \times 90)T_x(+2)$

変換行列の変更部分

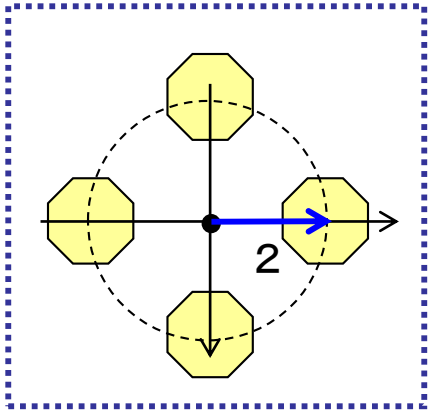


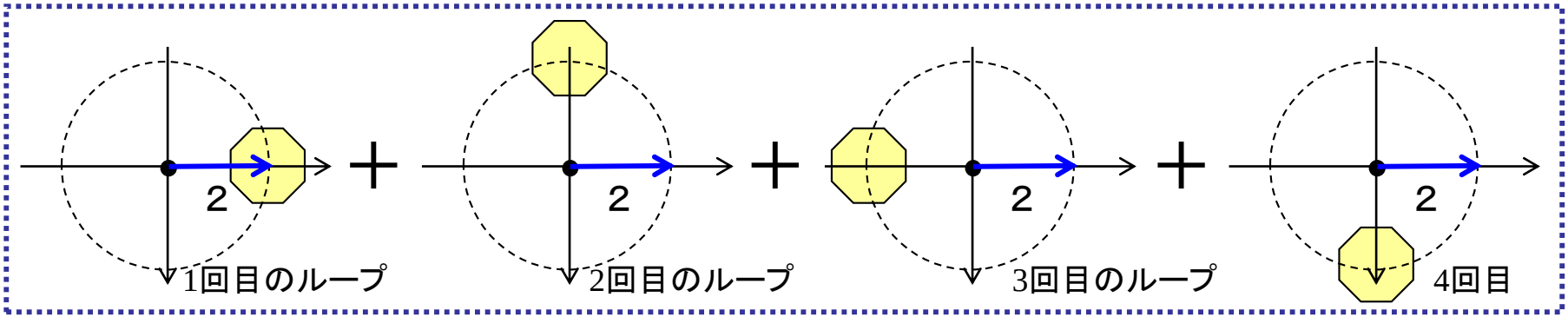
OpenGLにおけるローカル座標系の扱い(2)

復習

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    int i;
    for (i = 0; i < 4; i++)
    {
        glPushMatrix();
        glRotatef(RotAngle + i*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        OctPyramid();
        glPopMatrix();
    }
    glFlush();
}
```

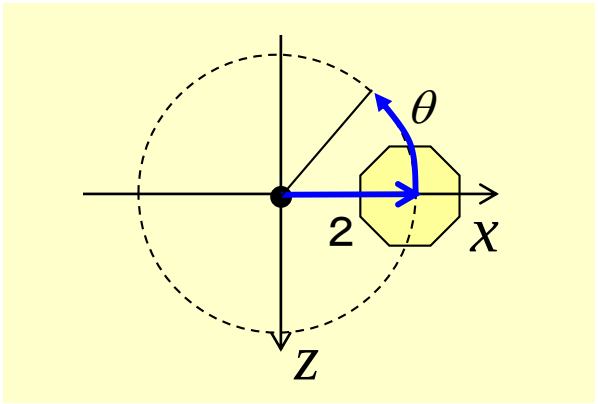
点線内のソースで描かれる図形全体をz方向に−6移動

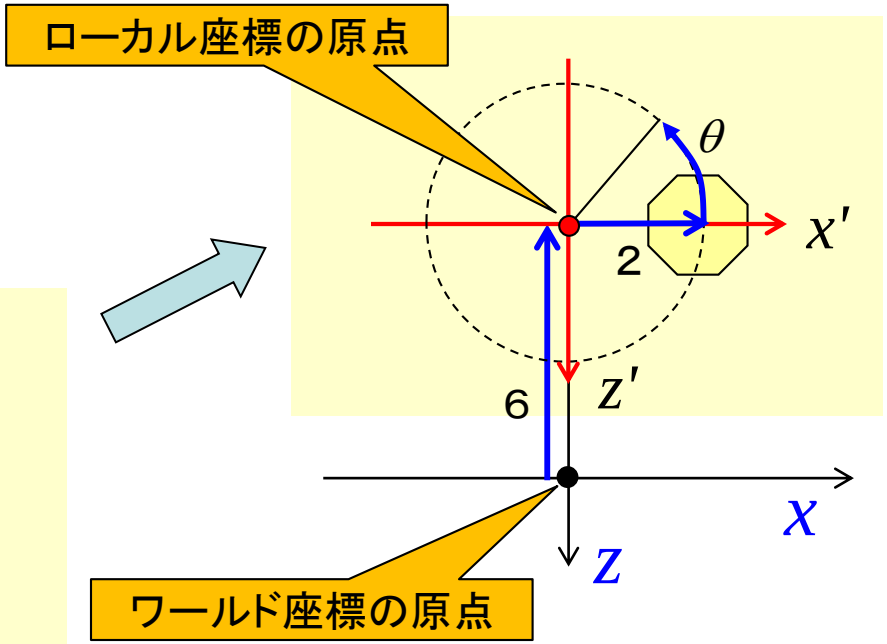




ワールド座標系とローカル座標系(1)

復習





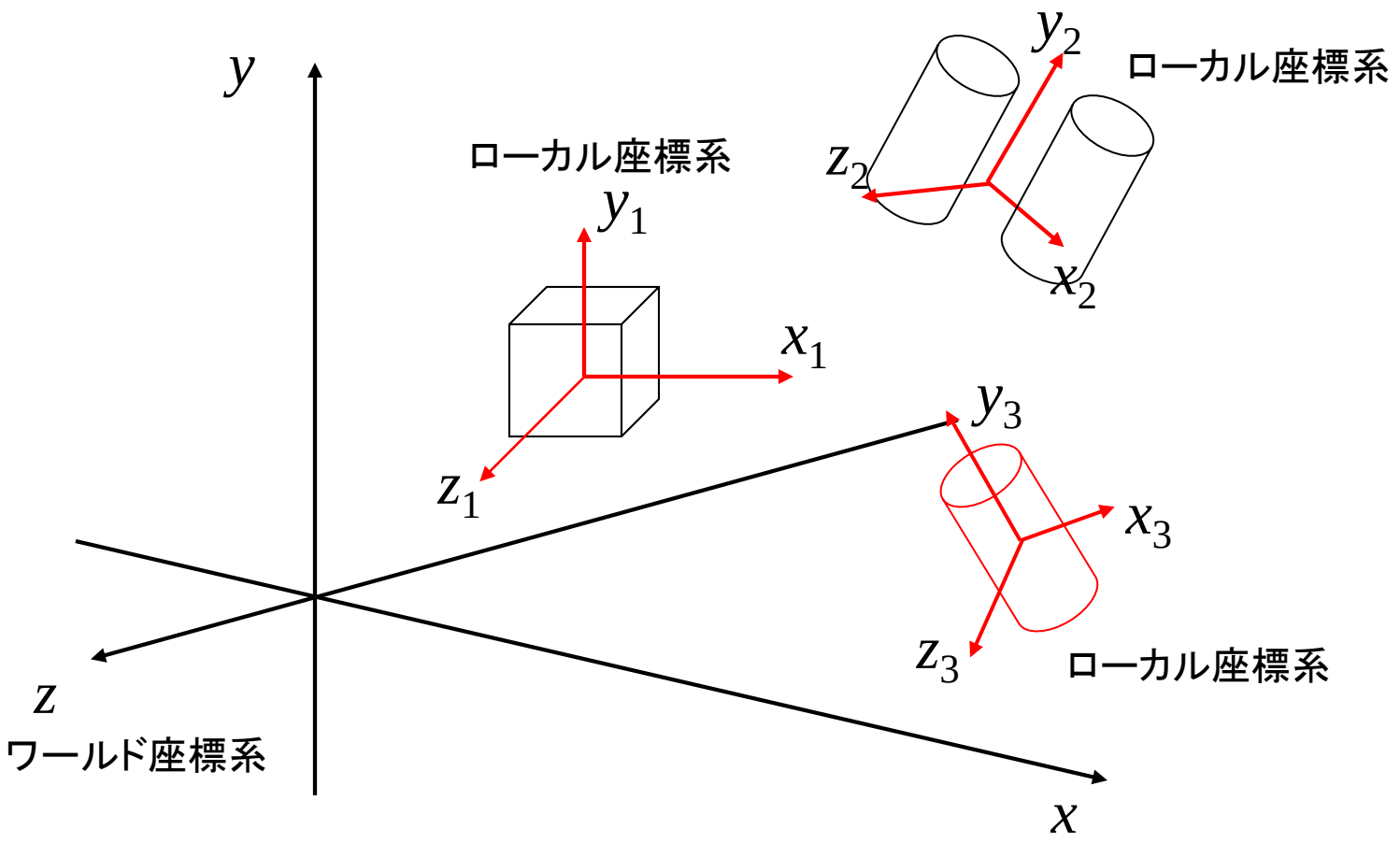
(i) 図形を距離2だけ移動
(ii) 図形を角度θだけ回転

(iii) 全体を距離6だけ負のz軸方向に移動

(x, y) ワールド座標系
(x', y') ローカル座標系

ワールド座標系とローカル座標系(2)

復習



ワールド座標系	描きたいシーンに一つだけ定義される座標系
ローカル座標系	物体ごと，あるいは物体の集団ごとに定義される座標系

OpenGLにおけるローカル座標系の扱い(4)

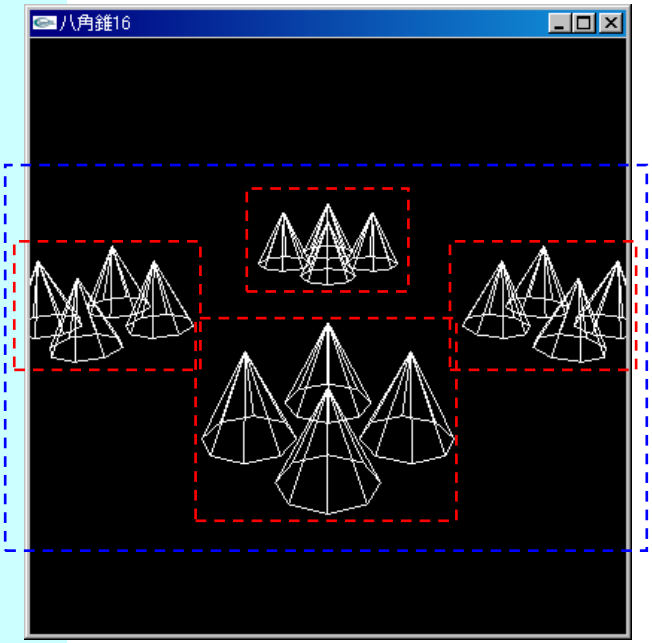
復習

Example11-5

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    int j;
    for (j = 0; j < 4; j++)
    {
        glPushMatrix();
        glRotatef(j*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        int i;
        for (i = 0; i < 4; i++)
        {
            glPushMatrix();
            glRotatef(i*90, 0.0, 1.0, 0.0);
            glTranslatef(+0.5, 0.0, 0.0); //半径0.5で回転
            glScalef(0.3, 0.3, 0.3 );     //0.3倍に縮小
            OctPyramid();
            glPopMatrix();
        }
        glPopMatrix();
    }

    glFlush();
}
```

