

修了作品

OpenGLを用いて自由にコンピュータグラフィックス映像を制作しなさい

- 静止画でもアニメーションでもOK
- ワイヤーフレームモデルでもサーフェースモデルでもOK
- ゲームでもOK

提出物 ⇒ 最終回の授業で詳細を説明
締め切り

- 8月10日(土曜)深夜24時

提出物を誤ると評価できません

評価

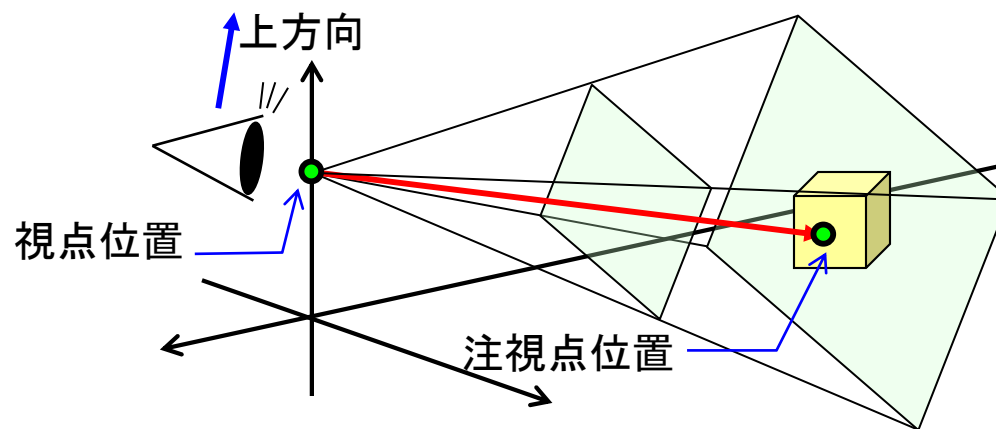
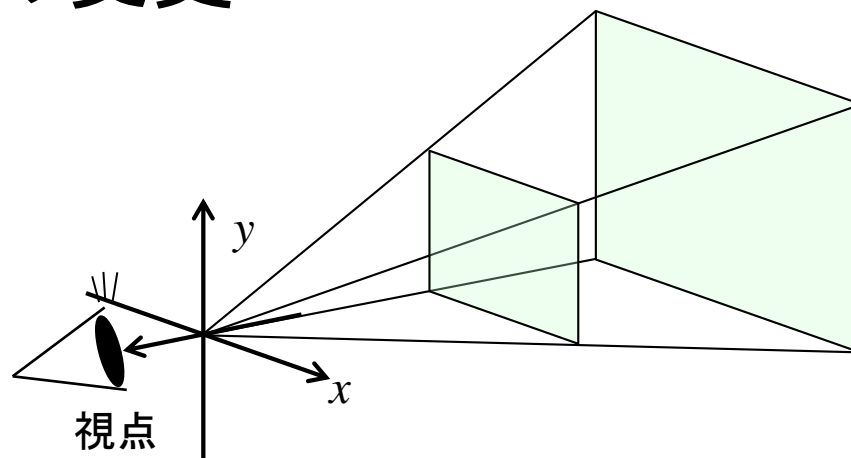
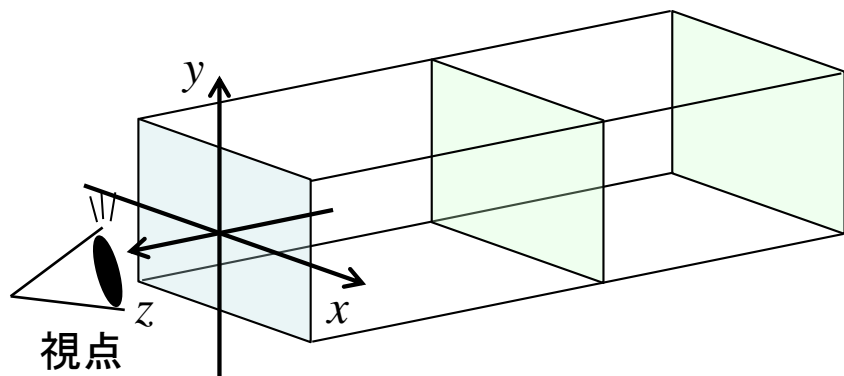
- 本授業の評価点100点のうち20点をこの課題で評価します.
- **独創的な作品**や**技術的に凝った作品**, **完成度が高い作品**を高く評価します.
- **美しいもの**や**楽しいもの**も高く評価します.

してはいけないこと

- 先輩の作品や、書籍やネットで見つけたソースをそのまま使って提出
⇒ 評価点を**マイナス50点**とします
- 同じ作品や似たような作品が提出された場合
⇒ その**作品数**で割り算して評価点にします.

他人が作らないものを作ってください！

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

タイマーによる正確なアニメーション

復習

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();
}
```

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度ずつ回転させている

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の中で再びタイマー番号1番のタイマーを起動する

```
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);

