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
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**

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

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

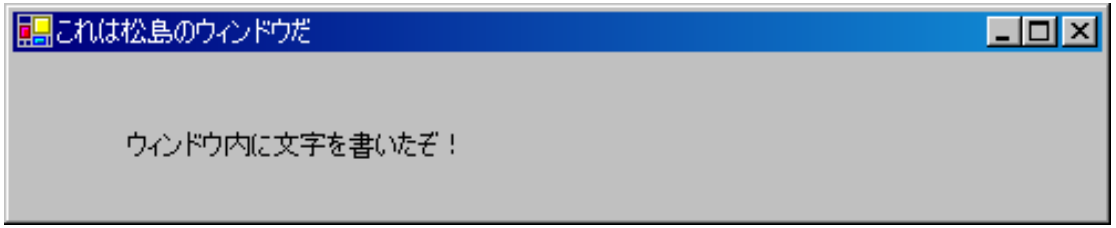
受講するためには
→「基礎プログラミング」程度のC言語の知識が必須

GUIプログラミングの基本＝イベント

復習

イベントとは

ウィンドウやウィンドウ内のパーツ(ボタン・スクロールバーなど)に発生する様々な**出来事**・きっかけ



イベントの例

- マウスがウィンドウ上に入った
- マウスがウィンドウ上から出た
- マウスのボタンが押された
- キーボードのキーが押された
- ウィンドウが閉じられた
- ウィンドウが開かれた
- ウィンドウのサイズが変わった
- 他のウィンドウの下になっていたウィンドウ全体が前に出た
- 他のウィンドウに隠れていた一部分が前に出た

このタイプのイベントが発生したらプログラムはウィンドウ内の文字や図形を描かなければならない → Paint イベント発生

Paintハンドラがイベント処理！

ウィンドウが隠れて出たり，サイズが変わったりするたびにウィンドウ内を描画するのか？
答え： YES！　それがウィンドウプログラミングの基本

GUIプログラムの実行方式＝イベント駆動(イベントドリブン)

復習

非GUIプログラム

```
int main(void)
{
    int x,i ;
    x = x + 10;
    printf(・・・);
    printf(・・・);
    for(i = 0; i < 10; i++)
    {
        ・・・・
        ・・・・
    }
    if (x > 20)
        ・・・・
    ・・・・
    ・・・・


}
```

予測可能な一定の順序で処理が進む

GUIプログラム

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    ・・・
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    ・・・
    glutDisplayFunc(display);
    glutMainLoop();
}
```

イベント発生のタイミングも順序も予測不能

イベントA発生

イベントA処理

プログラム終了

イベントループ

イベントB発生

イベントB処理

ウィンドウクローズイベント発生

イベントC発生

イベントC処理

イベントを処理する関数 → イベントハンドラ

display()関数は何か？　→正解ペイントイベントハンドラ

GUIではプログラマの仕事のほとんどはイベントハンドラの作成！

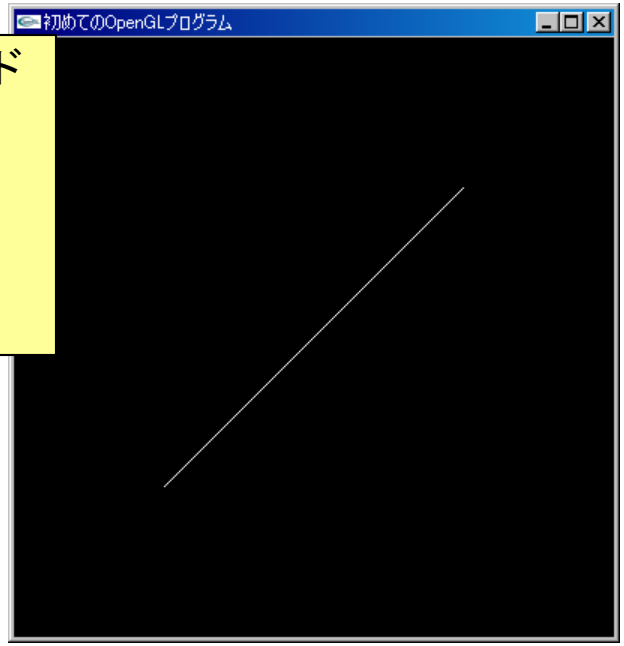
ペイントハンドラの登録

復習

Example9-1

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

void display(void)
{
    glClear( GL_COLOR_BUFFER_BIT );

    glColor3f(1.0, 1.0, 1.0);
    glBegin( GL_LINES );
        glVertex3f(-1.0, -1.0, 0.0);
        glVertex3f(+1.0, +1.0, 0.0);
    glEnd();

    glFlush();
}

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

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("初めてのOpenGLプログラム");
    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);
    glutMainLoop();
}
```

GLUTでのペイントハンドラ関数の形式
void func(void)
⇓
この形式の関数なら
どんな名前でも良い



ペイントハンドラとしてdisplay
という名前の関数を登録する.
関数ポインタ

アイドルハンドラとアニメーション

復習

Example10-2

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

void KeyboardHandler(unsigned char key, int x, int y);

double RotAngle = 0.0;      //グローバル(大域)変数

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);

    int N = 8;
    double angle = 2*3.1415/N;
    int i;
    glBegin( GL_LINE_LOOP );
        for (i = 0; i < N; i++)
            glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    glEnd();
    glBegin( GL_LINES );
        for (i = 0; i < N; i++)
        {
            glVertex3f(0.0, 1.0, 0.0);
            glVertex3f(cos(i*angle), -1.0, sin(i*angle));
        }
    glEnd();
    glFlush();
}
```

回転角度

変換行列を指定
- x軸周りで15度回転
- y軸周りでRotAngle度回転

アイドルハンドラ
- 回転角度を0.1度ずつ増加
- ウィンドウの再描画

```
void IncAngle(void)
{
    RotAngle = RotAngle + 0.1;
    if (RotAngle > 360.0)
        RotAngle = RotAngle - 360.0;
    glutPostRedisplay();
}

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

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("八角錐");
    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);
    glutIdleFunc(IncAngle);
    glutMainLoop();
}
```

ウィンドウの再描画

ペイントイベントの発生
↓
ペイントハンドラの呼び出し

Idleイベントは他にイベントが無い時に定期的に発生するイベント

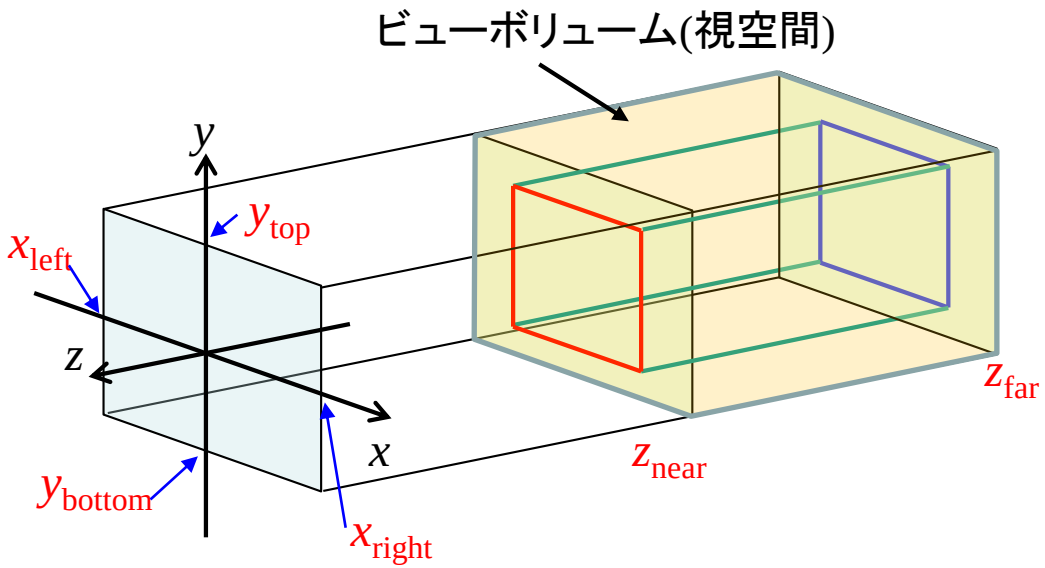
アイドルハンドラの登録
単語 idle = 遊んでいる. 暇な.

正投影(平行投影)とビューボリュームの設定

復習

平行投影を投影変換行列として設定する

```
glOrtho(xleft, xright, ybottom, ytop, znear, zfar)
```



平行投影
の投影変換行列

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

注意
ビューボリュームに
入っていない部分は
クリッピング処理により
表示されない

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

投影変換行列の設定開始

平行投影の行列を設定

透視投影とそのビューボリュームの設定

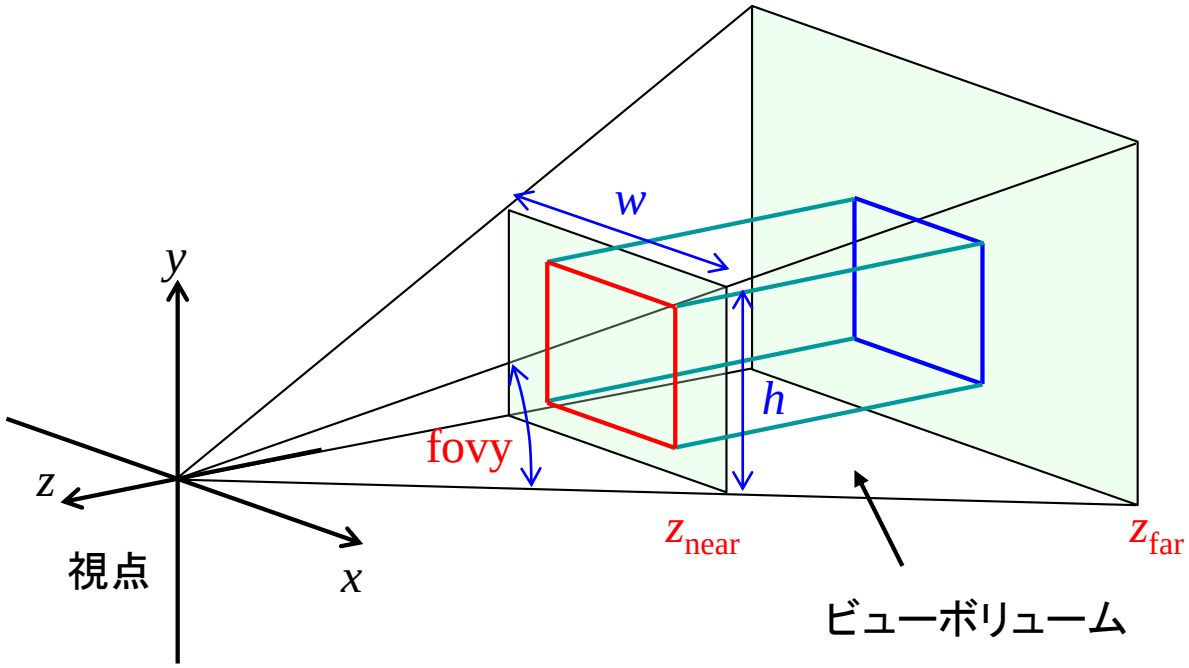
復習

透視投影を投影変換行列として設定する