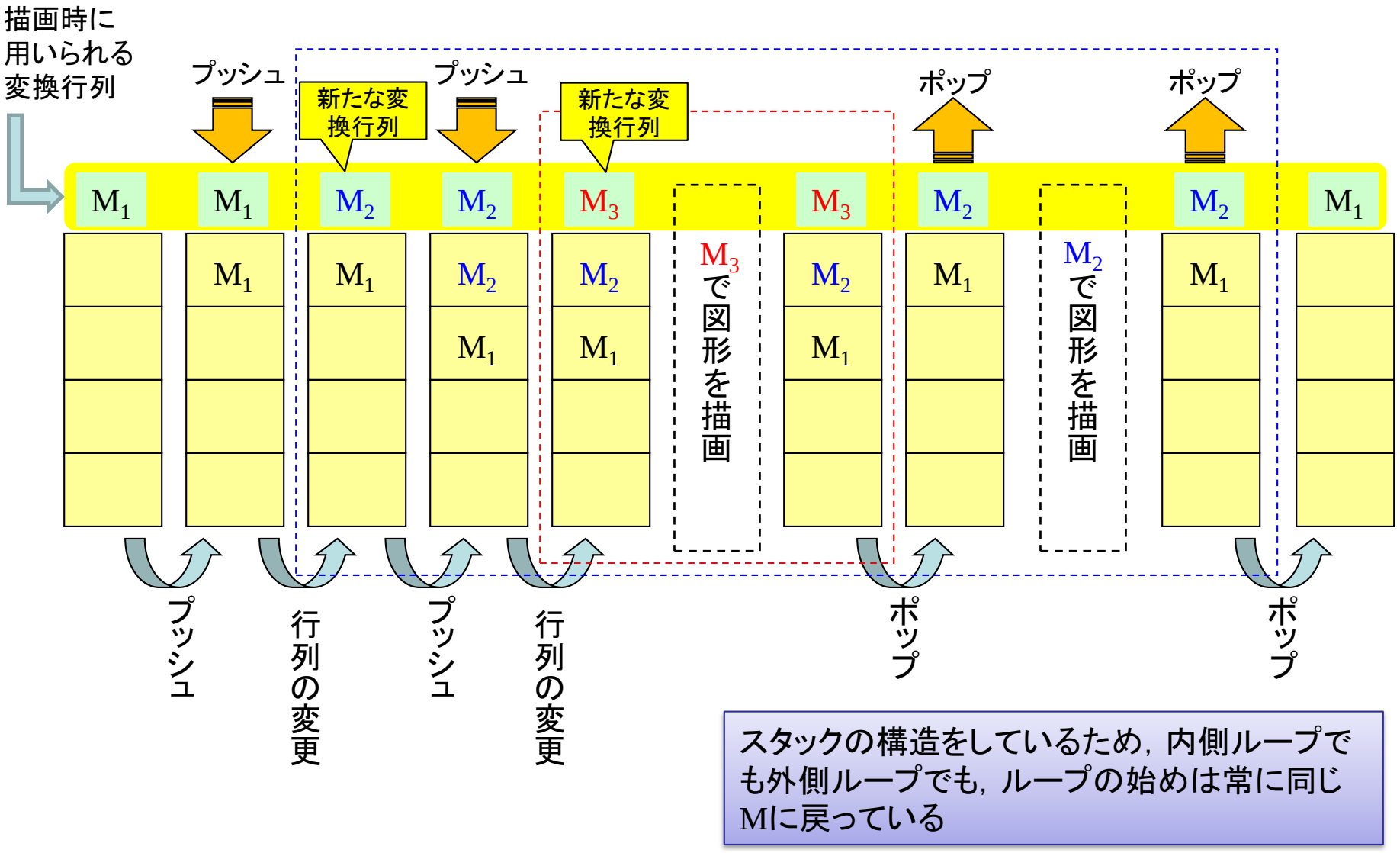
点線内のソースで描かれる図形全体をz方向に−6移動

点線内のソースで描かれる図形全体を半径2で回転



幾何変換行列スタックの変化(概念)

復習



スタックの構造をしているため，内側ループでも外側ループでも，ループの始めは常に同じMに戻っている

基本課題11

復習

図1のサイズの八角錐をローカル座標で図2の様に並べ，その組み合わせをワールド座標で図3の様に直径4の円周上に並べて表示したい．ペイントハンドラdisplay()を作成してプログラムを完成しなさい．なお，OctPyramid()及び下のmain()を利用しなさい．

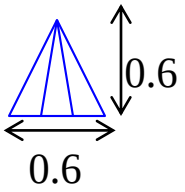


図1 八角錐のサイズ

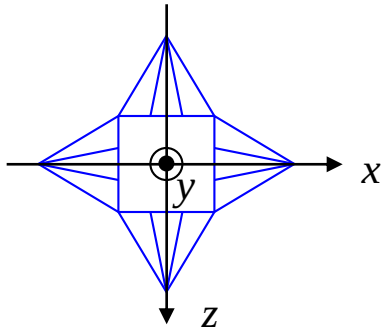


図2 ローカル座標

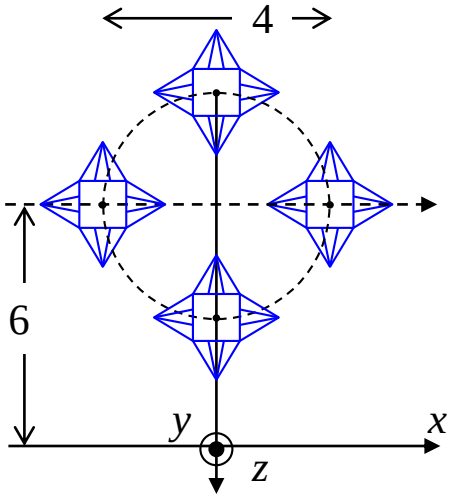


図3 ワールド座標

Report11-1

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
void OctPyramid(void);  //既存の関数を利用
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("八角錐16");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 0.0, 10.0);
    gluLookAt(0, 7, 0,    0, 0, -5,    0, 1, 0);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```