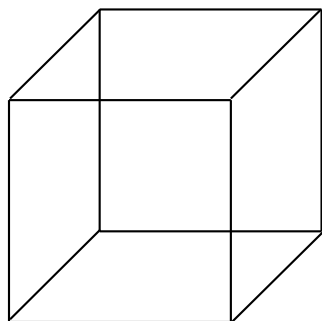
    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutTimerFunc(100, IncAngle, 1);
    glutMainLoop();
}
```

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が発生する。
- ✓ 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

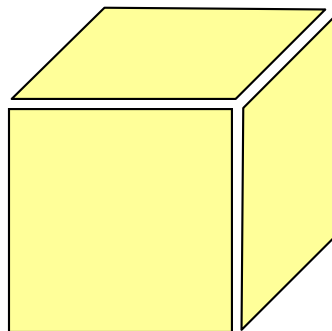
100ミリ秒毎にタイマーイベントが発生する

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

頂点1

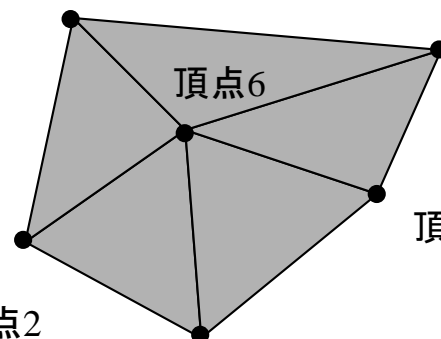
頂点6

頂点5

頂点4

頂点2

頂点3



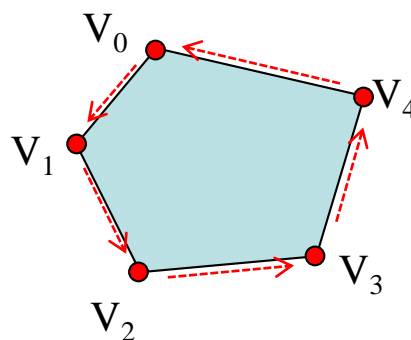
ポリゴン頂点の様々な指定方法

正方形を描
画する例

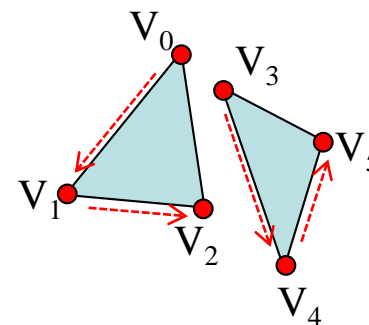
```
glBegin(GL_POLYGON);
glVertex3f(-1, -1, 0);
glVertex3f(+1, -1, 0);
glVertex3f(+1, +1, 0);
glVertex3f(-1, +1, 0);
glEnd();
```

頂点座標

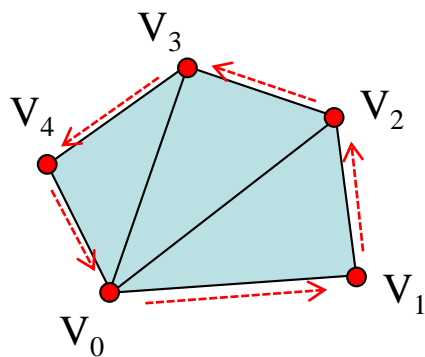
GL_POLYGON



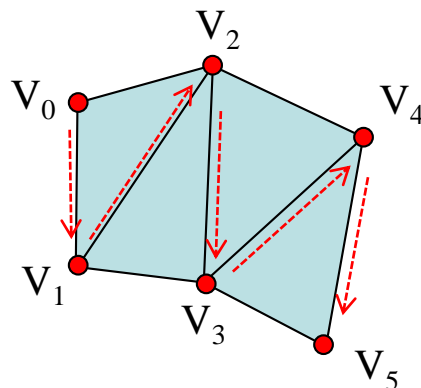
GL_TRIANGLES



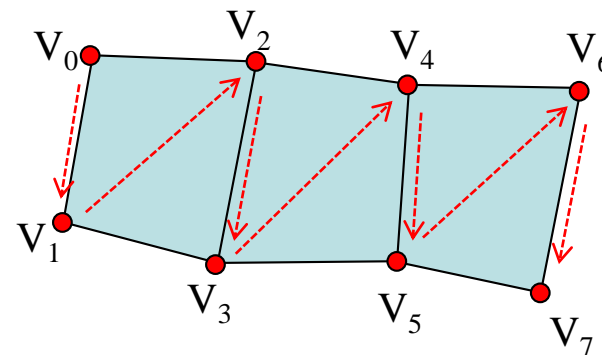
GL_TRIANGLE_FAN



GL_TRIANGLE_STRIP



GL_QUAD_STRIP



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 Squire(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 );

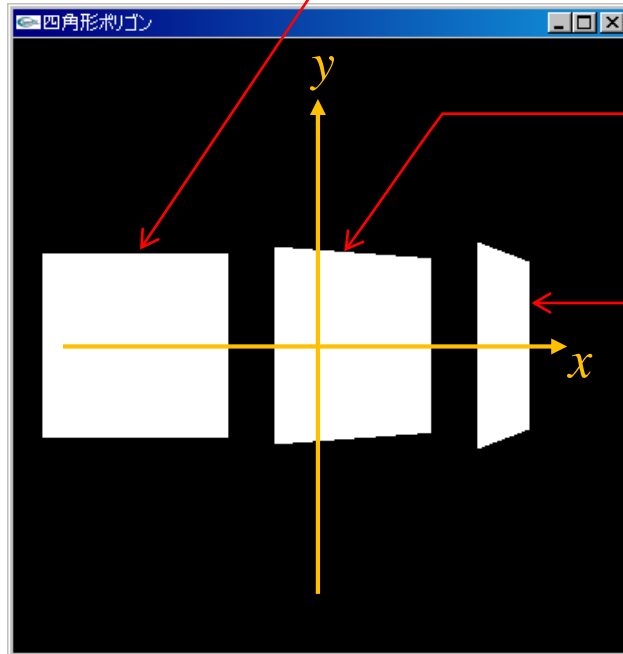
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

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

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

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

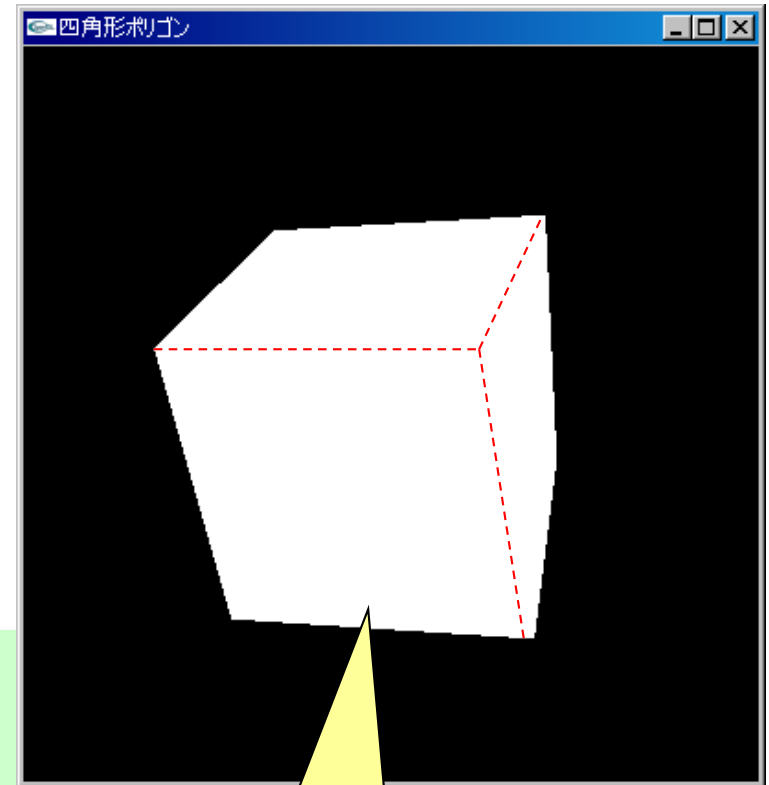
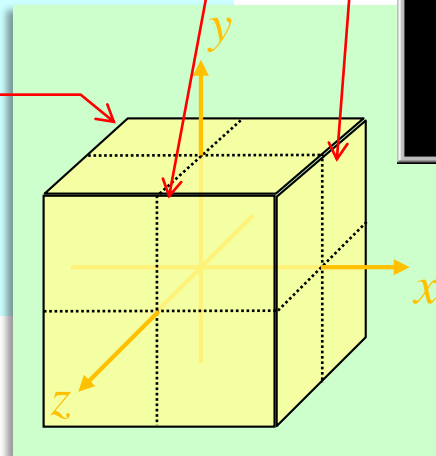
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glTranslatef( 0.0, 0.0, -6.0 );
    glRotatef( -15, 0, 1, 0 );
    glRotatef( 30, 1, 0, 0 );

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

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

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

    glFlush();
}
```



陰影が無いと立体に見えない！

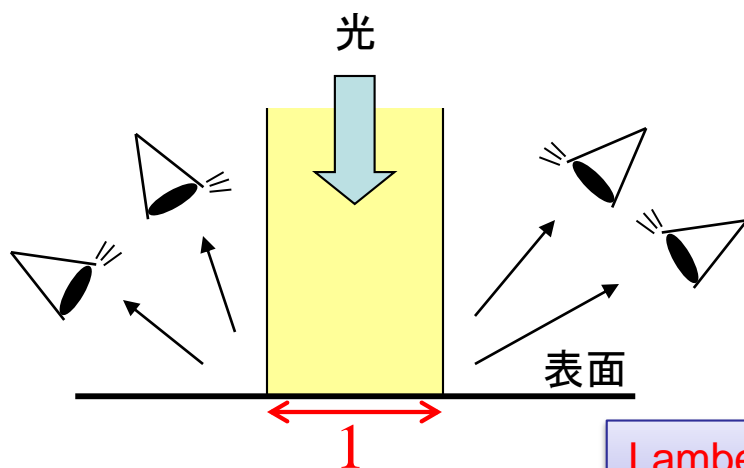
シェーディング

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。

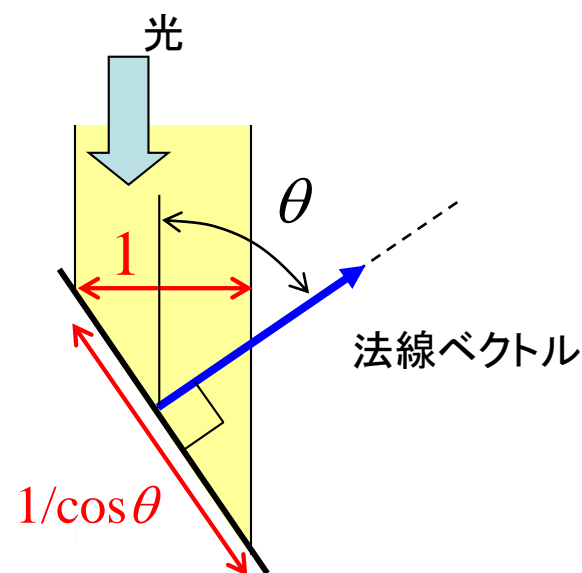


Lambertの余弦則
(Lambert反射)