```
gluPerspective(fovy, aspect, znear, zfar)
```

アスペクト比
aspect = w / h

注: znearとzfarは正値で設定する
(znearとzfarは原点からの距離を表す)



透視投影
の投影変換行列

$$\begin{bmatrix} \frac{1}{\tan\theta} & 0 & 0 & \cdots \\ 0 & \ddots & \ddots & \\ \vdots & & & \\ \vdots & & & \end{bmatrix}$$

複雑なので省略

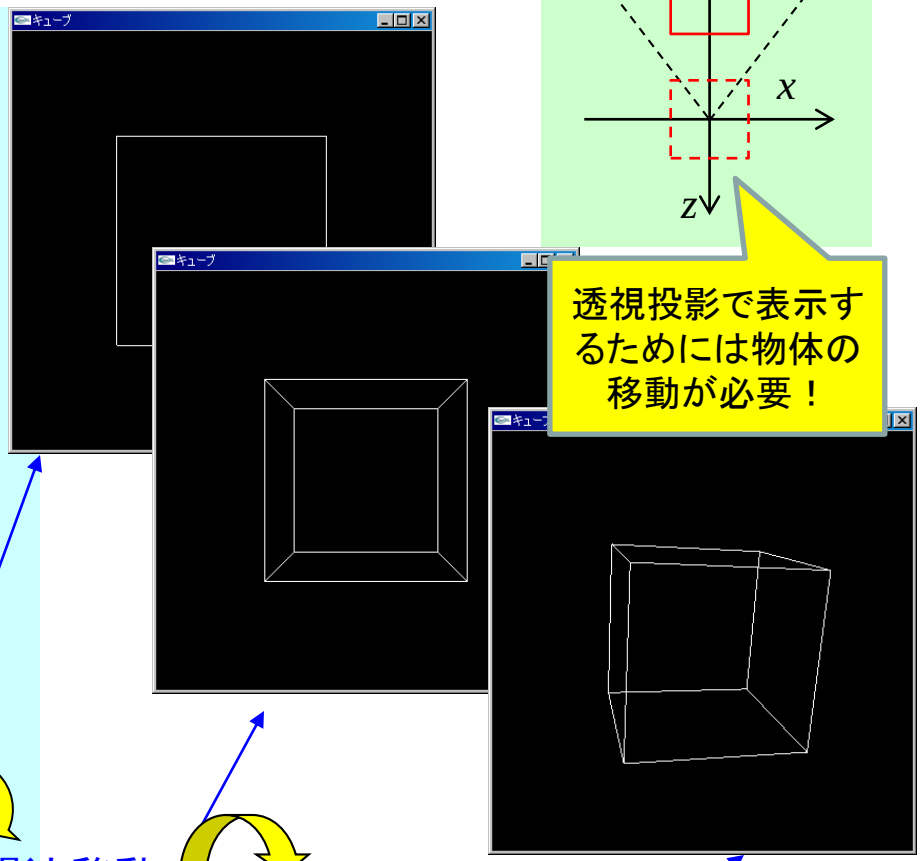
注意
ビューボリュームに入っていない部分はクリッピング処理により表示されない.

透視投影のプログラム

Example10-3

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

void KeyboardHandler(unsigned char key, int x, int y);

void display( void )
{
    // 辺長２の立方体を描く
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("キューブ");
    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);
    glutMainLoop();
}
```

平行投影



復習

透視投影で表示するためには物体の移動が必要！

透視投影と移動

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

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

透視投影と移動と回転

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

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

投影方法(投影変換行列)の設定

アフィン変換(幾何変換行列)の設定

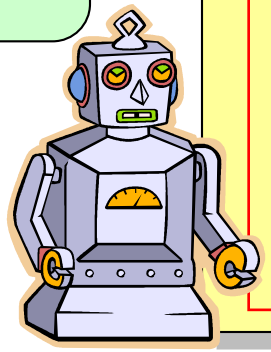
状態マシンとしてのOpenGL

復習

OpenGLは，現在の自分の状態に従って，頂点データを処理(描画)する

頂点データ
$v_1, v_2, v_3, \cdots$

OpenGL
状態
- 描く線の幅
- 描く線の色
- クリアで塗りつぶす色
- 幾何変換行列(GL_MODELVIEW) **M**
- 投影変換行列(GL_PROJECTION) **P**
- ・
- ・
- ・



$v'_1, v'_2, v'_3, \cdots$
で形成される
図形の描画

拡大・回転など頂点座標(物体モデル)の操作

3次元物体モデルをどのように2次元図形に変換するかを決定

状態変更コマンド
(関数)

OpenGLの現在の状態を変更する

幾何変換行列Mと投影変換行列Pが頂点をアフィン変換してから図形を描画する

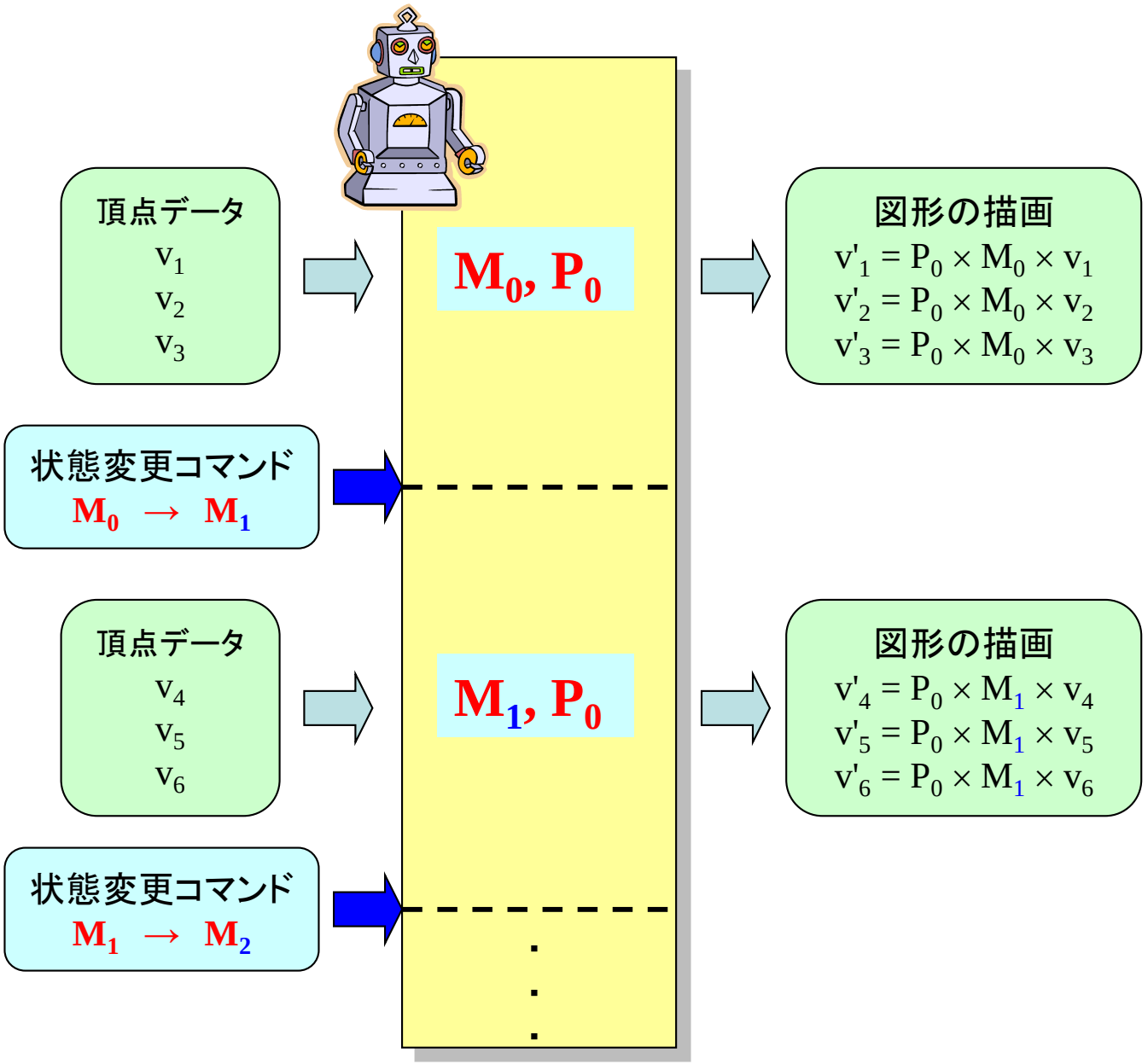
$$v' = P \times M \times v$$

$$M = M_1 M_2 M_3 \cdots$$

投影変換行列や幾何変換行列の設定関数は状態変更コマンド

OpenGL状態マシンの動作

復習



幾何変換行列の設定(状態変更)