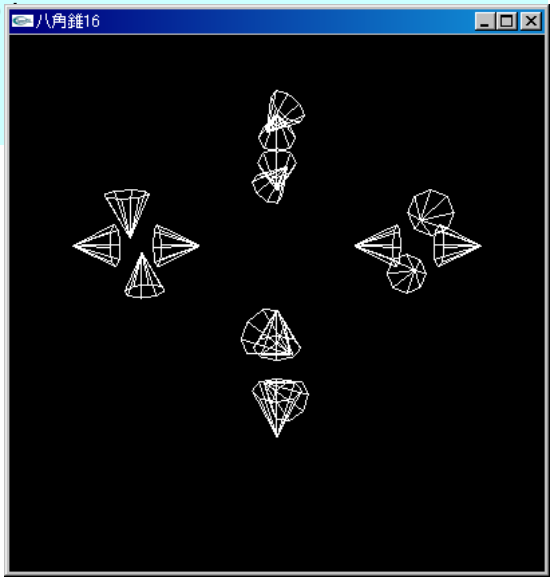
ウィンドウの名前を各自の学籍番号と氏名にすること

① Wordのレポートにソースプログラムと実行結果(glutウィンドウ)の画面コピーを貼りつけて提出

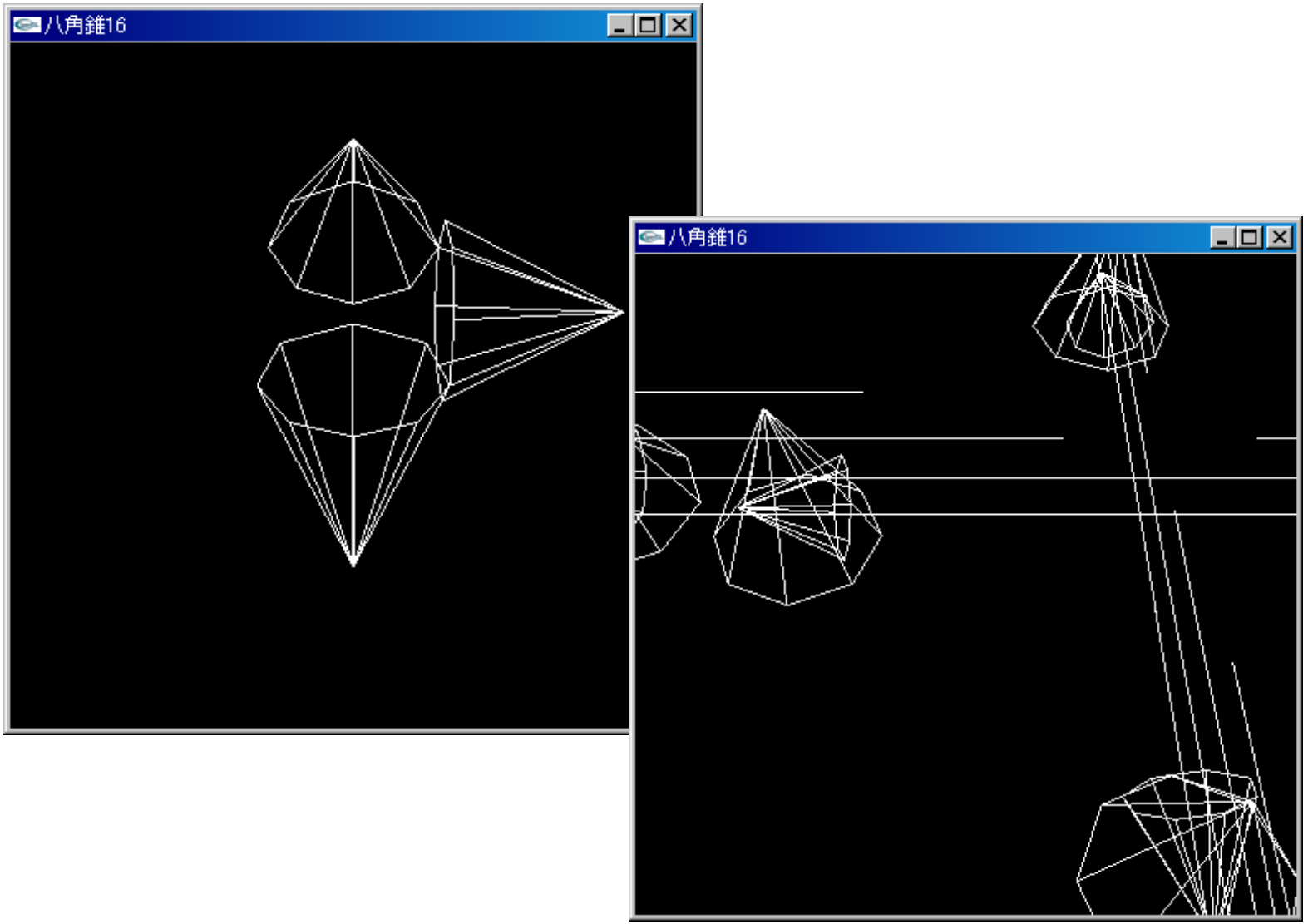
基本課題11 解答例

A君解答

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    int j;

    for (j = 0; j < 4; j++)
    {
        glPushMatrix();
        glRotatef(j*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);

        {
        glPushMatrix(); //z軸周りに-90度回転して描く
        glTranslatef(+0.6, 0.0, 0.0);
        glRotatef(-90, 0.0, 0.0, 1.0);
        glScalef(0.3, 0.3, 0.3); // S
        OctPyramid();
        glPopMatrix();

        glPushMatrix(); //z軸周りに+90度回転して描く
        glTranslatef(-0.6, 0.0, 0.0);
        glRotatef(+90, 0.0, 0.0, 1.0);
        glScalef(0.3, 0.3, 0.3); // S
        OctPyramid();
        glPopMatrix();

        glPushMatrix(); //z軸周りに+180度回転して描く
        glTranslatef(0, -0.6, 0);
        glRotatef(180, 1, 0, 0);
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef(180, 0.0, -0.3, 1.0);
        glScalef(0.3, 0.3, 0.3); // S
        OctPyramid();
        glPopMatrix();

        glPushMatrix(); //z軸周りに-180度回転して描く
        glTranslatef(0, 0.6, 0);
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef(-180, 0.0, -0.3, 1.0);
        glScalef(0.3, 0.3, 0.3); // S
        OctPyramid();
        glPopMatrix();
        }
        glPopMatrix();
    }
    glFlush();
}
```

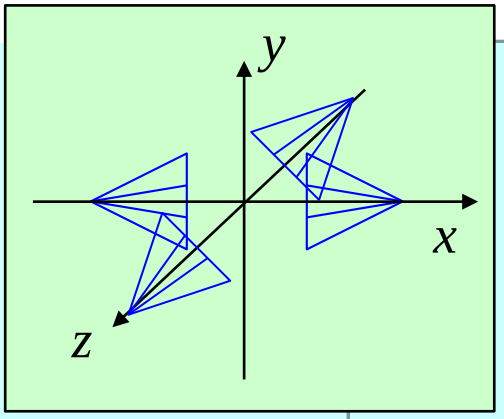


誤りの例

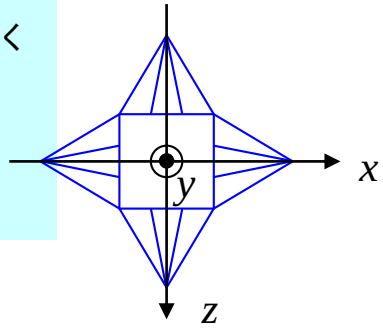


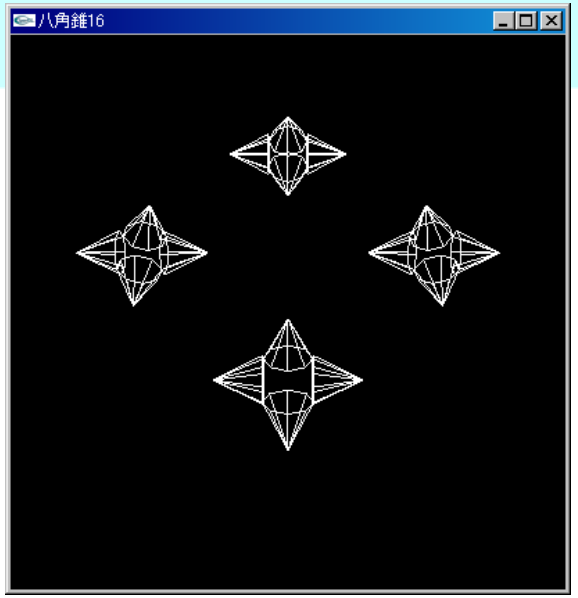
基本課題11 解答例

B君解答




```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    int i;
    for(i=0;i<4;i++)
    {

        glPushMatrix();
        glRotatef(90.0*i, 0.0, 1.0, 0.0);
        glTranslatef(2.0, 0.0, 0.0);

            glPushMatrix(); //z軸周りに-90度回転して描く
            glTranslatef(+0.6, 0.0, 0.0);
            glRotatef(-90, 0.0, 0.0, 1.0);
            glScalef(0.3, 0.3, 0.3 );
            OctPyramid();
            glPopMatrix();

            glPushMatrix(); //z軸周りに+90度回転して描く
            glTranslatef(-0.6, 0.0, 0.0);
            glRotatef(+90, 0.0, 0.0, 1.0);
            glScalef(0.3, 0.3, 0.3 );
            OctPyramid();
            glPopMatrix();

            glPushMatrix(); //x軸周りに-90度回転して描く
            glTranslatef(0.0, 0.0, +0.6);
            glRotatef(+90.0, 1.0, 0.0, 0.0);
            glScalef(0.3, 0.3, 0.3 );
            OctPyramid();
            glPopMatrix();

            glPushMatrix(); //x軸周りに+90度回転して描く
            glTranslatef(0.0, 0.0, -0.6);
            glRotatef(-90.0, 1.0, 0.0, 0.0);
            glScalef(0.3, 0.3, 0.3 );
            OctPyramid();
            glPopMatrix();

        glPopMatrix();
    }
    glFlush();
}
```

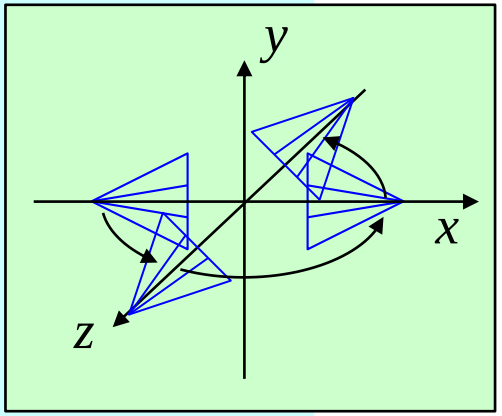


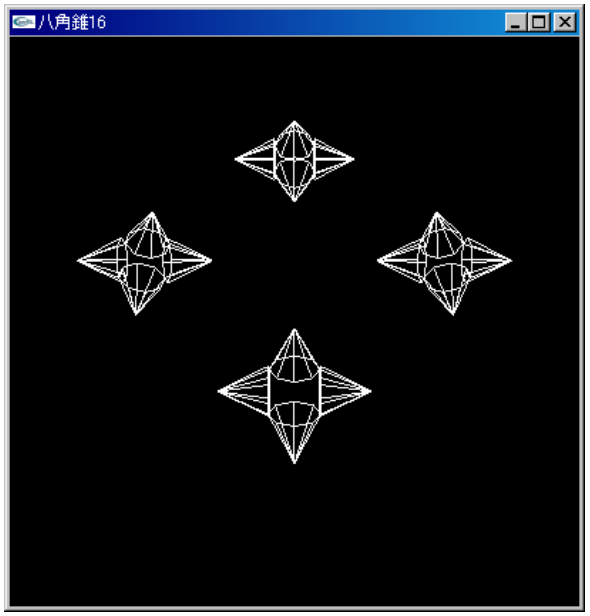


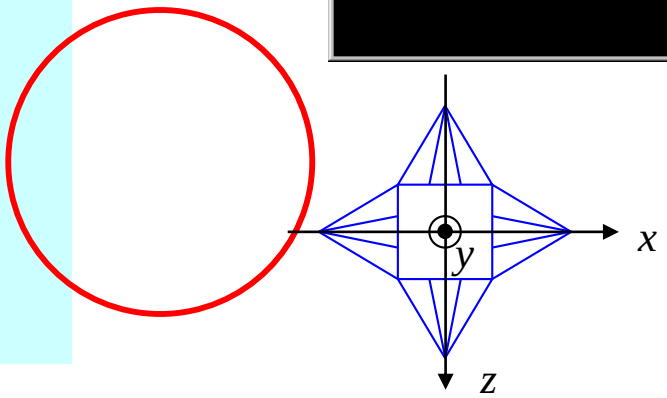
基本課題11 解答例

C君解答

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    int j;
    for (j = 0; j < 4; j++)
    {
        glPushMatrix();
        glRotatef(j*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        int i;
        for (i = 0; i < 4; i++)
        {
            glPushMatrix();
            glRotatef(i*90, 0.0, 1.0, 0.0);
            glTranslatef(-0.6, 0.0, 0.0); //半径.6で回転
            glRotatef(90, 0.0, 0.0, 1.0);
            glScalef(0.3, 0.3, 0.3 ); //0.3倍に縮小
            OctPyramid();
            glPopMatrix();
        }
        glPopMatrix();
    }
    glFlush();
}
```