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

立方体のシェーディング

Example12-5

```
void Squire(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);
    Squire(2.0);
    glPopMatrix();

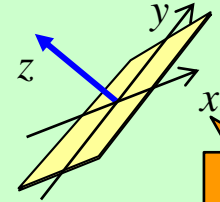
    glPushMatrix(); // 側面
    glTranslatef(1.0, 0.0, 0.0);
    glRotatef(90, 0, 1, 0);
    Squire(2.0);
    glPopMatrix();

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

復習

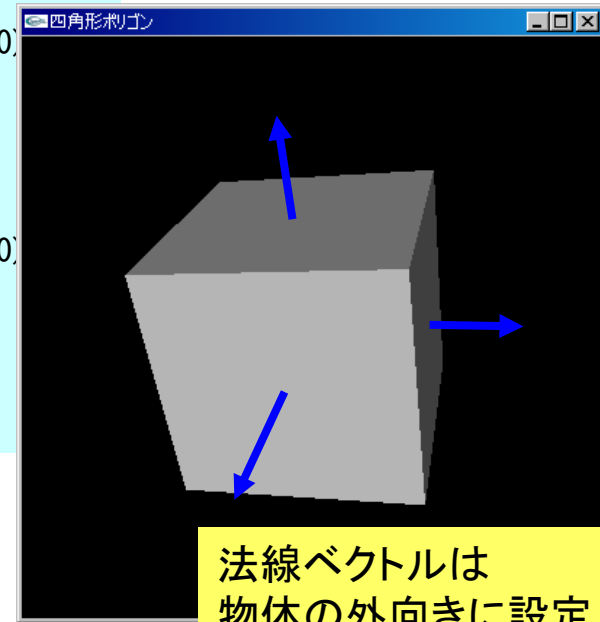
Example12-5から変更点無し

法線ベクトル



ローカル座標

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

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();
```

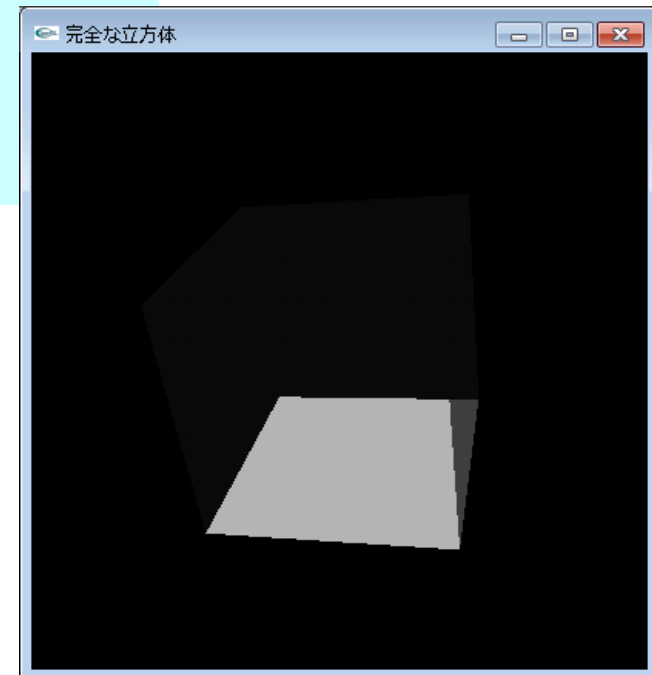
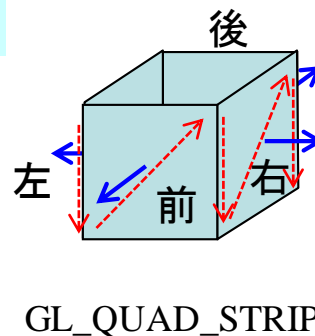
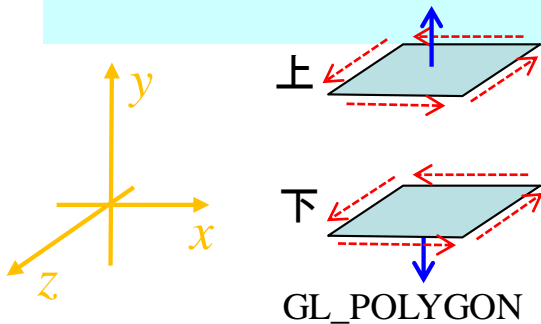
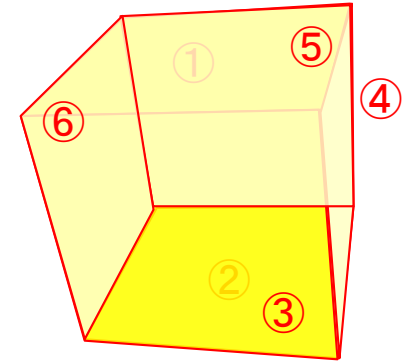
```
    glBegin(GL_QUAD_STRIP);
        glNormal3f(0, 0, 1); //前面③
        glVertex3f(-1, +1, +1);
        glVertex3f(-1, -1, +1);
        glVertex3f(+1, +1, +1);
        glVertex3f(+1, -1, +1);

        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();
}
```



z-バッファ法による隠面消去

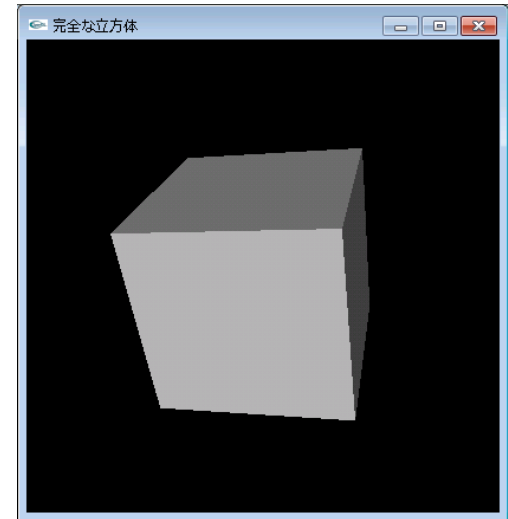
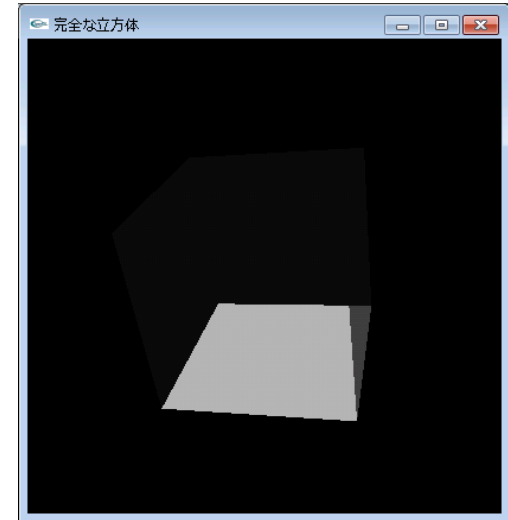
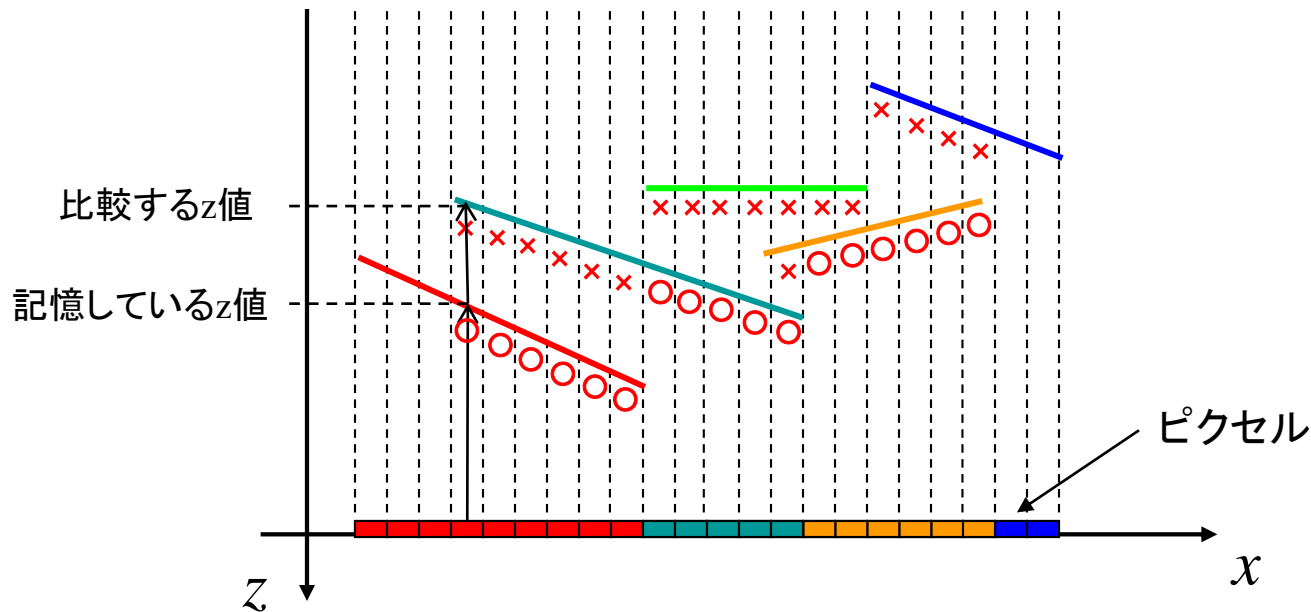
復習