復習

```
void glMatrixMode(GL_MODELVIEW)
```

幾何変換行列M(状態)の変更開始の指示

```
void glLoadIdentity(void)
```

Mとして単位行列 **I** を新たに設定する

単位行列に何を乗算しても同じ(行列をリセットする役割)

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
void glTranslatef(tx, ty, tz)
```

Mに, (tx, ty, tz)移動する変換行列 **T** を乗算する

```
void glSclalef(sx, sy, sz)
```

Mに, (sx, sy, sz)スケーリングする変換行列 **S** を乗算する

```
void glRotatef(angle, vx, vy, vz)
```

Mに, 回転軸ベクトル(vx, vy, vz)を中心に角度angle[度]回転する回転行列 **R** を乗算する

順序が問題！

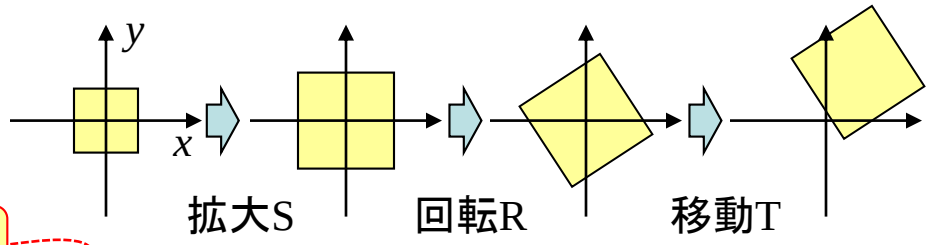
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();              // M=I
glTranslatef(2, 1, 0);         // M=IT
glRotatef(15, 0, 0, 1);        // M=ITR
glScalef(2, 2, 2);             // M=ITRS
```

$M = ITRS$

同じ

逆

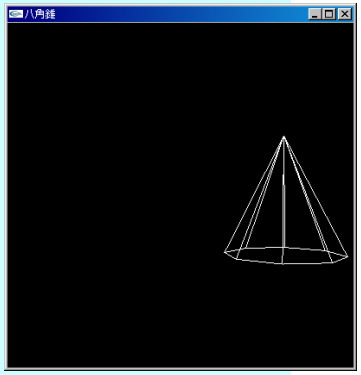
関数の実行順序と幾何変換操作の順序は逆になる

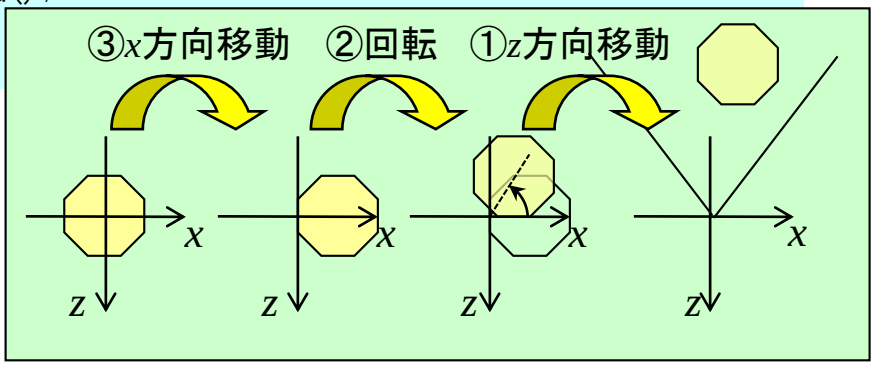


幾何変換と透視投影を用いたプログラ 復習

Example10-4

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

void KeyboardHandler(unsigned char key, int x, int y)
{
    if (key == ' ')
        exit(0);
}

void OctPyramid(void)
{
    int N = 8;
    double angle = 2*3.1415/N;
    int i;
    glBegin( GL_LINE_LOOP );
        for (i = 0; i < N; i++)
            glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    glEnd();
    glBegin( GL_LINES );
        for (i = 0; i < N; i++)
        {
            glVertex3f(0.0, 1.0, 0.0);
            glVertex3f(cos(i*angle), -1.0, sin(i*angle));
        }
    glEnd();
}
```






```
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);  //①
    glRotatef(30, 0.0, 1.0, 0.0);  //②
    glTranslatef(+2.0, 0.0, 0.0);  //③

    OctPyramid();

    glFlush();
}

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

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("八角錐");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

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

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

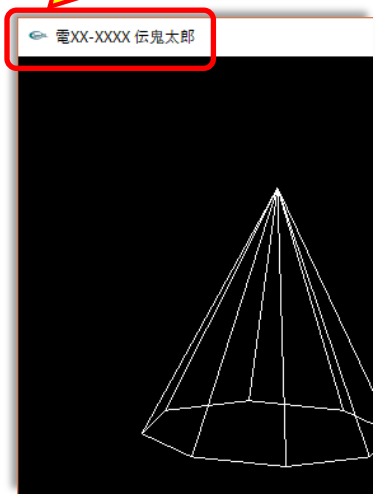
OpenGLの状態として幾何変換行列を登録

現在の状態で描画する頂点を登録

復習

課題10

ウィンドウの名前を各自の学籍番号と氏名にすること



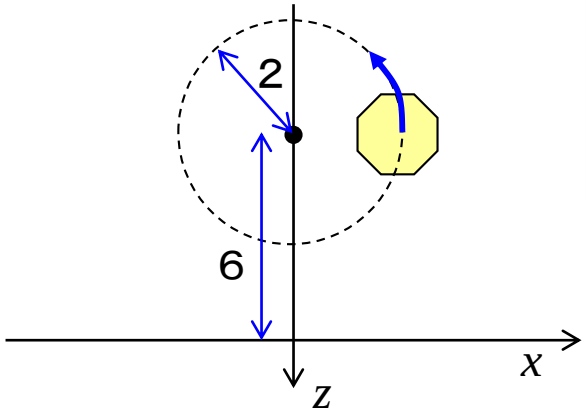
基本課題10

底面の直径が2で，高さが2の八角錐があり，その底面は$(x, -1, z)$平面にある．この八角錐の底面の中心が，$(0, -1, -6)$を中心とする$(x, -1, z)$平面上の半径2の円周上を回転するアニメーションを透視変換で作成せよ．但し，透視変換は上下開き角(fovy) が45度で，アスペクト比を1とする．

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．

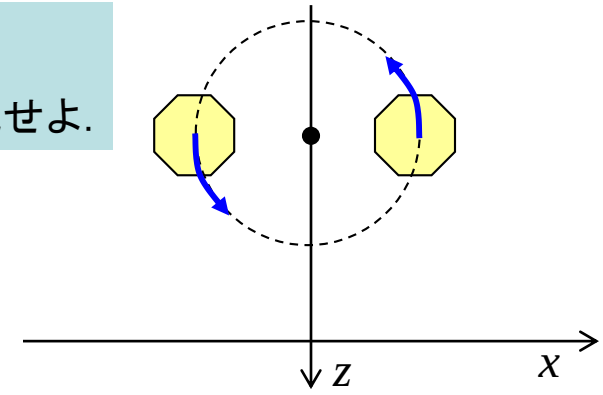
① Wordのレポート
- ソースプログラム
- glutウィンドウの画面コピー

② **実行プログラム(〇〇〇.exe)**



発展課題10

上の課題で八角錐の数を2個にしたアニメーションを作成せよ．



実行例

実行プログラム(Kadai10-1.exe, Kadai10-2.exe)をダウンロードして実行してみよ

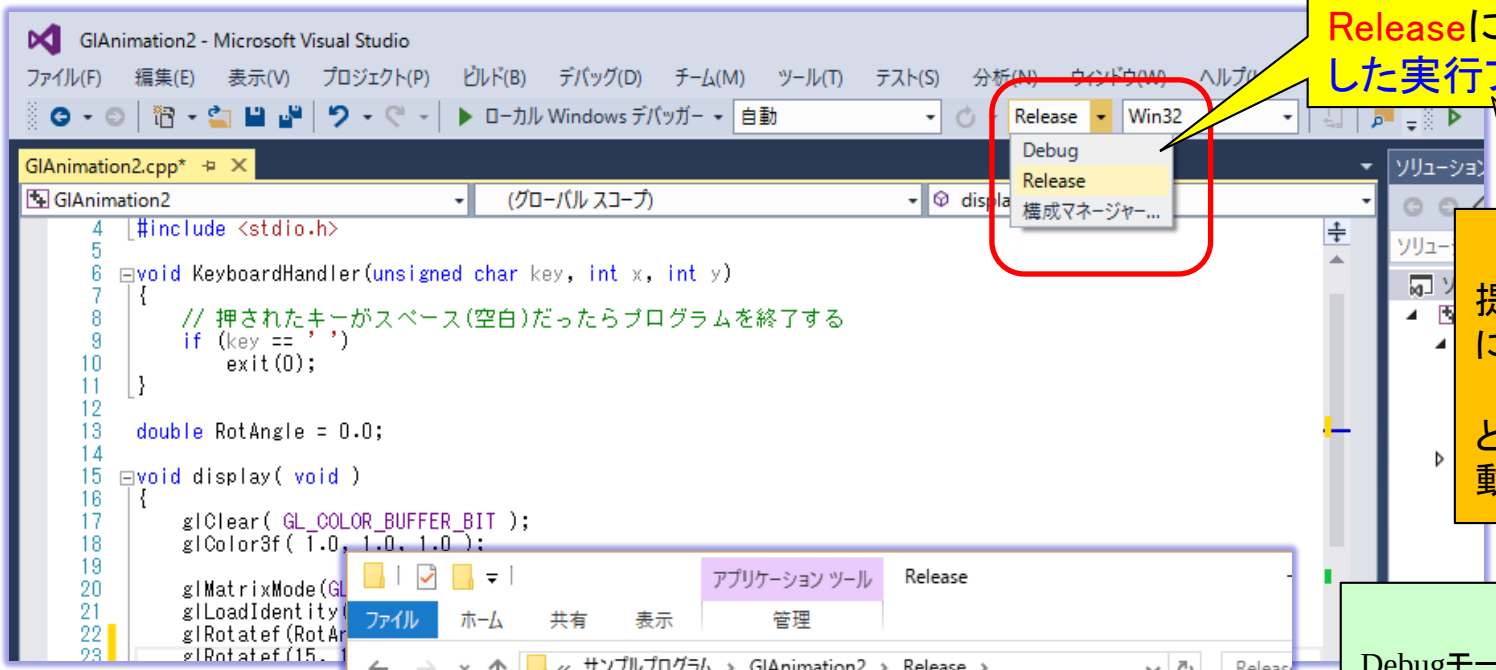
Wordに実行ファイルを添付することは禁止です!
(ウィルスとみなされることがあります) ⇒ 必ずzipファイルを利用