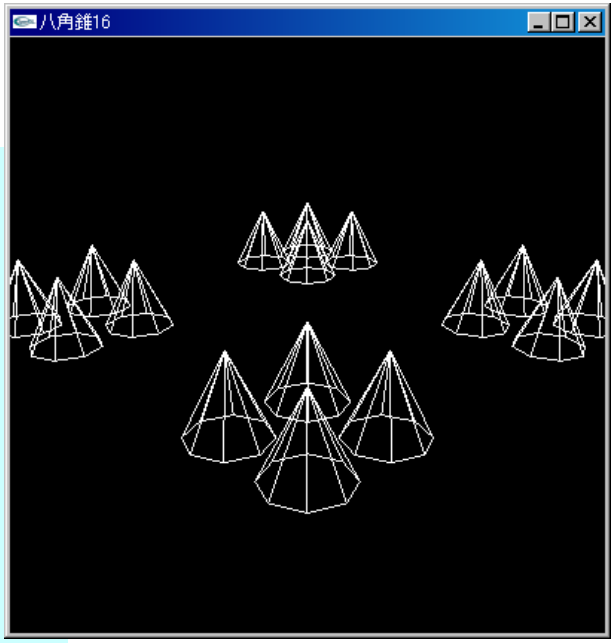


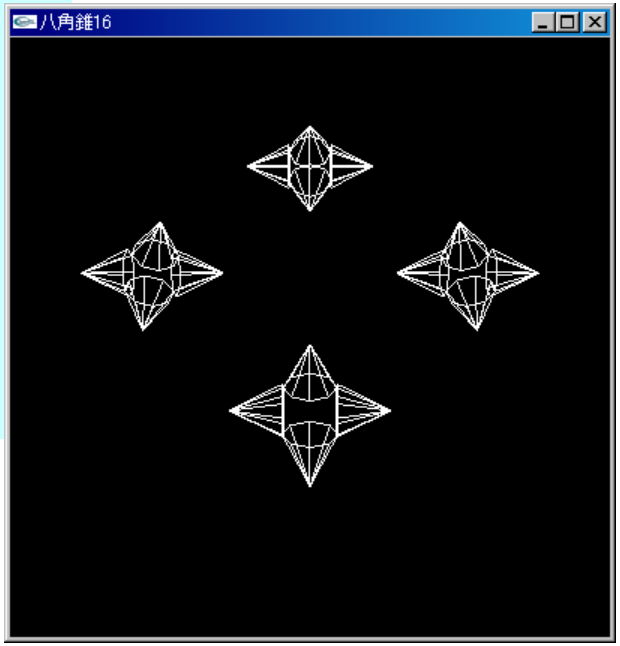
✓ glPushMatrix()とglPopMatrix()は対で用いる
✓ 同じ図形を2度，3度と上書きしない

視点と視線

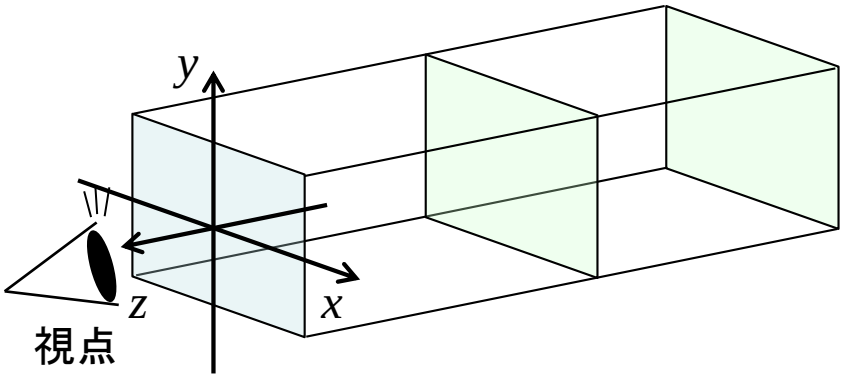
```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
void OctPyramid(void);
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("八角錐16");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 0.0, 10.0);
    gluLookAt(0, 7, 0,   0, 0, -5,   0, 1, 0);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```

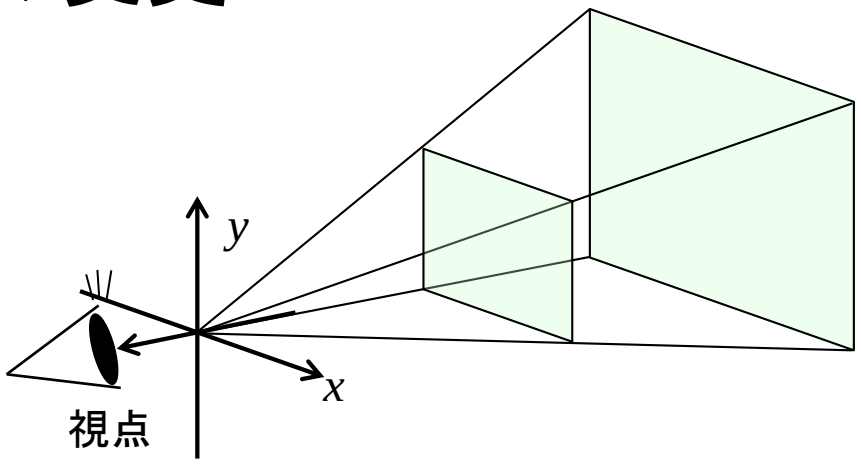
視点および視線
を変更している

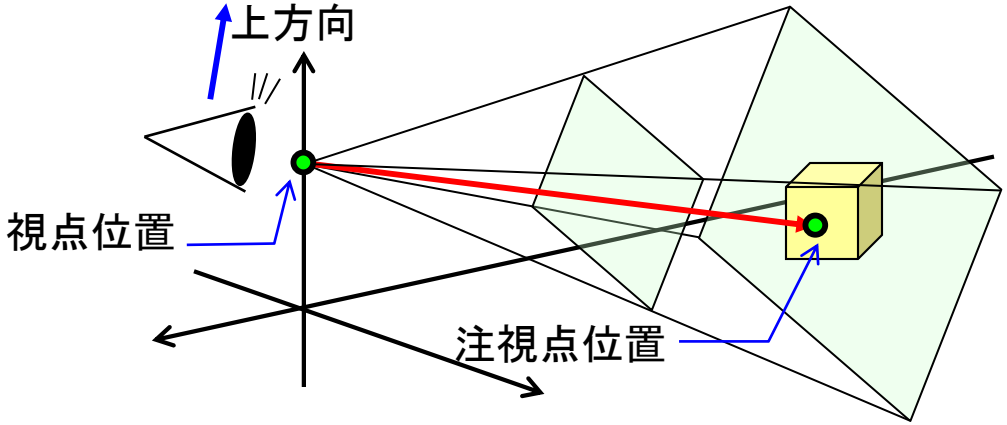




視点の変更








```
gluLookAt(x0, y0, z0, x1, y1, z1, ux, uy, uz)
```

視点位置　注視点位置　上方向ベクトル

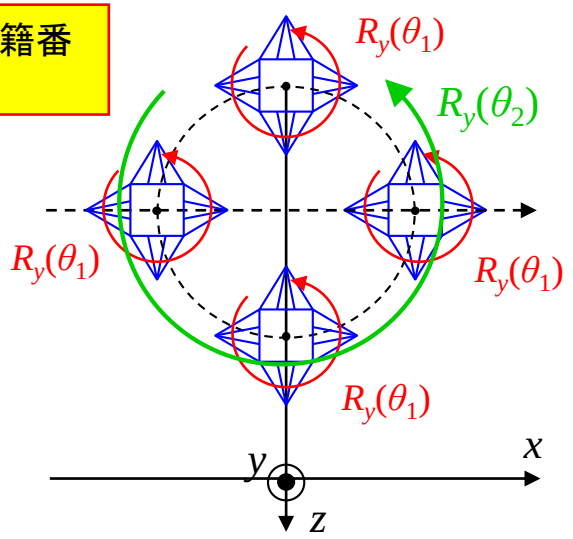
この関数はOpenGLの投影変換行列Pを変更する

復習

発展課題11

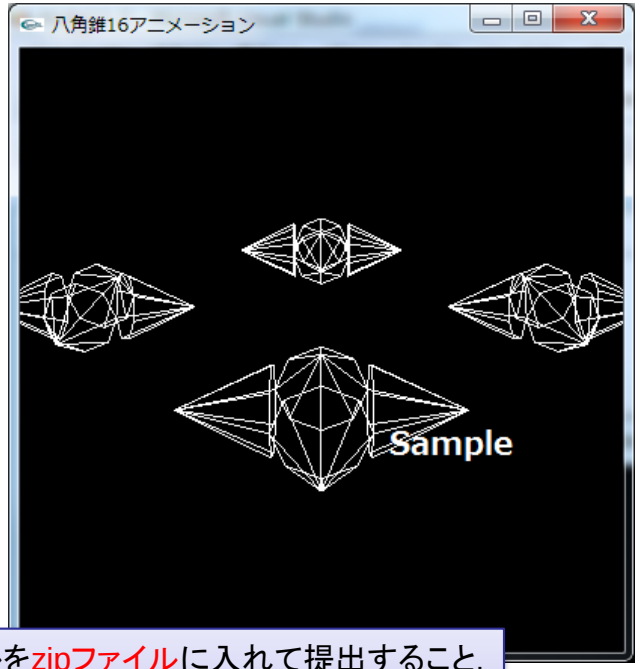
ウィンドウの名前を各自の学籍番号と氏名にすること

基本課題の図形を実行例(Kadai11-2.exe)のとおりにアニメーションしなさい．なお，このアニメーションでは，右図のとおりに2種類の回転が用いられている．このうち，$R_y(\theta_2)$は$R_y(\theta_1)$ の2分の1の回転速度でゆっくり回っている．このアニメーションをOctPyramid()及び下のmain()を利用して完成しなさい．



Report11-2

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
void OctPyramid(void);  //既存の関数を利用
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("八角錐16アニメーション");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 0.0, 10.0);
    gluLookAt(0, 2, 0,   0, 0, -5,   0, 1, 0);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutIdleFunc(IncAngle);
    glutMainLoop();
}
```