隠面消去

後ろ側に隠れていて見えない部分を消す処理

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, 射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

復習

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, 0, 0);
```

```
    glBegin(GL_POLYGON);
```

```
        glNormal3f(0, 0, 1);  
        glVertex3f(-1, +1, +1);  
        glVertex3f(+1, +1, +1);  
        glVertex3f(+1, -1, +1);  
        glVertex3f(-1, -1, +1);  
    glEnd();
```

```
    glBegin(GL_POLYGON);
```

```
        glNormal3f(0, 0, -1);  
        glVertex3f(-1, +1, -1);  
        glVertex3f(+1, +1, -1);  
        glVertex3f(+1, -1, -1);  
        glVertex3f(-1, -1, -1);  
    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();  
}
```

(3) z-バッファ
もクリアする

(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);
```

```
glNormal3f(0, 0, -1);    //裏面
```

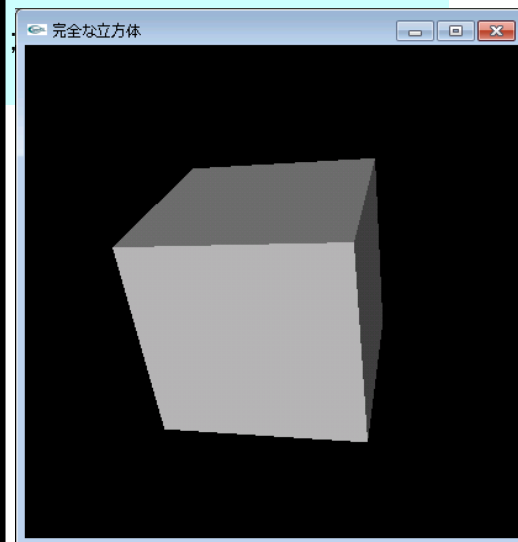
```
glVertex3f(-1, +1, -1);
```

```
glVertex3f(-1, -1, -1);
```

```
glNormal3f(-1, 0, 0);    //左側面
```

```
glVertex3f(-1, +1, +1);
```

```
glVertex3f(-1, -1, +1);
```



基本課題12

復習

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

復習

ウィンドウの名前を各自の
学籍番号と氏名にすること

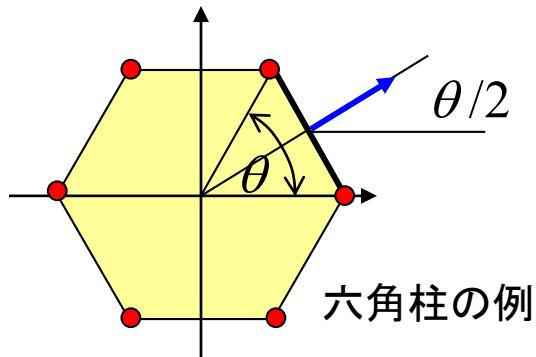
- ①底面の直径が2で高さが4, 中心の座標が(0, 0, -8)のサーフェスモデルの十二角柱
 - ②直径が1で高さが4, 中心の座標が(0, 0, -8)のサーフェスモデルの八角柱
- これら二つを組み合わせ, 基本課題と同じmain()関数を用いて実行例と同じ結果を得なさい。

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)
}
```

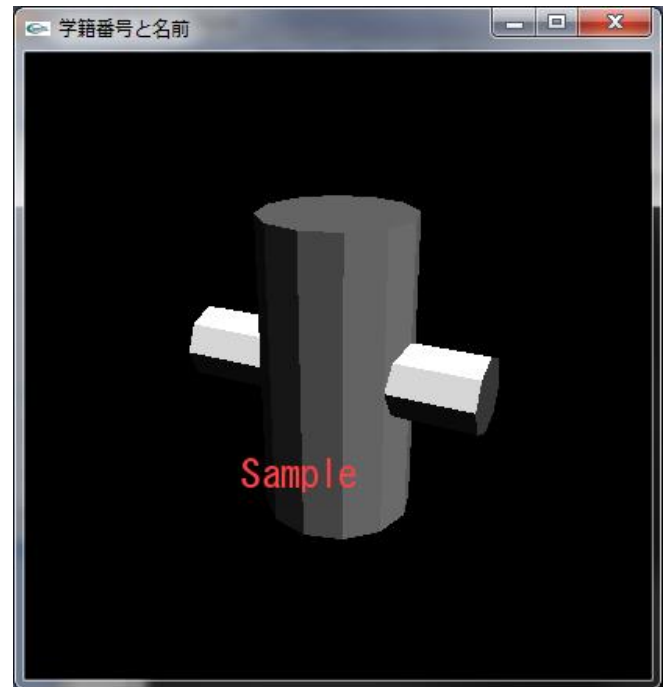
このとき, 側面の法線ベクトルは次のように考える



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



基本課題12 解答例

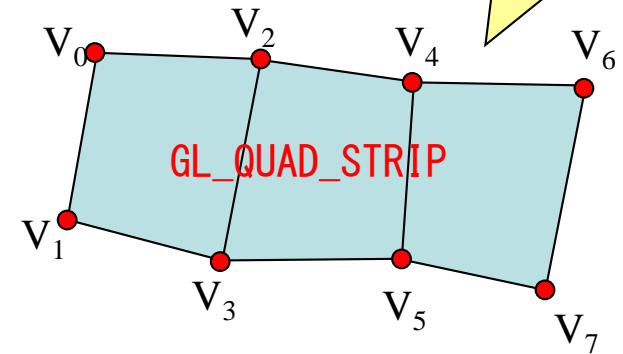
GL_QUAD_STRIPの使用法の誤り(1)