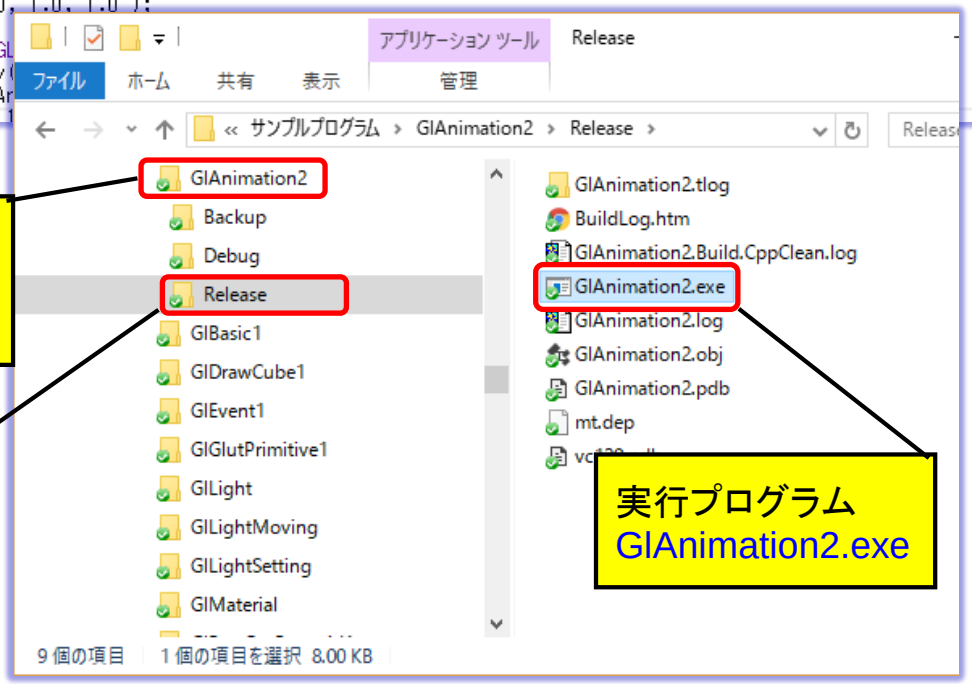
提出用の実行プログラム

復習

[ソリューション構成]を
Releaseに変更して，再ビルドした実行プログラムを提出．

【注意】
提出時以外はDebugにしておくこと．
ここがDebugでないと，デバッガが正常に動作しない．





プロジェクトフォルダ
プロジェクト名：
GlAnimation2

Releaseフォルダ

実行プログラム
GlAnimation2.exe

解説

Debugモードでは，ステップ実行などのために本来不要な部分が実行プログラム(exeファイル)に付け加えられてファイルサイズが増大し実行速度も低下する．

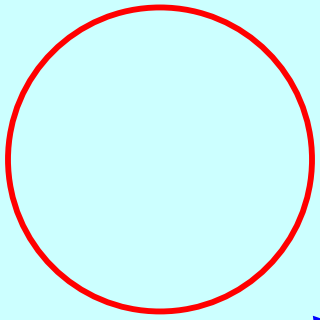
Releaseモードではそのような不要部分が取り除かれ，さらに最適化処理によって高速化し，ファイルサイズも小さくなる．そのため，他人に実行プログラムを渡す場合はReleaseにする方が良い．

ただし，Releaseモードではデバッガは正常に動作しない．

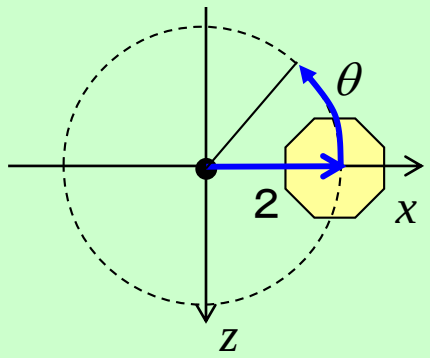
基本課題10 解答例

A君解答

```
void KeyboardHandler(unsigned char key, int x, int y);
void OctPyramid(void);

double RotAngle = 0.0;

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(RotAngle, 0.0, 1.0, 0.0);
    glTranslatef(+2.0, 0.0, 0.0);

    OctPyramid();
    glFlush();
}

void IncAngle(void)
{
    RotAngle = RotAngle + 0.01;
    if (RotAngle > 360.0)
        RotAngle = RotAngle - 360.0;
    glutPostRedisplay();
}
```



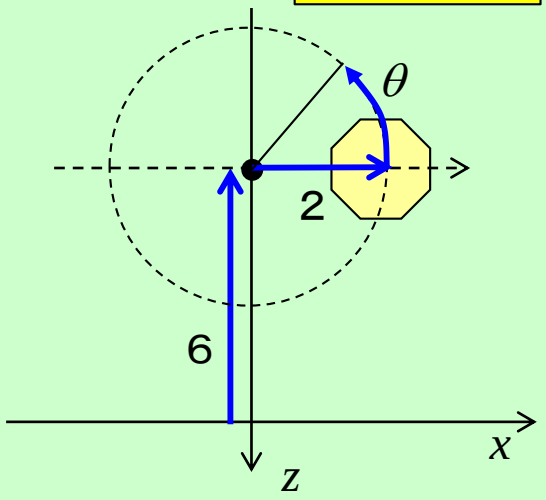
```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0
    glutInitWindowSize(400, 400
    glutInitDisplayMode(GLUT_RG
    glutCreateWindow("●●●●
    glClearColor ( 0.0, 0.0, 0.

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

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



(i) 図形を距離2だけ移動

(ii) 図形を角度θだけ回転

RotAngle

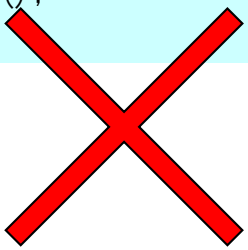


(iii) 全体を距離6だけ負のz軸方向に移動

基本課題10 特殊な解答例

Z君解答

```
double RotAngle = 0.0;

void IncAngle(void){
            //角度の変更
            RotAngle = RotAngle + 0.02;
            if (RotAngle > 360.0)
                        RotAngle = RotAngle - 360.0;
            OctPyramid();
            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);
            glRotatef(RotAngle, 0.0, 1.0, 0.0);
            glTranslatef(+2.0, 0.0, 0.0);
            OctPyramid();
            IncAngle();
            glFlush();
}
```



注意事項

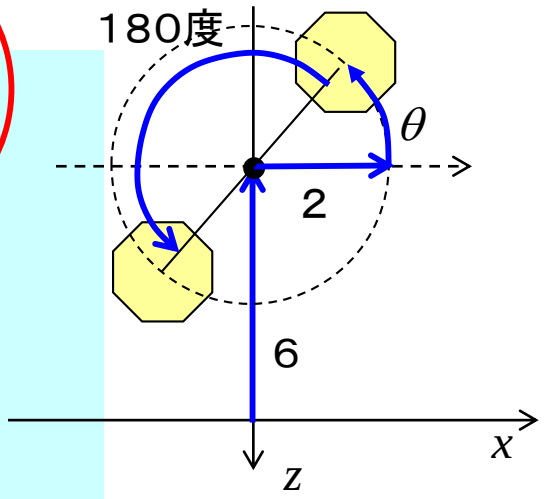
- ✓ 幾何変換(スケーリング・移動・回転)にはOpenGLの幾何変換行列を用いる
- ✓ Wordに実行プログラムを添付しない
- ✓ Releaseモードでビルドした実行プログラムを提出する.
- ✓ 実行プログラムとWordレポート以外をzipに入れない

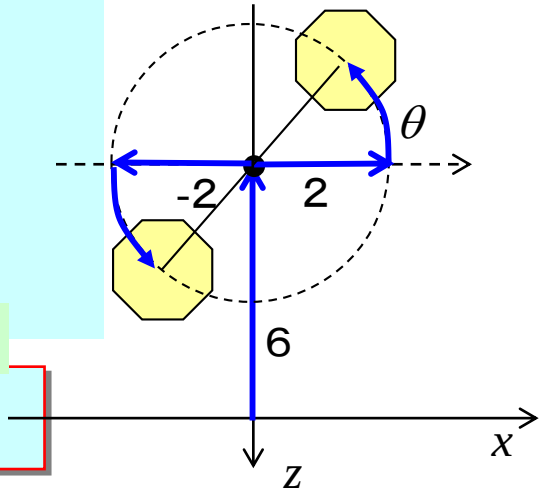
```
int main(int argc, char** argv){
            glutInit(&argc, argv);
            glutInitWindowPosition(0, 0);
            glutInitWindowSize(400, 400);
            glutInitDisplayMode(GLUT_RGBA);
            glutCreateWindow("ｘｘｘｘｘ");
            glClearColor ( 0.0, 0.0, 0.0, 1.0 );
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            gluPerspective(45, 1.0, 0.0, 10.0);
            glutDisplayFunc(display);
            glutKeyboardFunc(KeyboardHandler);
            glutMainLoop();
}
```

発展課題10 解答例

D君解答

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

    glMatrixMode(GL_MODELVIEW); //1つ目
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(RotAngle, 0.0, 1.0, 0.0);
    glTranslatef(2.0, 0.0, 0.0);
    OctPyramid();

    glMatrixMode(GL_MODELVIEW); //2つ目
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(RotAngle + 180, 0.0, 1.0, 0.0);
    glTranslatef(2.0, 0.0, 0.0);
    OctPyramid();

    glFlush();
}
```