以下の二つのファイルをzipファイルに入れて提出すること．

① Wordのレポート
　ソースプログラム
　画面コピー
② 実行プログラム(〇〇〇.exe)

発展課題11 解答例

D君解答

```
double RotAngle = 0.0;
double RotAngle2 = 0.0;

void IncAngle(void)
{
    RotAngle = RotAngle + 0.01;
    if (RotAngle > 360.0)
        RotAngle = 0;

    RotAngle2 = RotAngle2 + 0.005;
    if (RotAngle > 360.0)
        RotAngle = 0;

    glutPostRedisplay();
}
```

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    int j;
    for (j = 0; j < 4; j++)
    {
        glPushMatrix();
        glRotatef(RotAngle2 + j*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        int i;
        for (i = 0; i < 4; i++)
        {
            glPushMatrix();
            glRotatef(RotAngle + i*90, 0.0, 1.0, 0.0);
            glTranslatef(-0.6, 0.0, 0.0); //半径.6で回転
            glRotatef(90, 0.0, 0.0, 1.0);
            glScalef(0.3, 0.3, 0.3 ); //0.3倍に縮小
            OctPyramid();
            glPopMatrix();
        }
        glPopMatrix();
    }
    glFlush();
}
```

アイドルハンドラによるアニメーションの問題点

- 他のイベントがあるとアイドルイベントは発生しない.
- コンピュータの能力によりイベント発生タイミングが変わる場合がある.

タイマーによる正確なアニメーション

Example12-0

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
#include <stdio.h>

void KeyboardHandler;    //省略
void OctPyramid(void);   //省略

double RotAngle = 0.0;

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glRotatef(RotAngle, 0, 1, 0);
    glRotatef(15, 1, 0, 0);

    OctPyramid();
    glFlush();
}

void IncAngle(int timer)
{
    if (timer != 1)    return;  //1番以外のタイマーは無視
    RotAngle = RotAngle + 0.6;
    if (RotAngle >= 360.0)
        RotAngle = RotAngle - 360.0;

    glutTimerFunc(100, IncAngle, 1);
    glutPostRedisplay();
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("正確なアニメーション");
    glClearColor(0.0, 0.0, 0.0, 1.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutTimerFunc(100, IncAngle, 1);
    glutMainLoop();
}
```

この例では100ミリ秒ごとに発生するタイマーイベントで0.6度づつ回転させている

0.6度/100ms = 6度/秒

360度/(6度/秒) なので60秒で1回転する

図形の回転角度

タイマーイベントハンドラ
void func(int timer)
timer　タイマーの番号

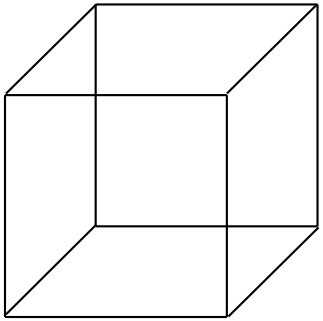
タイマーイベントハンドラを登録してタイマーを起動する.
glutTimerFunc(100, func, 1)

- ✓ この関数の実行後，100ミリ秒経過したらタイマーイベントを発生する.
- ✓ タイマーイベントハンドラとして関数funcを登録する.
- ✓ このタイマーの番号は1番
  ⇒ 複数のタイマーを設定することができる

タイマーイベントハンドラの中で再びタイマー番号1番のタイマーを起動する

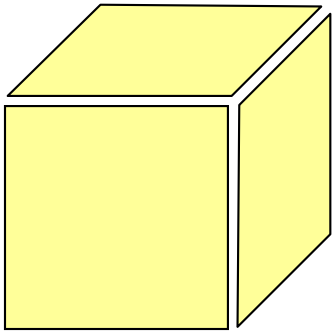
100ミリ秒毎にタイマーイベントが発生する

3次元形状モデルの分類



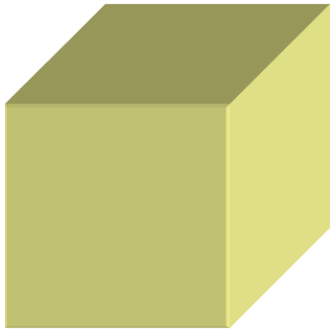
ワイヤーフレームモデル

線の集合



サーフェースモデル

面の集合

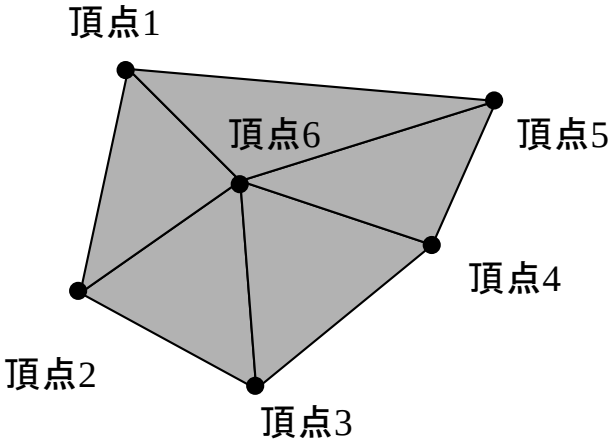


ソリッドモデル

立体領域の集合

稜線の接続や面の接続等の立体を形成する情報.
内側外側の判別等もできる.

面の形状
- **曲面**
  - ベジエ曲面
  - NURBS曲面
- **平面**
  - ポリゴンモデル



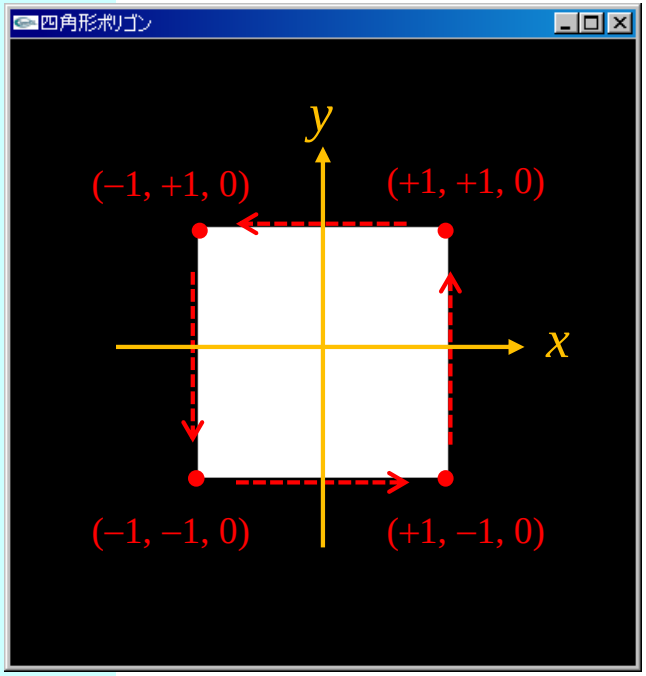
OpenGLでのポリゴンの描画(1)

Example12-1

```
#include "glut.h"
#include <GL/gl.h>
void display(void)
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glBegin(GL_POLYGON);
        glVertex3f(-1, -1, 0);
        glVertex3f(+1, -1, 0);
        glVertex3f(+1, +1, 0);
        glVertex3f(-1, +1, 0);
    glEnd();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("四角形ポリゴン");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 0.0, 10.0);

    glutDisplayFunc(display);
    glutMainLoop();
}
```

頂点を結ぶポリゴンを描画する

辺の長さが2の正方形ポリゴンの頂点を指定

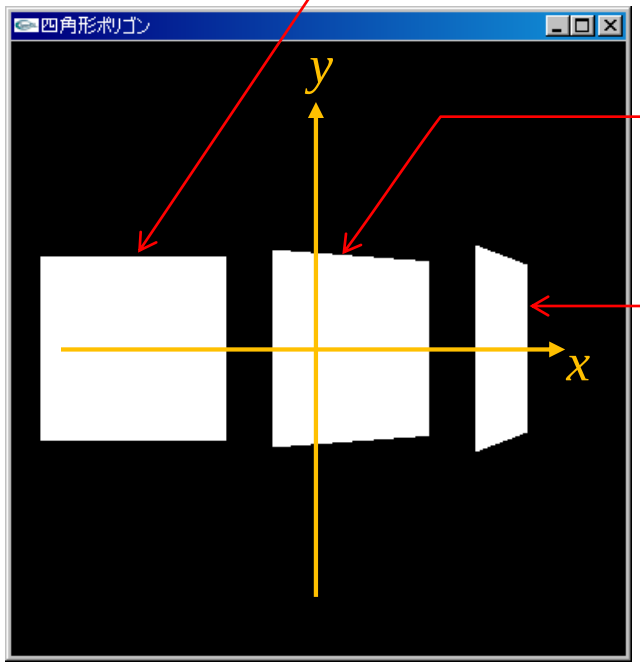


OpenGLでのポリゴンの描画(2)

Example12-2

辺の長さがsizeの正方形ポリゴン

```
void Squre(double size)
{
    glPushMatrix();
    glScalef(size/2.0, size/2.0, size/2.0);
    glBegin(GL_POLYGON);
        glVertex3f(-1, -1, 0);
        glVertex3f(+1, -1, 0);
        glVertex3f(+1, +1, 0);
        glVertex3f(-1, +1, 0);
    glEnd();
    glPopMatrix();
}
```