```
void Prism(int N)
{
    double angle=2*3.1415/N;
    int i;
    glBegin(GL_POLYGON);      //上面
    glNormal3f(0, +1, 0);
    for (i = 0; i < N; i++)
    {
        glVertex3f(cos(i*angle), +1.0, sin(i*angle));
    }
    glEnd();

    glBegin(GL_POLYGON);      //下面
    glNormal3f(0, -1, 0);
    for (i = 0; i < N; i++)
    {
        glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    }
    glEnd();

    glBegin(GL_QUAD_STRIP);    //側面
    for(i=0; i < N; i++)
    {
        glNormal3f(cos((i+0.5)*angle), 0, sin((i+0.5)*angle));
        glVertex3f(cos(i*angle), +1.0, sin(i*angle));
        glVertex3f(cos(i*angle), -1.0, sin(i*angle));
        glVertex3f(cos((i+1)*angle), +1.0, sin((i+1)*angle));
        glVertex3f(cos((i+1)*angle), -1.0, sin((i+1)*angle));
    }
    glEnd();
}
```

n 個の面を作るためには $2n+2$ 個の頂点が必要



$$\begin{array}{c} 4 + 2 + 2 + \dots = 4 + 2(n-1) \\ \uparrow \\ \text{1個目の面} \quad \quad \quad (n-1)\text{個の面} \end{array}$$

$4n$ 個の頂点を指定してしまっている。

基本課題12 解答例

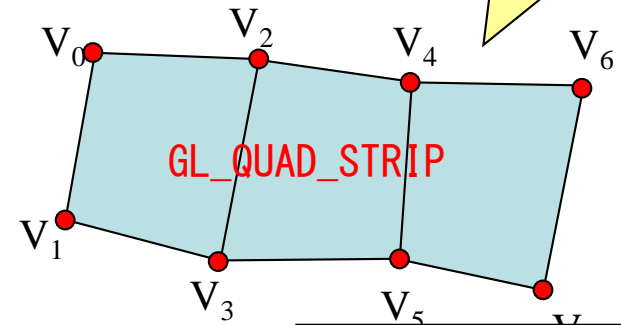
GL_QUAD_STRIPの使用法の誤り(2)

```
void Prism(int N)
{
    double angle=2*3.1415/N;
    int i;
    glBegin(GL_POLYGON);    //上面
    glNormal3f(0, +1, 0);
    for (i = 0; i < N; i++)
    {
        glVertex3f(cos(i*angle), +1.0, sin(i*angle));
    }
    glEnd();

    glBegin(GL_POLYGON);    //下面
    glNormal3f(0, -1, 0);
    for (i = 0; i < N; i++)
    {
        glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    }
    glEnd();

    glBegin(GL_QUAD_STRIP); //側面
    for(i=0; i < N; i++)
    {
        glNormal3f(cos((i+0.5)*angle), 0, sin((i+0.5)*angle));
        glVertex3f(cos(i*angle), +1.0, sin(i*angle));
        glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    }
    glEnd();
}
```

n 個の面を作るためには $2n+2$ 個の頂点が必要



だが、法線がまだおかしい...

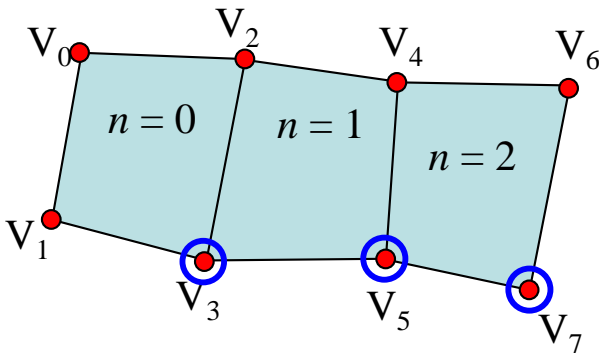
$n+1$ 回ループすると $2n+2$ 個の頂点が指定される。

$2n$ 個の頂点しか指定していない。

基本課題12 解答例

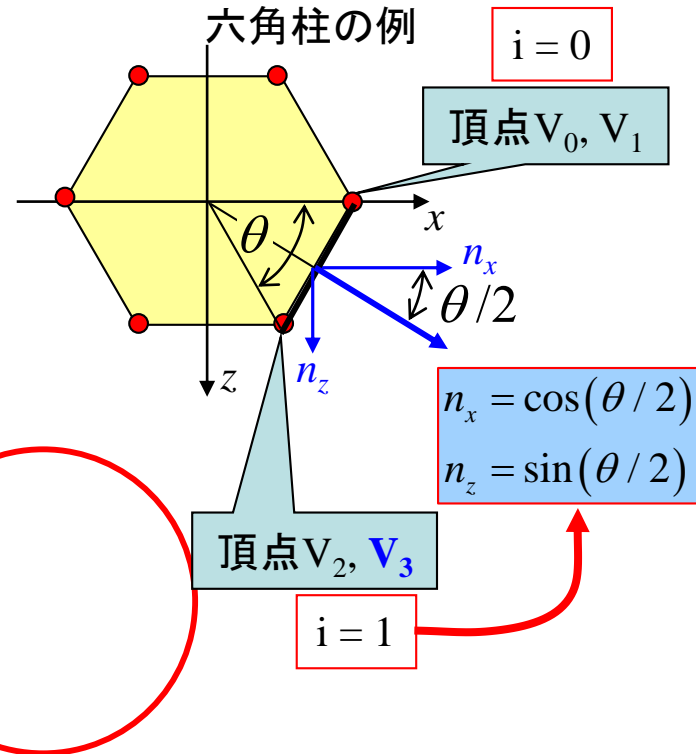
法線ベクトル設定の誤り

```
glBegin(GL_QUAD_STRIP); //側面
for (i = 0 ; i <= N ; i++)
{
    glNormal3f(cos((i+0.5)*angle), 0, sin((i+0.5)*angle));
    glVertex3f(cos(i*angle), +1.0, sin(i*angle));
    glVertex3f(cos(i*angle), -1.0, sin(i*angle));
}
glEnd();
```



ポリゴンを形成する
のに必要な最後の
頂点を設定した時
の法線ベクトルがそ
のポリゴンの法線ベ
クトルになる

```
glBegin(GL_QUAD_STRIP); //側面
for (i = 0 ; i <= N ; i++)
{
    glNormal3f(cos((i-0.5)*angle), 0, sin((i-0.5)*angle));
    glVertex3f(cos(i*angle), +1.0, sin(i*angle));
    glVertex3f(cos(i*angle), -1.0, sin(i*angle));
}
glEnd();
```



基本課題12 解答例

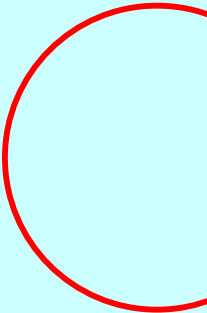
```
void Prism(int N) {
    int i;
    double angle = 2 * 3.1415 / N;

    glBegin(GL_POLYGON);          //上面
    glNormal3f(0, +1, 0);
    for (i = 0; i < N; i++) {
        glVertex3f(cos(i*angle), +1.0, sin(i*angle));
    }
    glEnd();

    glBegin(GL_POLYGON);          //下面
    glNormal3f(0, -1, 0);
    for (i = 0; i < N; i++) {
        glVertex3f(cos(i*angle), -1.0, sin(i*angle));
    }
    glEnd();

    //側面
    glBegin(GL_QUAD_STRIP); // 1 面目
    glNormal3f(sin(angle) / 2, 0.0, (1 + cos(angle)) / 2);
    glVertex3f(0, +1.0, 1);
    glVertex3f(0, -1.0, 1);
    glVertex3f(sin(angle), +1.0, cos(angle));
    glVertex3f(sin(angle), -1.0, cos(angle));

    //2からN面目
    for (i = 2; i <= N; i++) {
        glNormal3f((sin(i*angle) + sin((i - 1)*angle)) / 2, 0.0, (cos(i*angle) + cos((i - 1)*angle)) / 2);
        glVertex3f(sin(i*angle), +1.0, cos(i*angle));
        glVertex3f(sin(i*angle), -1.0, cos(i*angle));
    }
    glEnd();
}
```



発展課題12 解答例

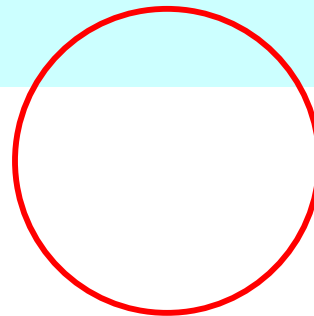
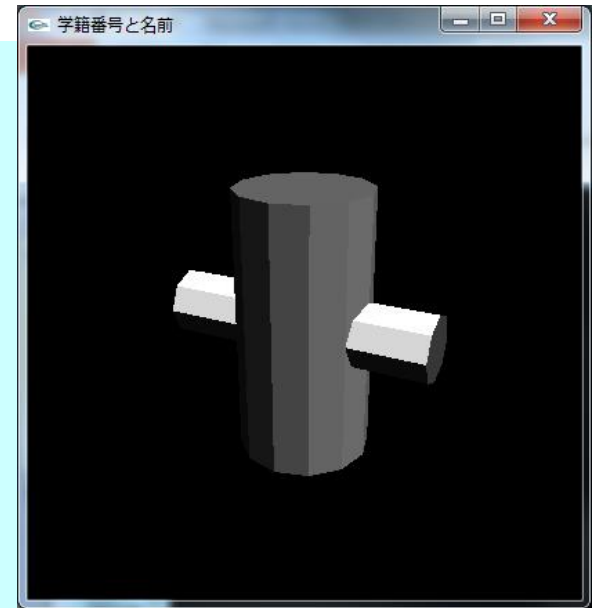
```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0, -8.0);