別解

```
glRotatef(RotAngle, 0.0, 1.0, 0.0);
glTranslatef(-2.0, 0.0, 0.0);
```

発展課題10 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);
 glRotatef(30 , 0.0, 1.0 , 0.0);
 glRotatef(RotAngle , 0 , 1 , 0);
 glTranslatef(+2.0 , 0.0 , 0.0);
 double angle = 2*3.1415/8;
 int i;
  glBegin(GL_LINE_LOOP); //全点を結ぶ
  for(i=0;i<8;i++)
  {
    glVertex3f(cos(i*angle), -1.0 , sin(i*angle));
    glVertex3f(cos((i+1)*angle), -1.0 , sin((i+1)*angle));
    glVertex3f(0.0 , 1.0 ,0.0);
  }
  glEnd();

  glBegin(GL_LINE_LOOP);
  for(i=0;i<8;i++)
  {
    glVertex3f(cos(i*angle)-4, -1.0 , sin(i*angle));
    glVertex3f(cos((i+1)*angle)-4, -1.0 , sin((i+1)*angle));
    glVertex3f(-4.0 , 1.0 ,0.0);
  }
  glEnd();
  glFlush();
}
```

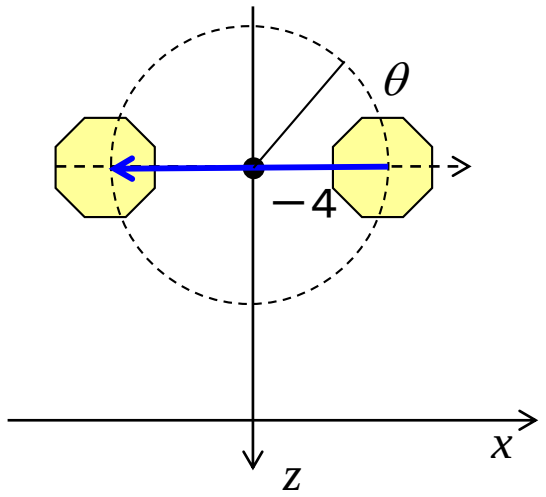
?

同じ処理のソースを繰り返し書かない！

幾何変換(スケーリング・移動・回転)には
OpenGLの幾何変換行列を用いる

B君解答

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

    glMatrixMode(GL_MODELVIEW);

    glLoadIdentity();    //現在の単位行列を設定
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(RotAngle, 0.0, 1.0, 0.0);
    glTranslatef(+2.0, 0.0, 0.0);
    OctPyramid();

    glTranslatef(-4.0, 0.0, 0.0);//既に+2されているのでスタート地点の座標を-2にするために4を引く
    OctPyramid();    //もうひとつの八角錐を描く

    glFlush();
}
```



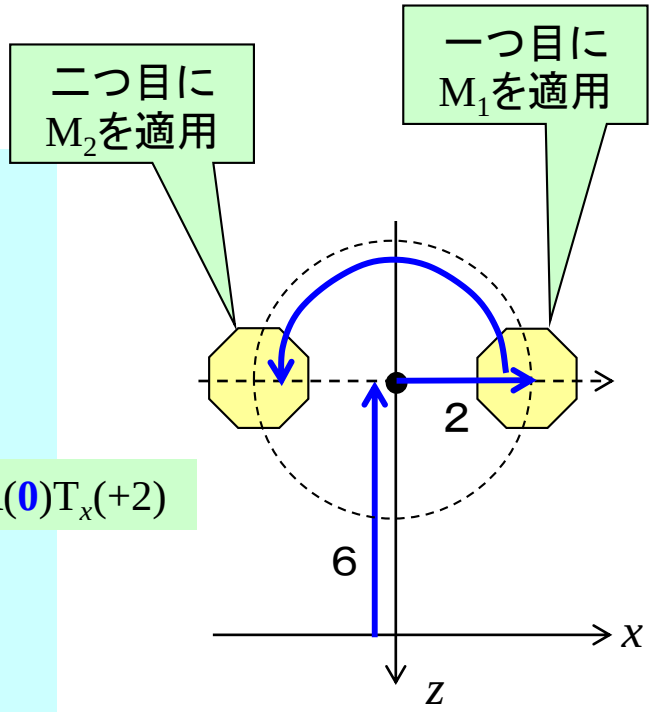
複数の八角錐を描く

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

    glMatrixMode(GL_MODELVIEW); //1つ目
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(RotAngle + 0, 0.0, 1.0, 0.0);
    glTranslatef(+2.0, 0.0, 0.0);
    OctPyramid();

    glMatrixMode(GL_MODELVIEW); //2つ目
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glRotatef(RotAngle + 180, 0.0, 1.0, 0.0);
    glTranslatef(+2.0, 0.0, 0.0);
    OctPyramid();

    glFlush();
}
```

共通処理

$M_1 = IT_z(-6)R(\mathbf{0})T_x(+2)$

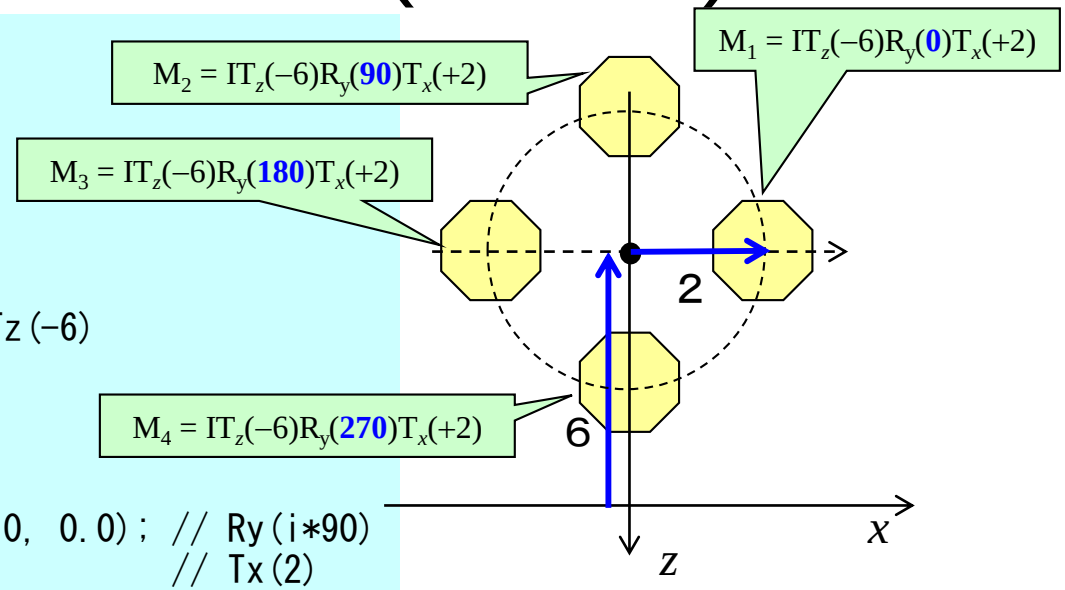
$M_2 = IT_z(-6)R(\mathbf{180})T_x(+2)$

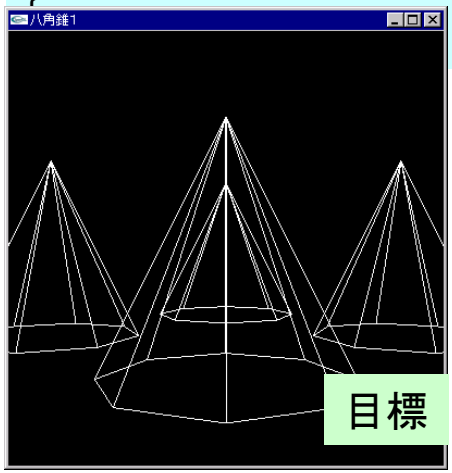


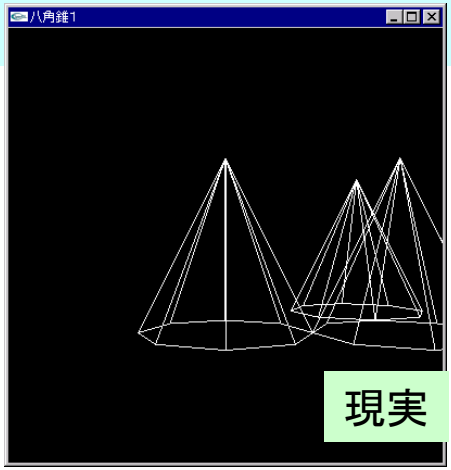
アニメーションしない場合は
"RotAngle +"
を削除すれば良いので，とりあえず考えないでおく

幾何変換行列の操作(失敗例)

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

共通処理
    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)

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

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

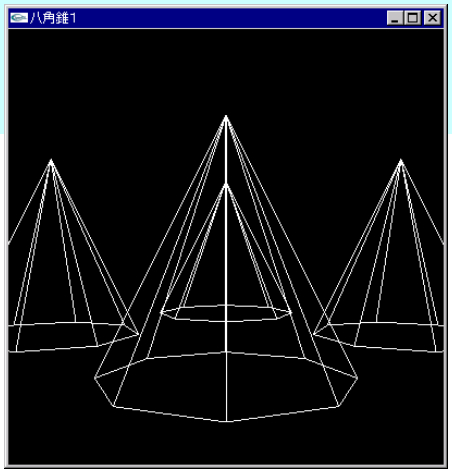
・・・

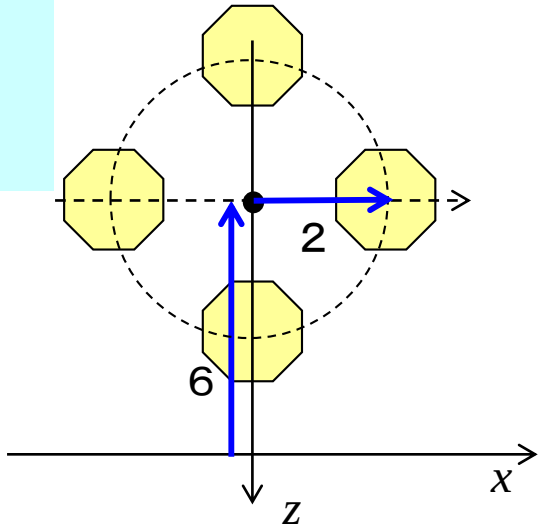
幾何変換行列の操作(成功例1)

Example11-1

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

    glMatrixMode(GL_MODELVIEW);

    int i;
    for (i = 0; i < 4; i++)
    {
        glLoadIdentity();
        glTranslatef(0.0, 0.0, -6.0);
        glRotatef(RotAngle + i*90, 0.0, 1.0, 0.0);
        glTranslatef(+2.0, 0.0, 0.0);
        OctPyramid();
    }
    glFlush();
}
```