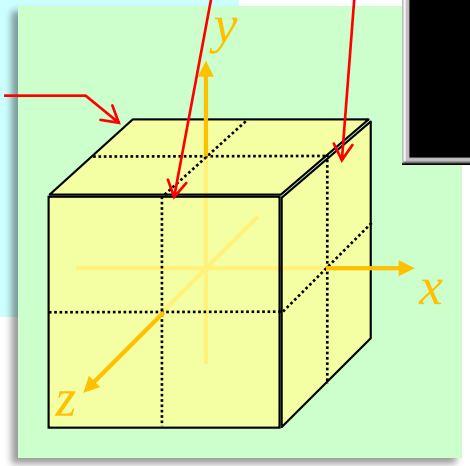


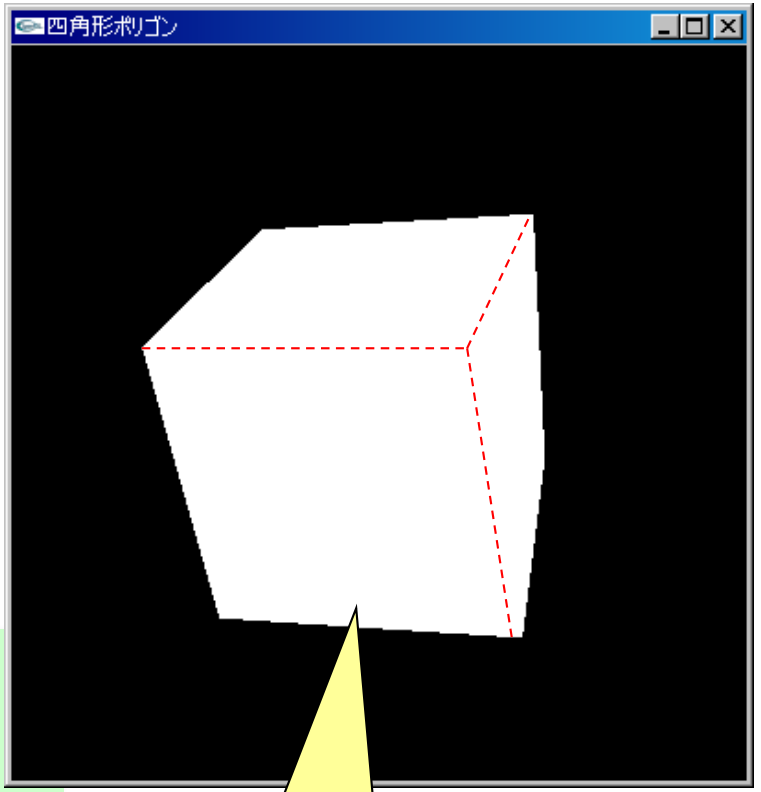

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    glPushMatrix();
    glTranslatef(-1.5, 0.0, 0.0);
    Squre(1.5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(+0.3, 0.0, 0.0);
    glRotatef(30, 0, 1, 0);
    Squre(1.5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(+1.5, 0.0, 0.0);
    glRotatef(60, 0, 1, 0);
    Squre(1.5);
    glPopMatrix();

    glFlush();
}
```

立方体の描画

Example12-3

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(-15, 0, 1, 0);
    glRotatef(30, 1, 0, 0);

    glPushMatrix(); // 前面
    glTranslatef(0.0, 0.0, +1.0);
    Squre(2.0);
    glPopMatrix();

    glPushMatrix(); // 右側面
    glTranslatef(1.0, 0.0, 0.0);
    glRotatef(90, 0, 1, 0);
    Squre(2.0);
    glPopMatrix();

    glPushMatrix(); // 上面
    glTranslatef(0.0, 1.0, 0.0);
    glRotatef(-90, 1, 0, 0);
    Squre(2.0);
    glPopMatrix();

    glFlush();
}
```



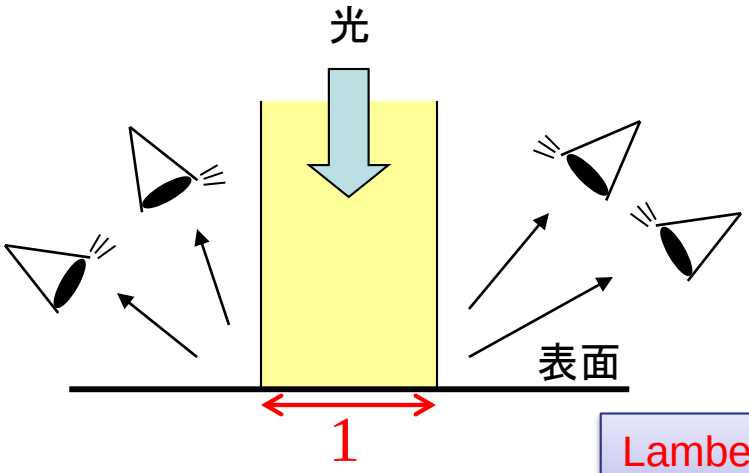


陰影が無いと立体に
見えない！

シェーディング

シェーディング　＝　陰影付け
　物体を照明している光に応じてポリゴン面の明るさを変えること

拡散反射モデル
- 見る方向によって面の明るさは変わらない.
- 面に光が照射される角度によって面の明るさは変わる.

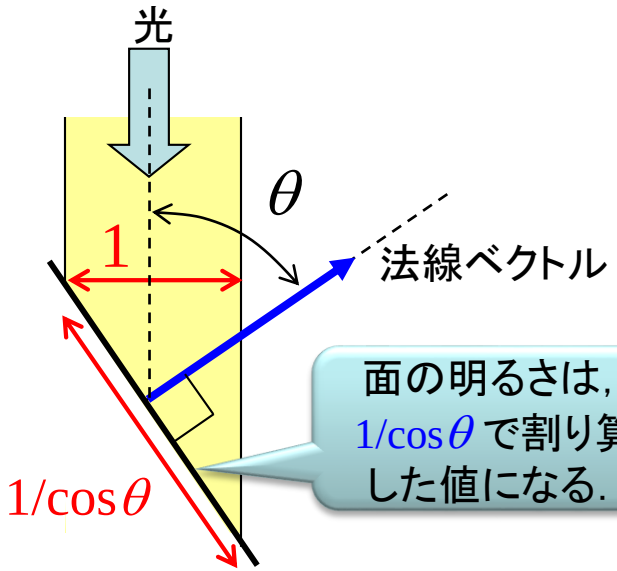


Lambertの余弦則
(Lambert反射)

面の明るさ　$I_d = I_0 \cos\theta$

I_0　:元の光の強さ

θ　:光線が面の法線と為す角度

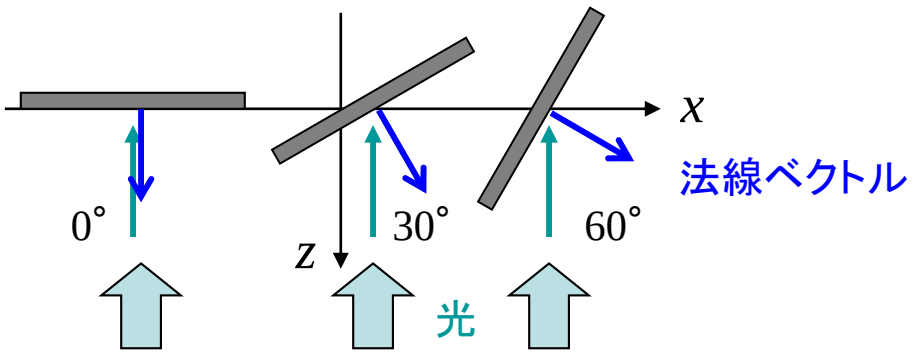


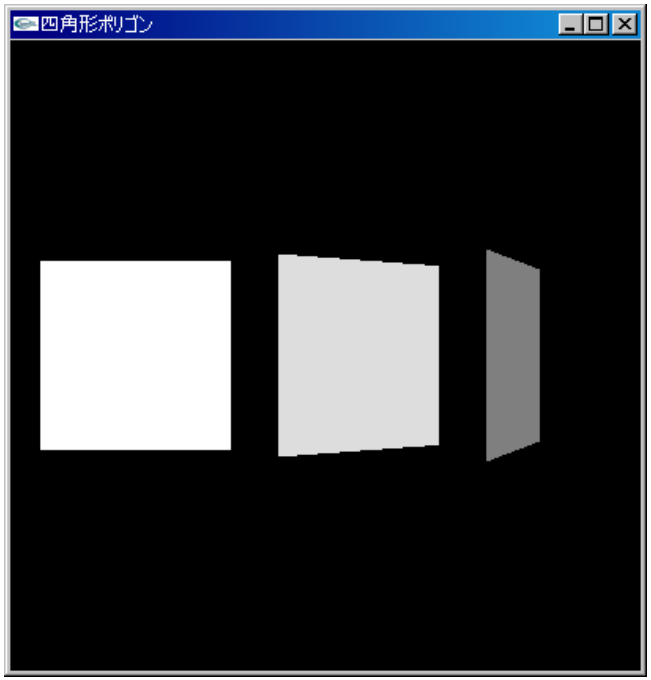
レンダリング
＝　シェーディング等を行い，その物体を実際に見た場合に近い画像にすること

拡散反射モデルによるフラットシェーディングの例

Example12-4

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    double bw;

    glPushMatrix();
    glTranslatef(-1.5, 0.0, 0.0);
    bw = 1.0;
    glColor3f(bw, bw, bw);
    Squre(1.5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(+0.3, 0.0, 0.0);
    glRotatef(30, 0, 1, 0);
    bw = cos(30.0/180.0*3.1415);
    glColor3f(bw, bw, bw);
    Squre(1.5);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(+1.5, 0.0, 0.0);
    glRotatef(60, 0, 1, 0);
    bw = cos(60.0/180.0*3.1415);
    glColor3f(bw, bw, bw);
    Squre(1.5);
    glPopMatrix();

    glFlush();
}
```





フラットシェーディング
＝面全体が均一な明るさ(I_d)になるシェーディング法

立方体のシェーディング

Example12-5

```
void Squre(double size)
{
    glPushMatrix();
    glScalef(size/2.0, size/2.0, size/2.0);
    glBegin(GL_POLYGON);
        glNormal3f(0.0, 0.0, 1.0);
        glVertex3f(-1, -1, 0);
        glVertex3f(+1, -1, 0);
        glVertex3f(+1, +1, 0);
        glVertex3f(-1, +1, 0);
    glEnd();
    glPopMatrix();
}
```

面の法線ベクトルを指定

ローカル座標で$(x, y, 0)$平面上の正方形

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("四角形ポリゴン");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 0.0, 10.0);

    glShadeModel(GL_FLAT);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);

    glutDisplayFunc(display);
    glutMainLoop();
}
```