    glPushMatrix();
    glScalef(1.0, 2.0, 1.0);
    Prism(12);
    glPopMatrix();

    glPushMatrix();
    glRotatef(90, 0, 0, 1);
    glScalef(0.5, 2.0, 0.5);
    Prism(8);
    glPopMatrix();

    glFlush();
}
```



ライトの位置設定

Example13-1

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

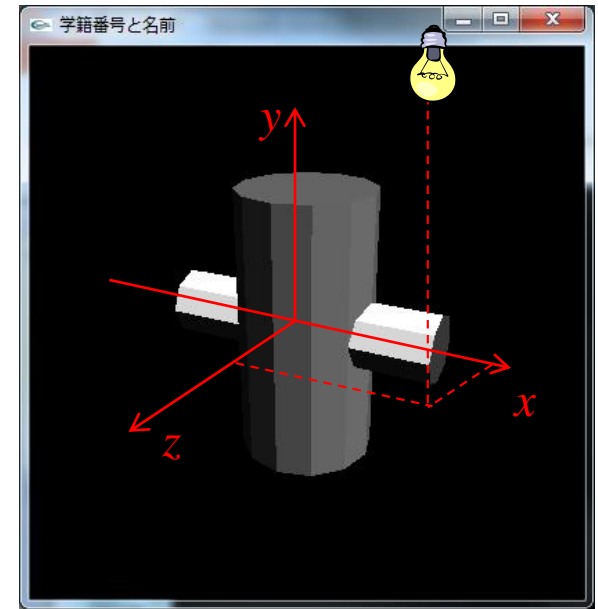
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    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);

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

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



位置ベクトル position
= (x, y, z, 0)

positionをライト0番
の位置として設定する

平行光源と点光源

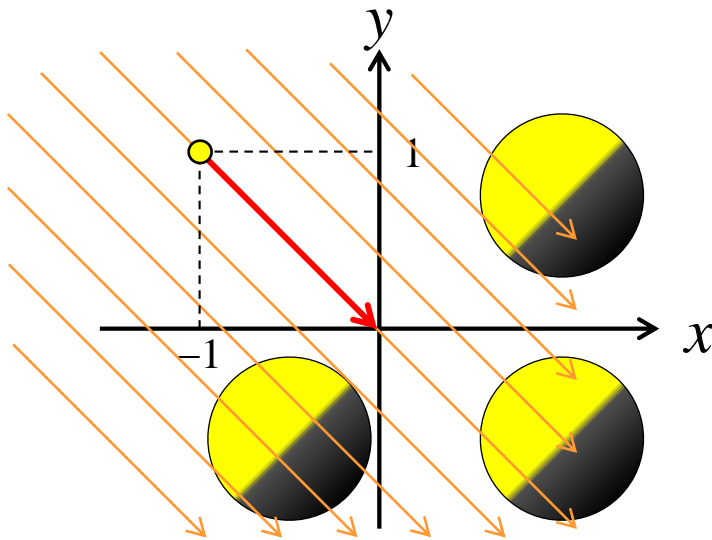
LIGHT0を**平行光源**に設定

```
float position[] = {-1.0, 1.0, 0.0, 0.0}; // 平行光源位置  
glLightfv(GL_LIGHT0, GL_POSITION, position); // 平行光源に設定
```

LIGHT0を**点光源**に設定

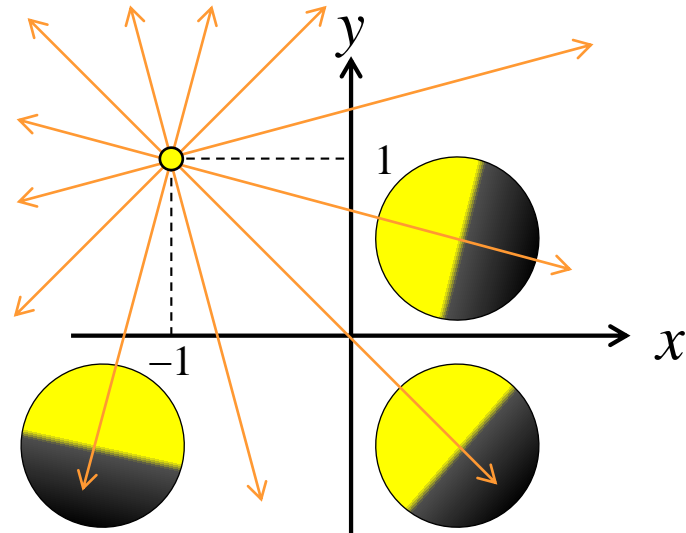
```
float position[] = {-1.0, 1.0, 0.0, 1.0}; // 点光源位置  
glLightfv(GL_LIGHT0, GL_POSITION, position); // 点光源に設定
```

配列の4番目の要素を1にすると点光源になる



平行光源

座標値は光源から原点に向かう**方向ベクトル**を表す



点光源

座標値は光源の**位置**を表す

ライト移動のアニメーション

Example13-2

```
// 問題12-2の改良
float angle = 0;

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

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0, -8.0);

    float position[] = {0.0, 5.0, 2.0, 0.0};
    glPushMatrix();
        glRotatef(angle, 0.0, 1.0, 0.0);
        glLightfv(GL_LIGHT0, GL_POSITION, position);
    glPopMatrix();

    glPushMatrix();
        glScalef(1.0, 2.0, 1.0);
        Prism(12);
    glPopMatrix();

    glPushMatrix();
        glRotatef(90, 0, 0, 1);
        glScalef(0.5, 2.0, 0.5);
        Prism(8);
    glPopMatrix();

    glFlush();
}
```

位置ベクトル
position
= (x, y, z, 0)

y軸周りでangle度回
転する幾何変換行列
を設定

ここでライトの位置を
設定. このとき, 幾何
変換行列で変換される

この物体の周りを
ライトが回転する

ライトの位置は幾何変換の
対象である

= ライトの座標に幾何変換行列Mをかけ
てからシェーディング処理が行われる

```
void idleFunc(void)
{
    angle += 0.05;
    if (angle >= 360)
        angle = 0;
    glutPostRedisplay();
}