$M = IT_z(-6)R_y(i\times 90)T_x(+2)$





幾何変換行列の操作(成功例2)

Example11-2

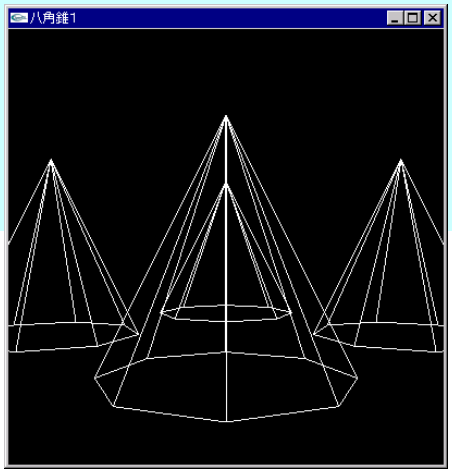
```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    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();
}
```

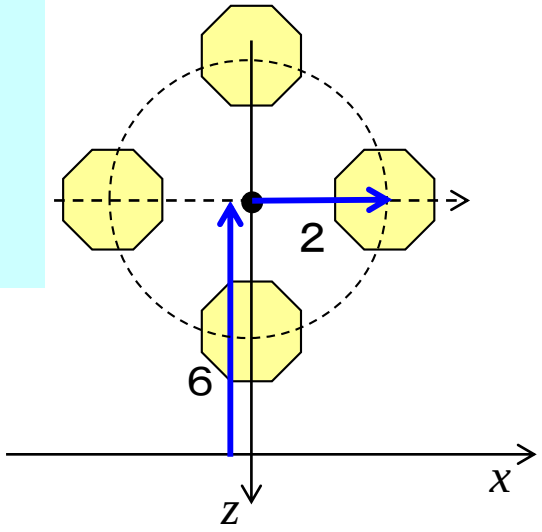
glPushMatrix()関数とglPopMatrix()関数をペアで用いる

より高度な変換操作のためにはこちらが望ましい

ここまでのMを記録する

記録しておいたMを呼び戻す



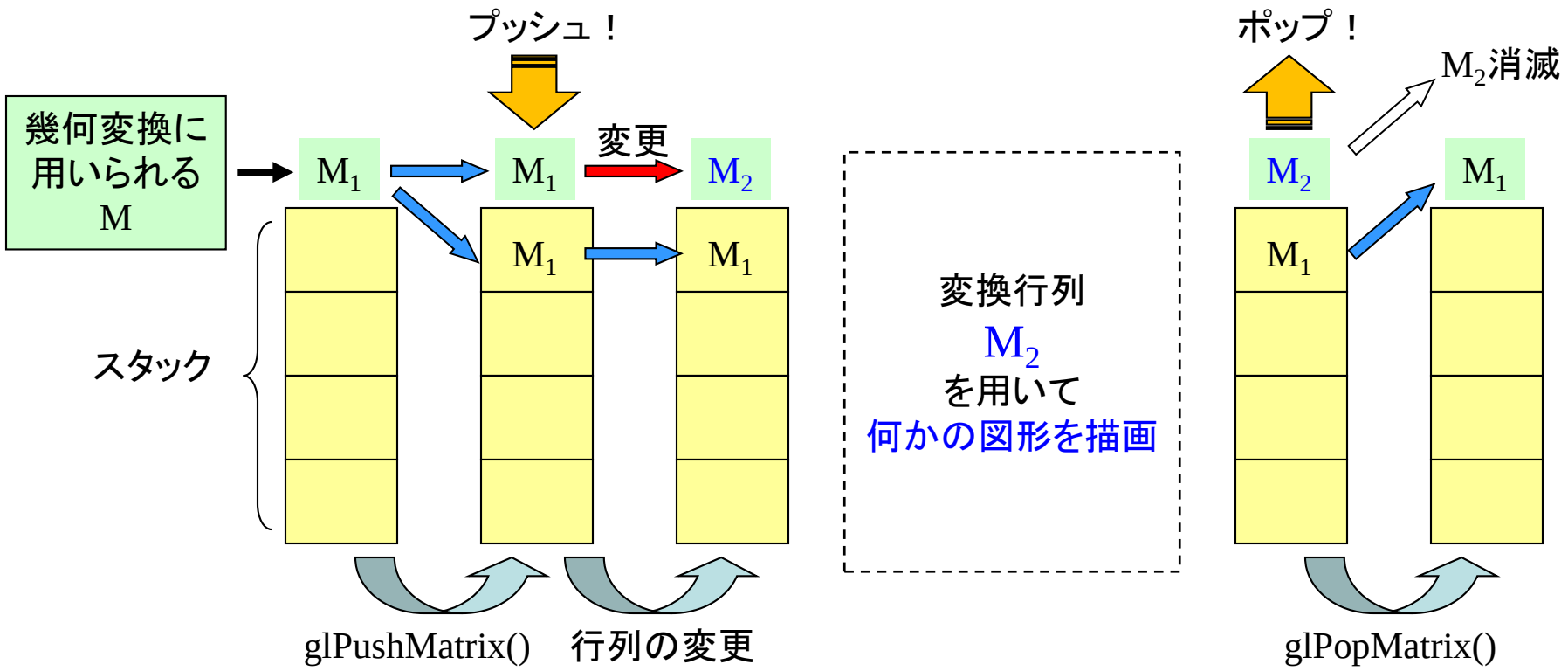


OpenGLにおける変換行列の変更

OpenGLには二つの変換行列がある.
P　透視変換行列
M　幾何変換行列

$$v' = P M v$$

glPushMatrix()とglPopMatrix()は，一時的に幾何変換行列を変更してもその影響が後に及ぼされないようにしている



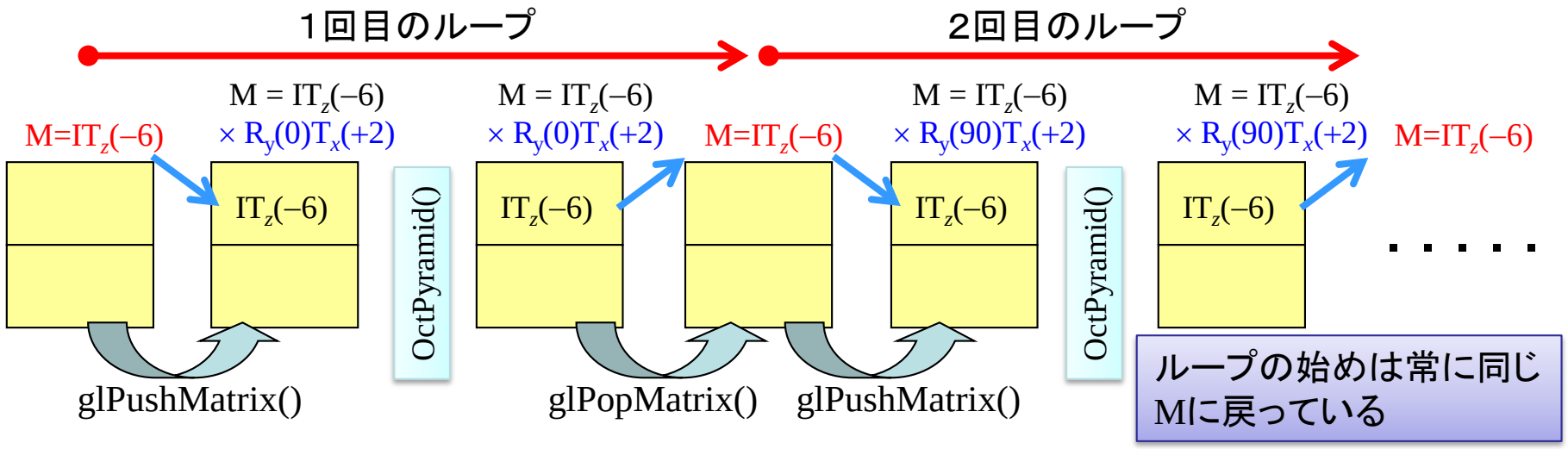
OpenGLにおけるローカル座標系の扱い(1)

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

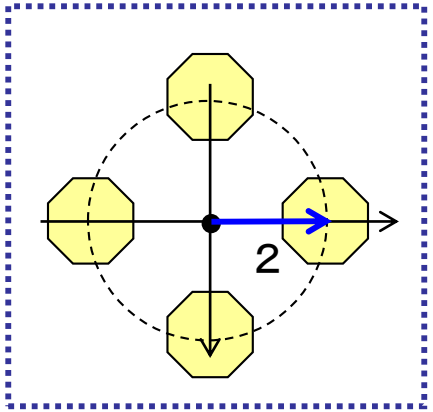
$\times R_y(i \times 90)T_x(+2)$ — 変換行列の変更部分