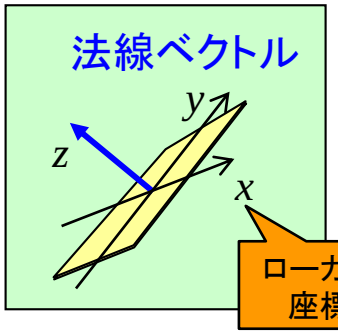
フラットシェーディングを指定

ライト0番をオン

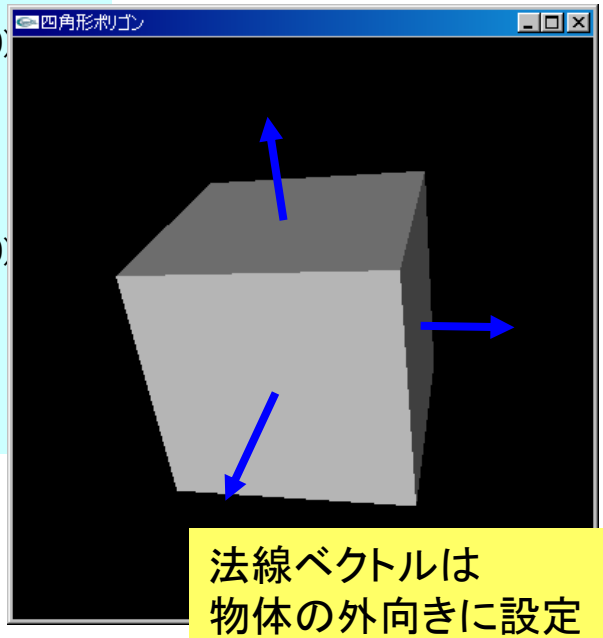
シェーディング処理をオン

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(-15, 0, 1, 0);
    glRotatef(30, 1, 0, 0);

    glPushMatrix(); // 前面
    glTranslatef(0.0, 0.0, +1.0);
    Squre(2.0);
    glPopMatrix();

    glPushMatrix(); // 側面
    glTranslatef(1.0, 0.0, 0.0);
    glRotatef(90, 0, 1, 0);
    Squre(2.0);
    glPopMatrix();

    glPushMatrix(); // 上面
    glTranslatef(0.0, 1.0, 0.0);
    glRotatef(-90, 1, 0, 0);
    Squre(2.0);
    glPopMatrix();
    glFlush();
}
```

ペイントハンドラは Example12-3 から変更無し



法線ベクトルは頂点と同じ幾何変換行列により変換される



法線ベクトルは物体の外向きに設定

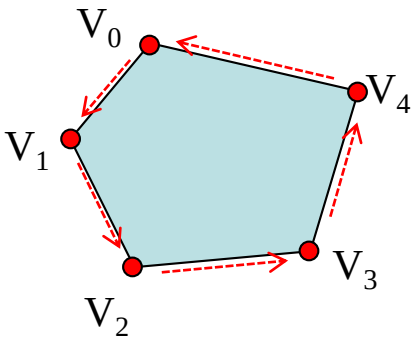
ポリゴン頂点の様々な指定方法

正方形を描画する例

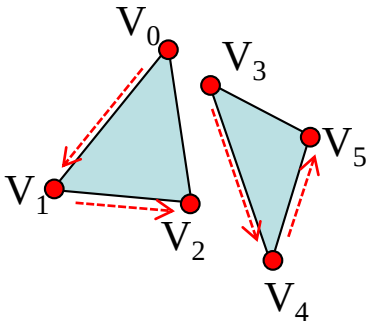
```
glBegin(GL_POLYGON);
    glVertex3f(-1, -1, 0);
    glVertex3f(+1, -1, 0);
    glVertex3f(+1, +1, 0);
    glVertex3f(-1, +1, 0);
glEnd();
```

頂点座標

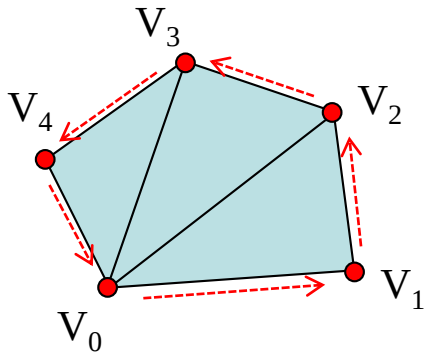
GL_POLYGON



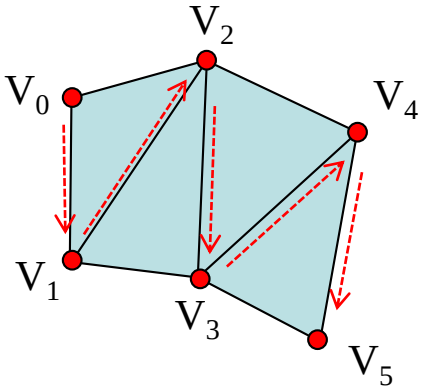
GL_TRIANGLES



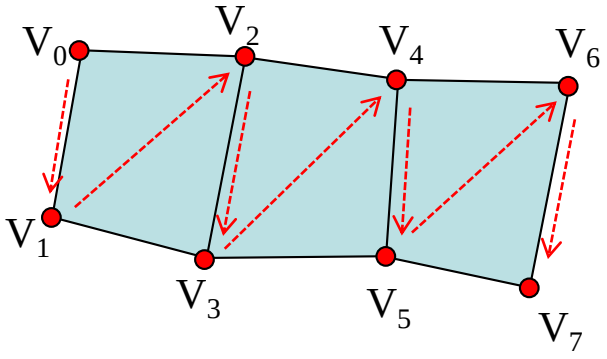
GL_TRIANGLE_FAN



GL_TRIANGLE_STRIP



GL_QUAD_STRIP



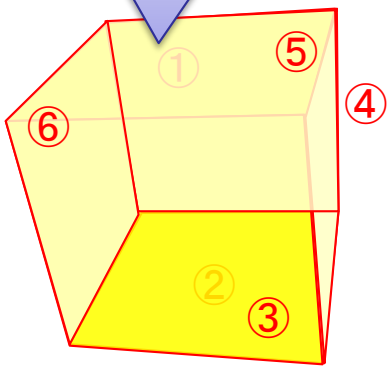
完全な立方体?

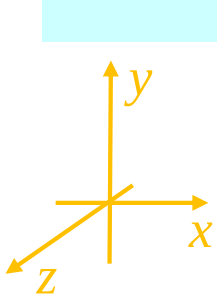
Example12-6

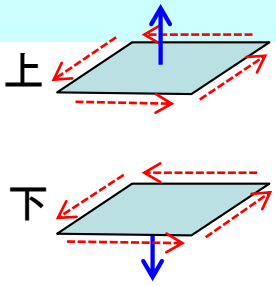
```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(-15, 0, 1, 0);
    glRotatef(30, 1, 0, 0);

    glBegin(GL_POLYGON);     //上面①
        glNormal3f(0, +1, 0);
        glVertex3f(-1, +1, +1);
        glVertex3f(+1, +1, +1);
        glVertex3f(+1, +1, -1);
        glVertex3f(-1, +1, -1);
    glEnd();

    glBegin(GL_POLYGON);     //下面②
        glNormal3f(0, -1, 0);
        glVertex3f(-1, -1, +1);
        glVertex3f(+1, -1, +1);
        glVertex3f(+1, -1, -1);
        glVertex3f(-1, -1, -1);
    glEnd();
```

```
    glBegin(GL_QUAD_STRIP);
        glNormal3f(0, 0, 1);  //前面③
        glVertex3f(-1, +1, +1);
        glVertex3f(-1, -1, +1);
        glVertex3f(+1, +1, +1);
        glVertex3f(+1, -1, +1);

        glNormal3f(+1, 0, 0); //右側面④
        glVertex3f(+1, +1, -1);
        glVertex3f(+1, -1, -1);

        glNormal3f(0, 0, -1); //後面⑤
        glVertex3f(-1, +1, -1);
        glVertex3f(-1, -1, -1);

        glNormal3f(-1, 0, 0); //左側面⑥
        glVertex3f(-1, +1, +1);
        glVertex3f(-1, -1, +1);
    glEnd();