int main(void)
{
    // . . .
    // main() 関数にて

    glutIdleFunc(idleFunc);
}
```

ライト色の設定

Example13-3

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

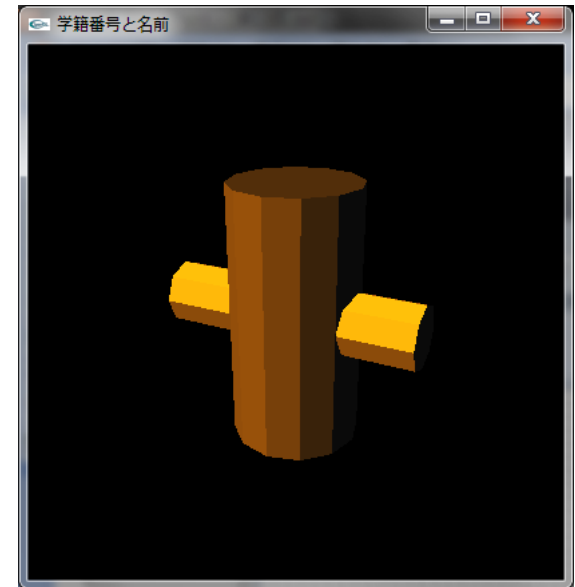
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    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);

    float amber[] = {1.0, 0.5, 0.0, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);

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



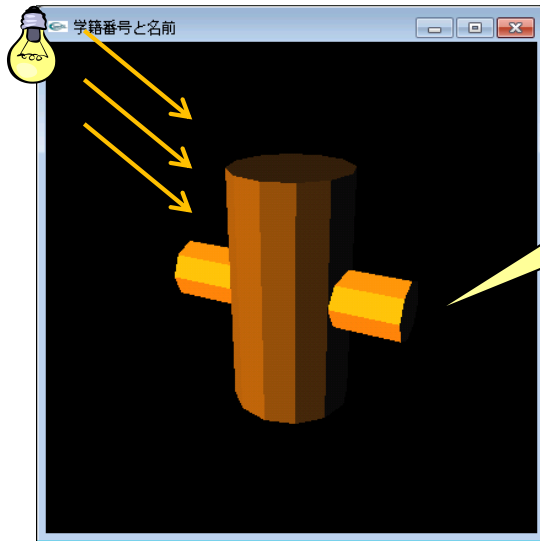
色 amber
= (R, G, B, 1.0)

amberをライト0番の色
として設定する

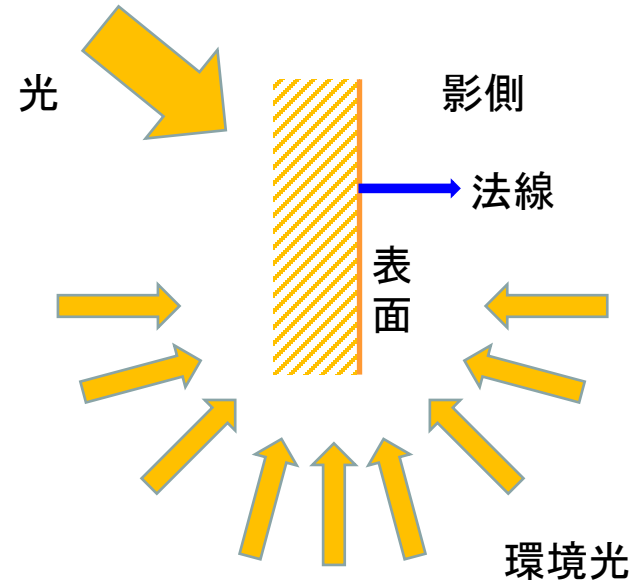
色自体をアニメーションで
変化することも可能！

amber[0] 赤色の明度
amber[1] 緑色の明度
amber[2] 青色の明度

環境光



こちらの面が真っ黒



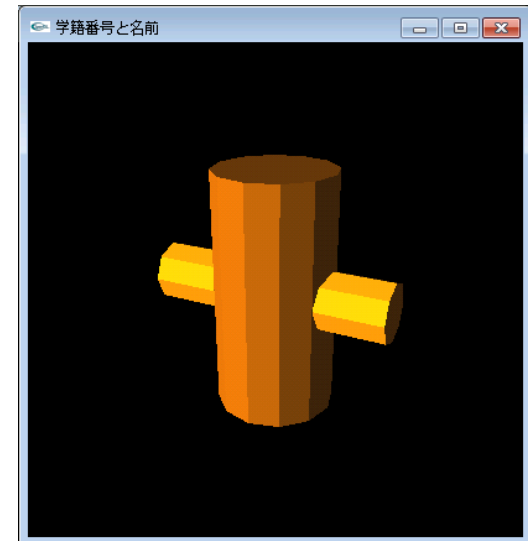
ambient

環境光 = 光源の位置や向きに関係なく周囲全体にある光

```
glShadeModel(GL_FLAT);  
glEnable(GL_LIGHT0);  
glEnable(GL_LIGHTING);  
  