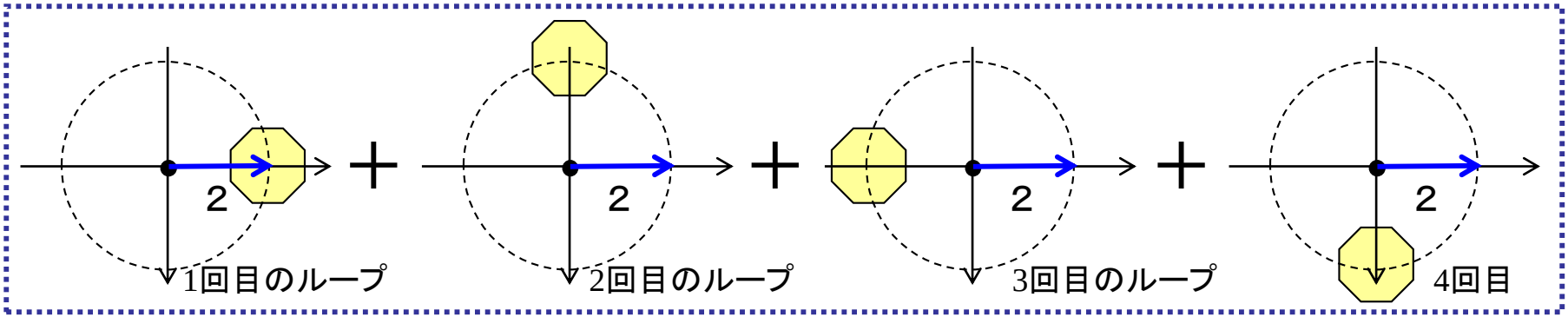


OpenGLにおけるローカル座標系の扱い(2)

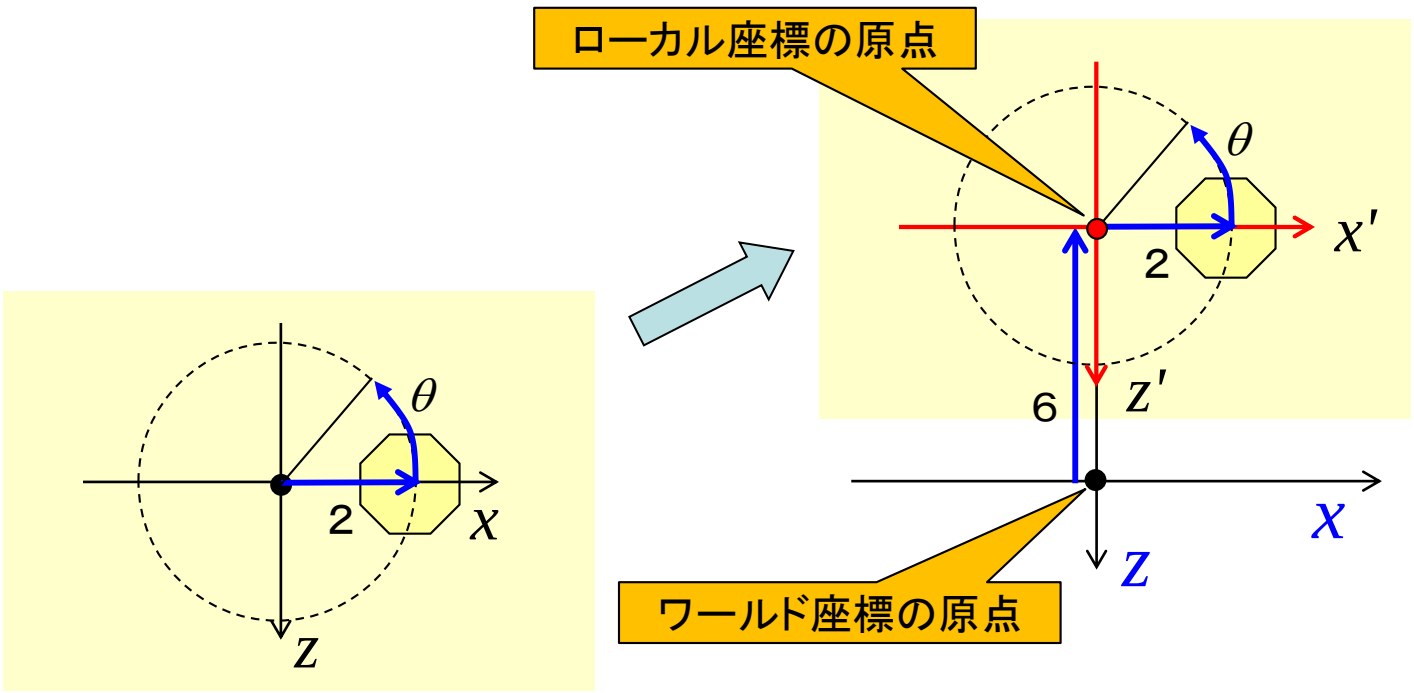
```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    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)

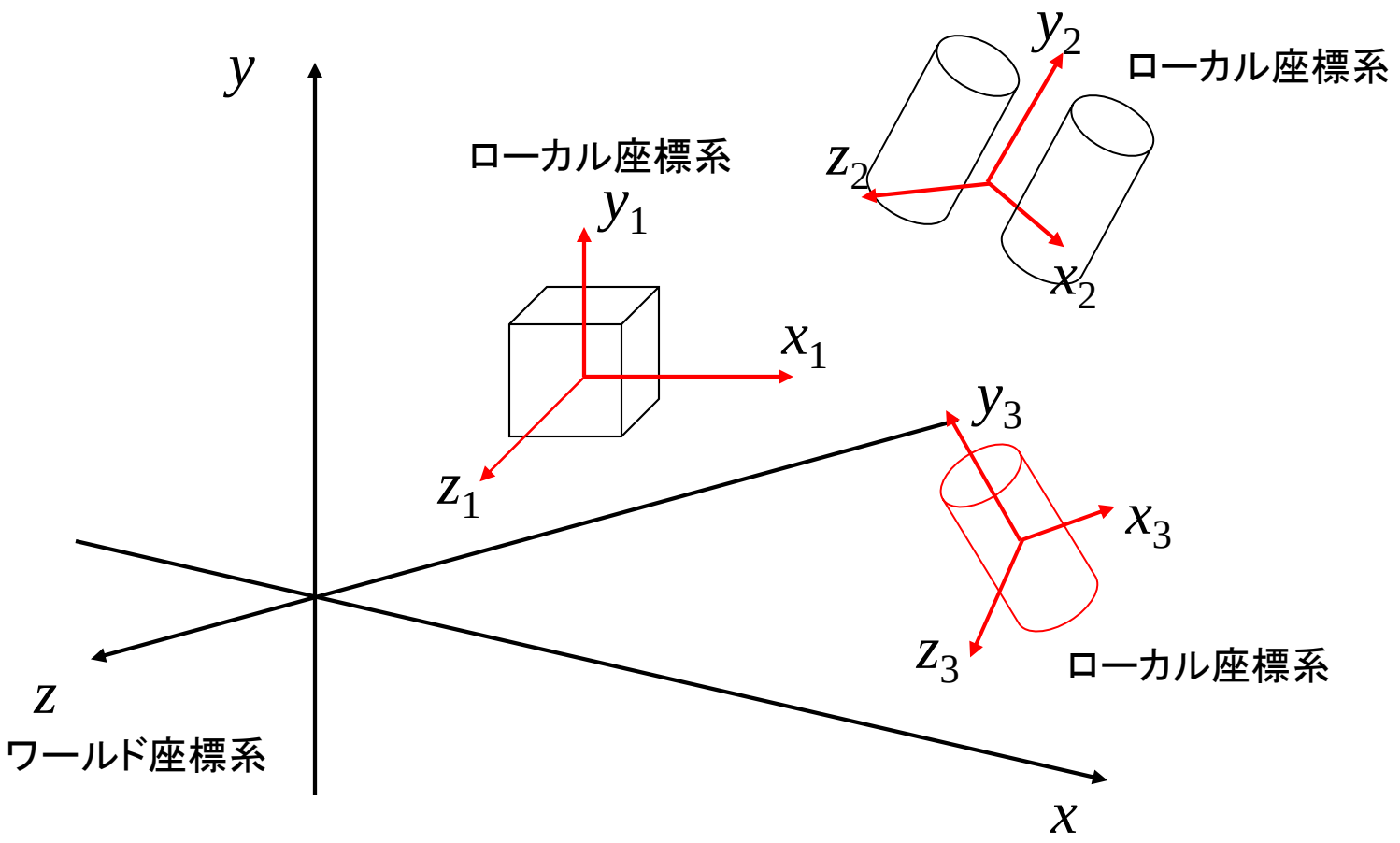


(i) 図形を距離2だけ移動
(ii) 図形を角度θだけ回転

(iii) 全体を距離6だけz軸の負の方向に移動

(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)



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

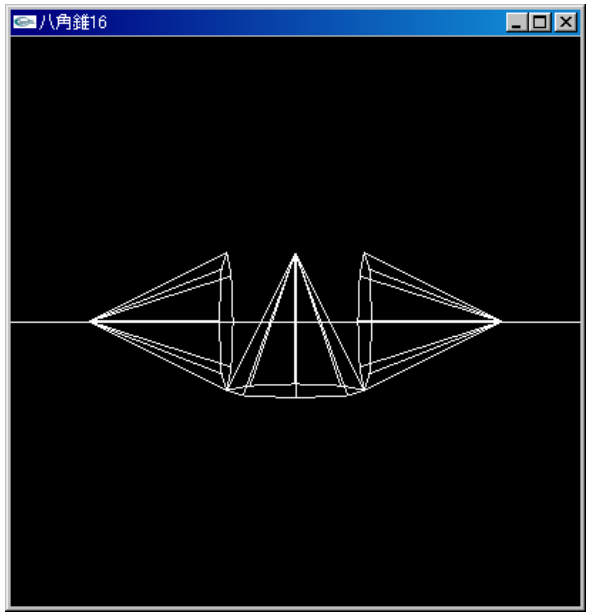
OpenGLにおけるローカル座標系の扱い(3)

Example11-4

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

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -10.0);

    glBegin(GL_LINES);              //見やすいように水平線を描く
        glVertex3f(-5.0, 0.0, 0.0);
        glVertex3f(+5.0, 0.0, 0.0);
    glEnd();

    OctPyramid();                   //普通に描く

    glPushMatrix();                 //z軸周りに-90度回転して描く
    glTranslatef(+2.0, 0.0, 0.0);
    glRotatef(-90, 0.0, 0.0, 1.0);
    OctPyramid();
    glPopMatrix();

    glPushMatrix();                 //z軸周りに+90度回転して描く
    glTranslatef(-2.0, 0.0, 0.0);
    glRotatef(+90, 0.0, 0.0, 1.0);
    OctPyramid();
    glPopMatrix();

    glFlush();
}
```