    glFlush();
}
```

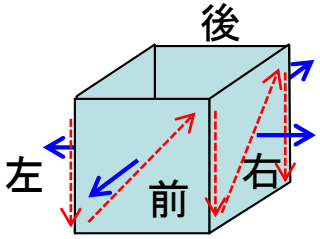
⑤と⑥は光が当たらないため真っ黒



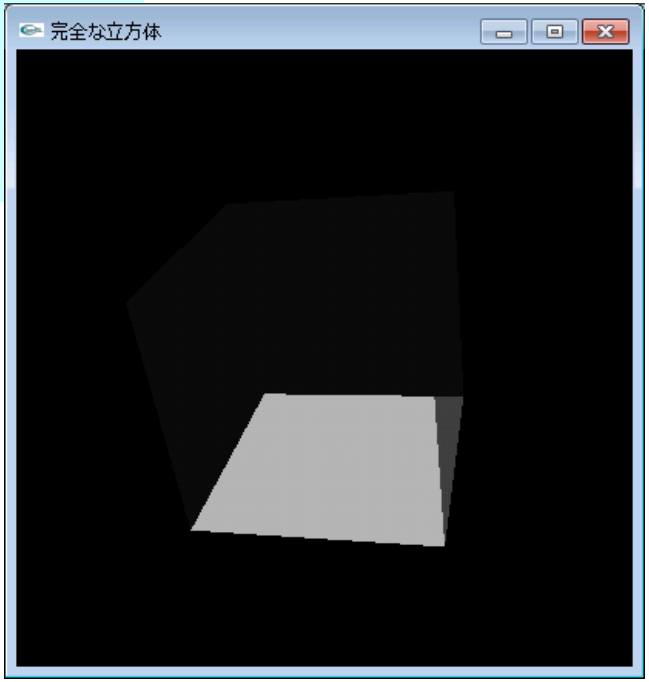




GL_POLYGON



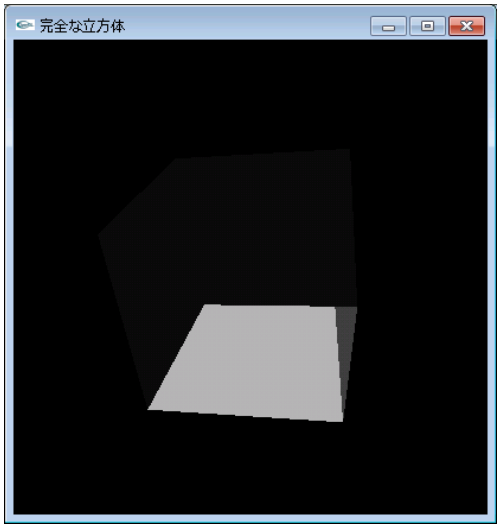
GL_QUAD_STRIP

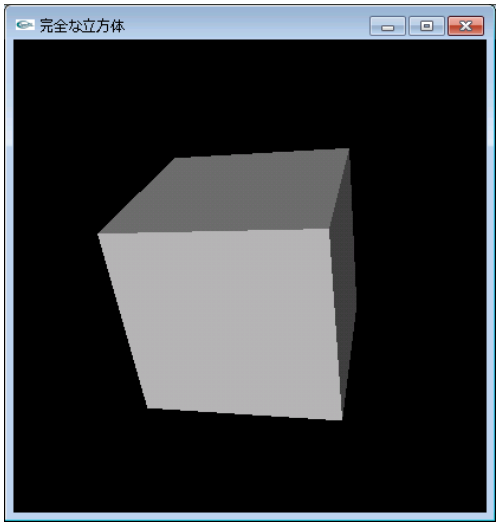


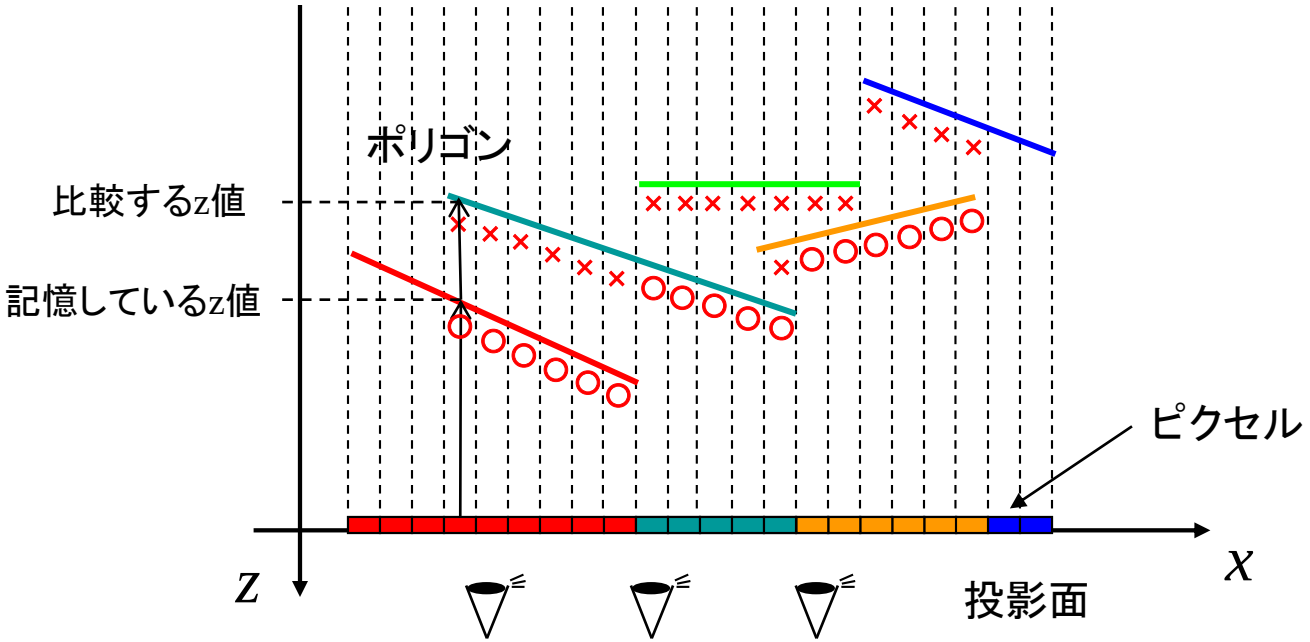
z-バッファ法による隠面消去

隠面消去
後ろ側に隠れていて見えない部分を消す（描かない）処理

z-バッファ法
2次元画像の各ピクセル毎に描画するポリゴン面の奥行き位置(デプス, depth, z値)をz-バッファに記憶しておき，投影面にもっとも近いポリゴンをそのピクセルに描画する手法







OpenGLにおけるz-バッファ法の設定

Example12-7

```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(-15, 0, 1, 0);
    glRotatef(30, 1,
```

(3) z-バッファもクリアする

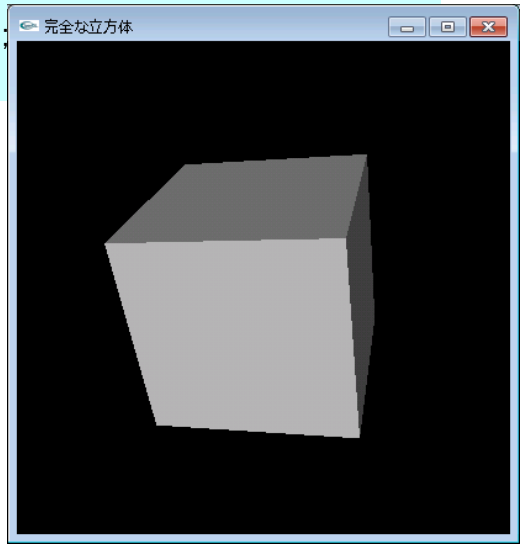
```
    glBegin(GL_POLYG
        glNormal3f(0
        glVertex3f(-
        glVertex3f(+
        glVertex3f(+
        glVertex3f(-
    glEnd();

    glBegin(GL_POLYG
        glNormal3f(0
        glVertex3f(-
        glVertex3f(+
        glVertex3f(+
        glVertex3f(-
    glEnd();
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("完全な立方体");

    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 10.0);

    glShadeModel(GL_FLAT);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutMainLoop();
}
```

(1) z-バッファの準備

(2) z-バッファ法による隠面消去を有効化する

```
    glBegin(GL_QUAD_STRIP);     //前面
        glNormal3f(0, 0, 1);
        glVertex3f(-1, +1, +1);
        glVertex3f(-1, -1, +1);
        glVertex3f(+1, +1, +1);
        glVertex3f(+1, -1, +1);

        glNormal3f(+1, 0, 0);   //右側面
        glVertex3f(+1, +1, -1);
        glVertex3f(+1, -1, -1);

mal3f(0, 0, -1);   //裏面
tex3f(-1, +1, -1);
tex3f(-1, -1, -1);

mal3f(-1, 0, 0);   //左側面
tex3f(-1, +1, +1);
tex3f(-1, -1, +1);
```



基本課題12

ウィンドウの名前を各自の学籍番号と氏名にすること

底面の直径が2で高さが2，中心の座標が(0, 0, –8)のサーフェースモデルの十二角柱を描画するペイントハンドラを作成し，下記のmain()関数で実行しなさい．

Report12-1

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 20.0);
    gluLookAt(4, 4, 0,   0, 0, -8,   0, 1, 0);

    glShadeModel(GL_FLAT);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    float light_position[] = {5.0, 10.0, 2.0, 0.0};
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```

正しい陰影になっているかどうか十分に注意すること

ペイントハンドラ内でz-バッファをクリアするのを忘れないこと

Wordのレポートにソースプログラムと実行結果(glutウィンドウ)の画面コピーを貼りつけて提出

発展課題12

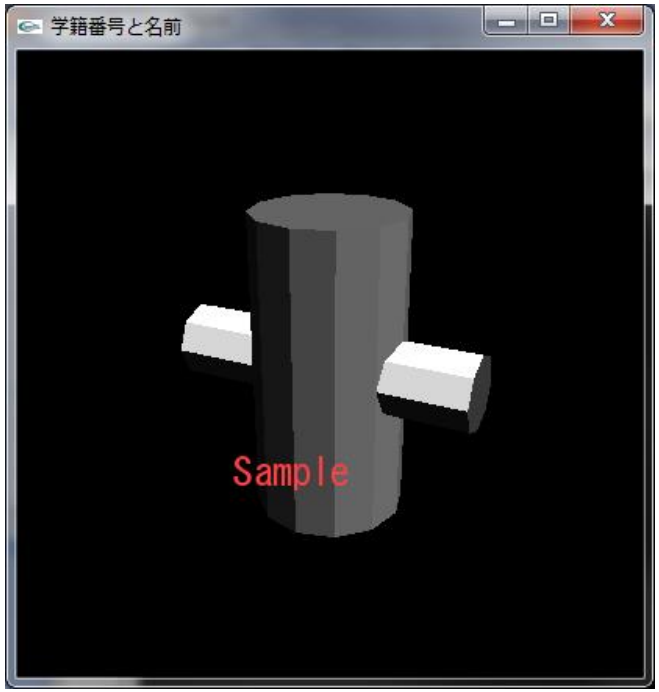
ウィンドウの名前を各自の学籍番号と氏名にすること

①底面の直径が2で高さが4，中心の座標が(0, 0, –8)のサーフェースモデルの十二角柱
②直径が1で高さが4，中心の座標が(0, 0, –8)のサーフェースモデルの八角柱
これら二つを組み合わせ，基本課題と同じmain()関数を用いて実行例と同じ結果を得なさい．

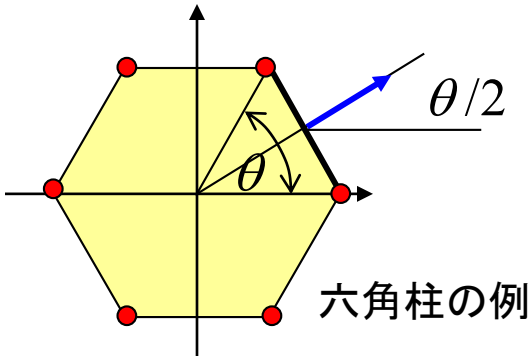
基本課題と発展課題の共通ヒント

下記のように，N角柱を描く関数を作成する

```
void Prism(int N)     // N角柱を描く関数
{
    //上面を描画 (GL_POLYGON)
    //下面を描画 (GL_POLYGON)
    //側面を描画 (GL_QUAD_STRIP)
}
```



このとき，側面の法線ベクトルは次のように考える



六角柱の例

法線ベクトルは必ず外向きになるように設定する

発展課題12では異なった幾何変換行列を設定してPrism()関数を2回呼び出す

Wordのレポートにソースプログラムと実行結果(glutウィンドウ)の画面コピーを貼りつけて提出