点線内のソースで描かれる図形全体をz方向に−10移動

glPushMatrix()とglPopMatrix()は，一時的に幾何変換行列を変更してもその影響が後に及ぼされないようにしている



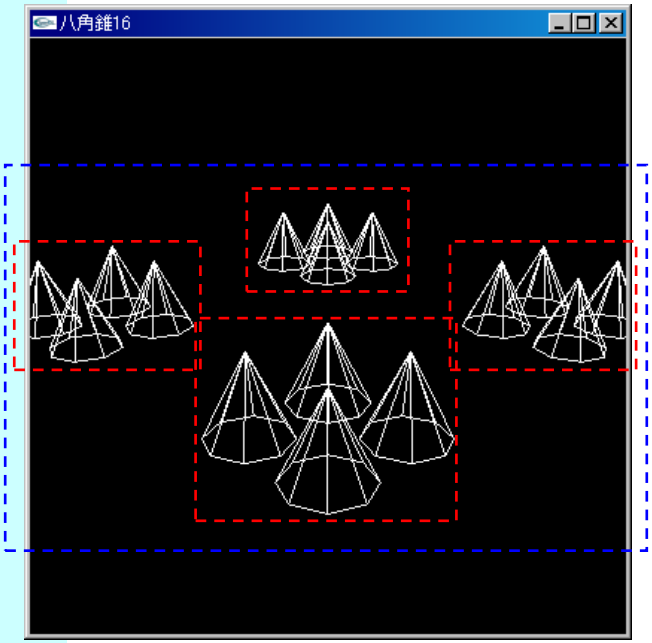
OpenGLにおけるローカル座標系の扱い(4)

Example11-5

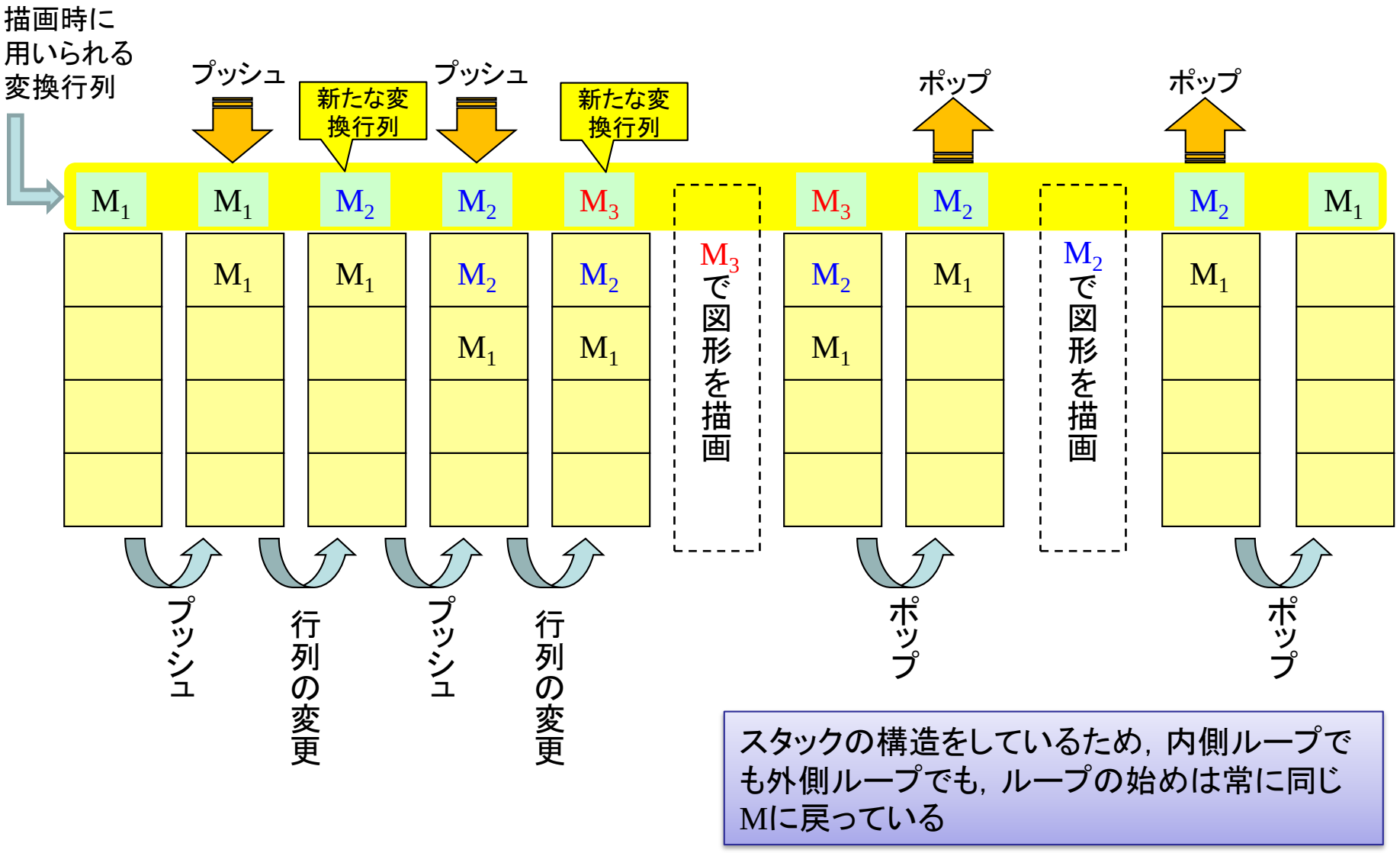
```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glColor3f( 1.0, 1.0, 1.0 );

    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)



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

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．

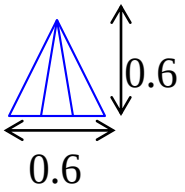


図1 八角錐のサイズ

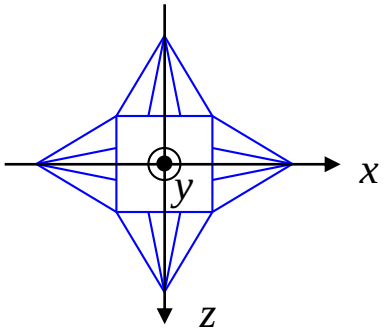


図2 ローカル座標

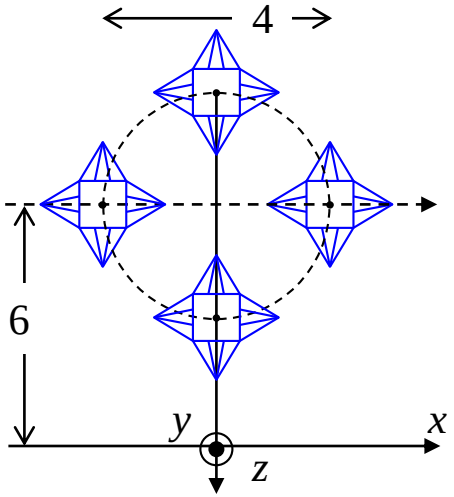


図3 ワールド座標

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();
}
```

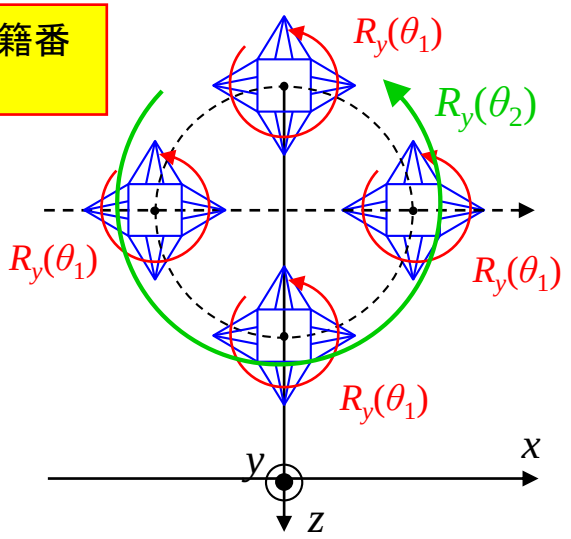
ウィンドウの名前を各自の学籍番号と氏名にすること

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

発展課題11

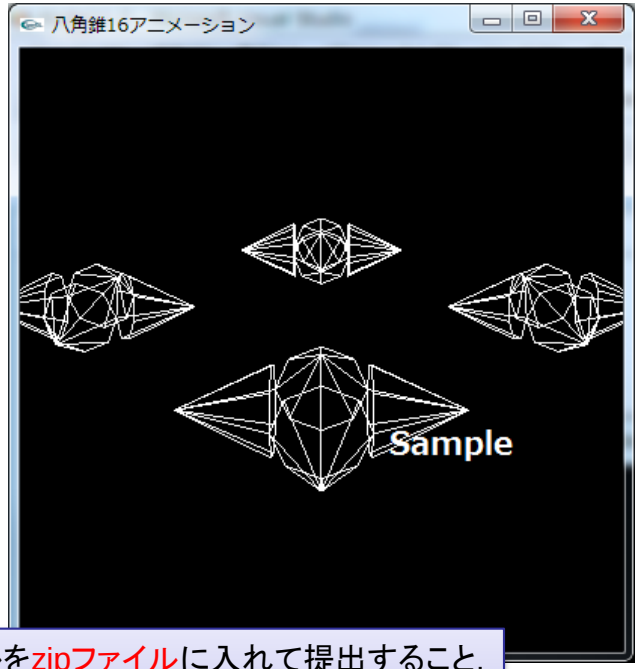
ウィンドウの名前を各自の学籍番号と氏名にすること

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)