float amber[] = {1.0, 0.5, 0.0, 1.0};  
glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);  
glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
```

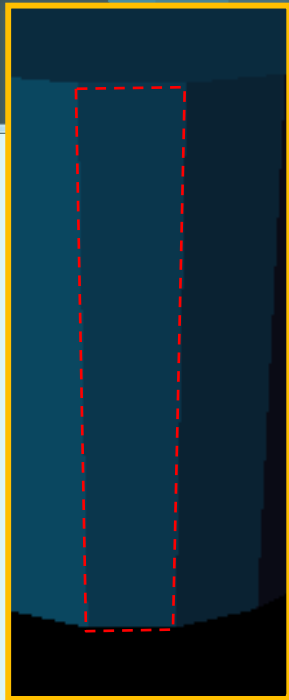
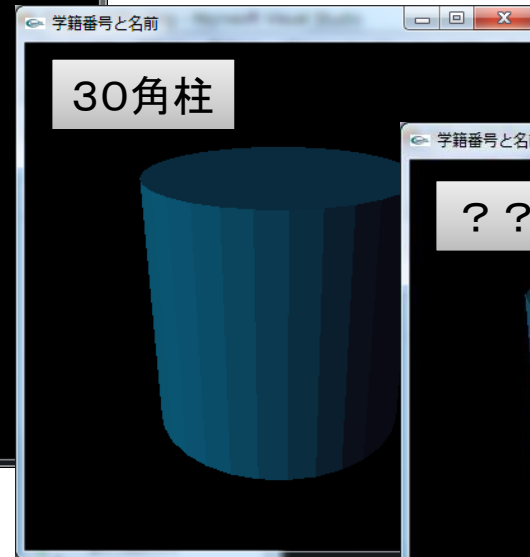
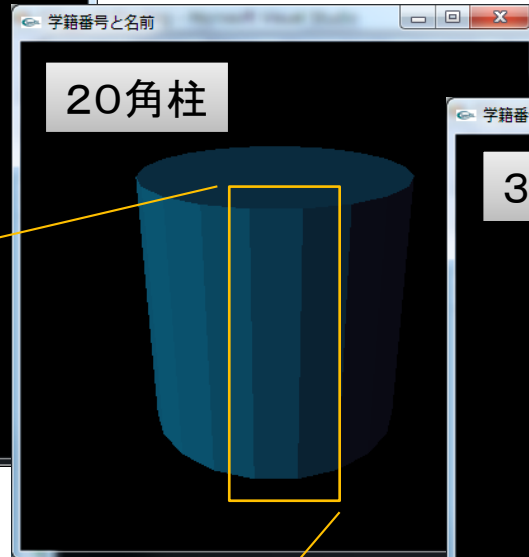
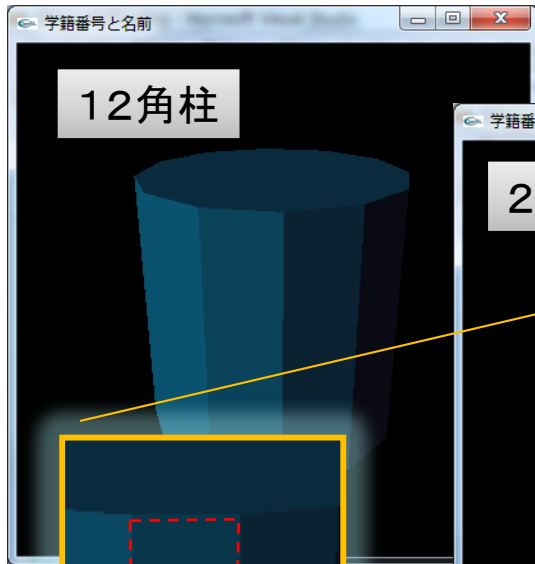
環境光もアンバー色に設定

環境光の色をライトの色と違う色に設定することも可能



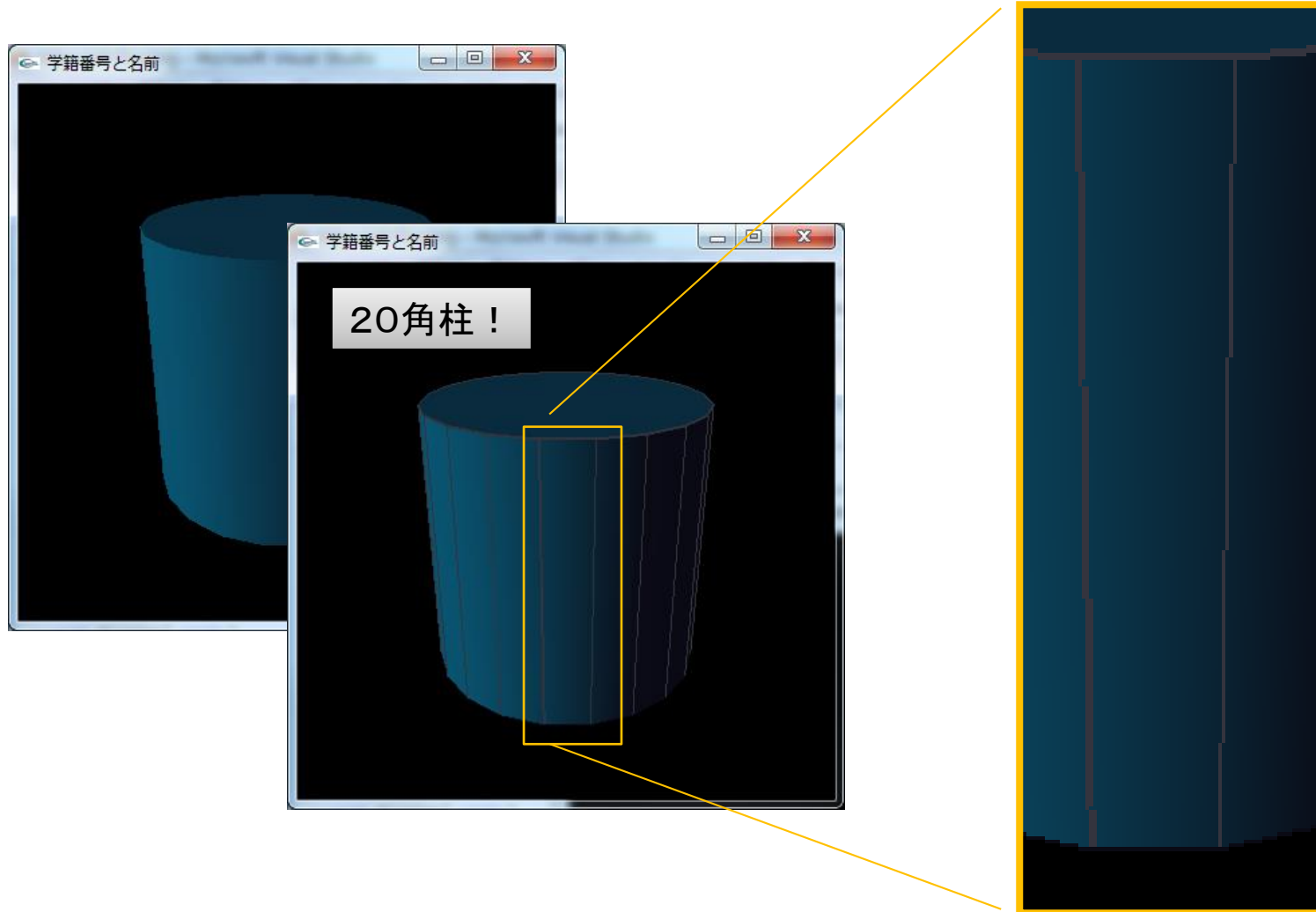
スムーズシェーディング(1)

滑らかな円柱を描こう！



フラットシェーディング
= 一つのポリゴンを同じ色で塗りつぶす

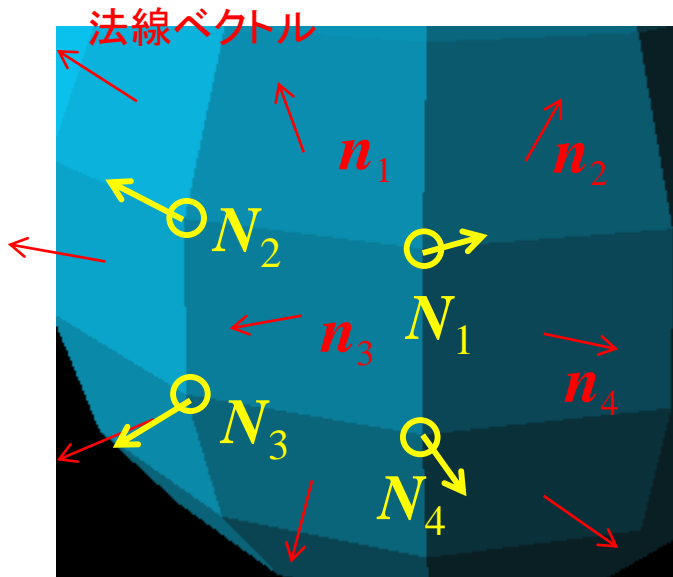
スムーズシェーディング(2)



スムーズシェーディング
＝ 一つのポリゴン内でグラデーションする

グーローシェーディング

(Gouraud shading)



(1) 頂点の法線ベクトルを求める

頂点の法線ベクトルは隣接するポリゴンの法線ベクトルの平均値

(例)

$$N_1 = \frac{n_1 + n_2 + n_3 + n_4}{4}$$

(2) 各頂点の明るさを求める

Lambertの余弦則を用いる

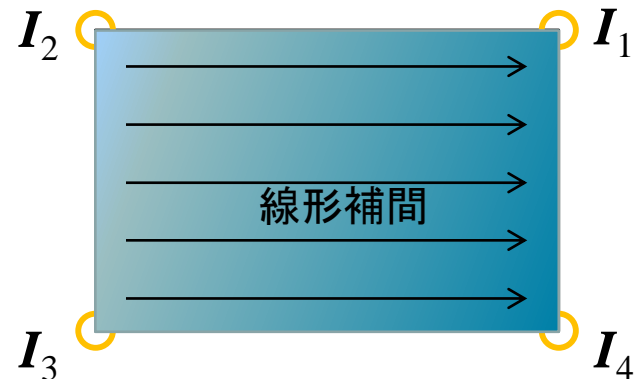
(3) 頂点の明るさを線形補間してポリゴン内を塗りつぶす

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



OpenGLでのスムーズシェーディングの設定

Example13-4

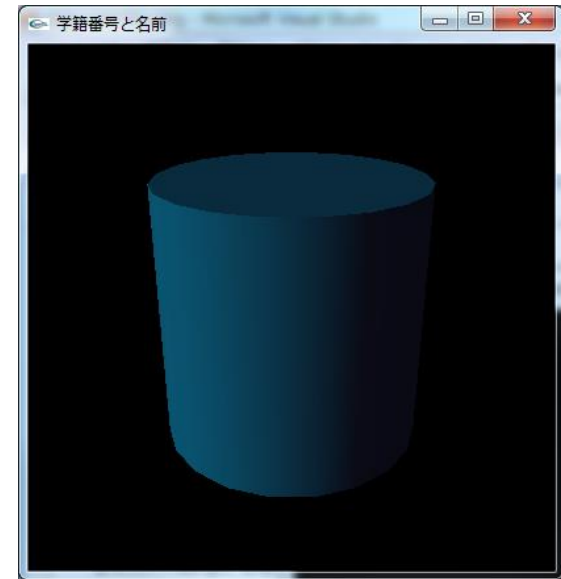
```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

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

    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);

    float blue[] = {0.0, 0.8, 1.0, 1.0};
    float tblue[] = {0.0, 0.0, 0.2, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, blue);
    glLightfv(GL_LIGHT0, GL_AMBIENT, tblue);

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



シェーディングをスムーズに切り替える

ペイントハンドラでは20角柱を描画

GLUTの幾何オブジェクト(1)

Example13-5

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
```

```
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, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutWireSphere(1, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutWireTorus(0.3, 0.8, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glScalef(0.3, 0.3, 0.3);
    glutWireDodecahedron();
    glPopMatrix();
```

```
    glPushMatrix();
    glTranslatef(-1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glutWireTeapot(0.8);
    glPopMatrix();

    glFlush();
}

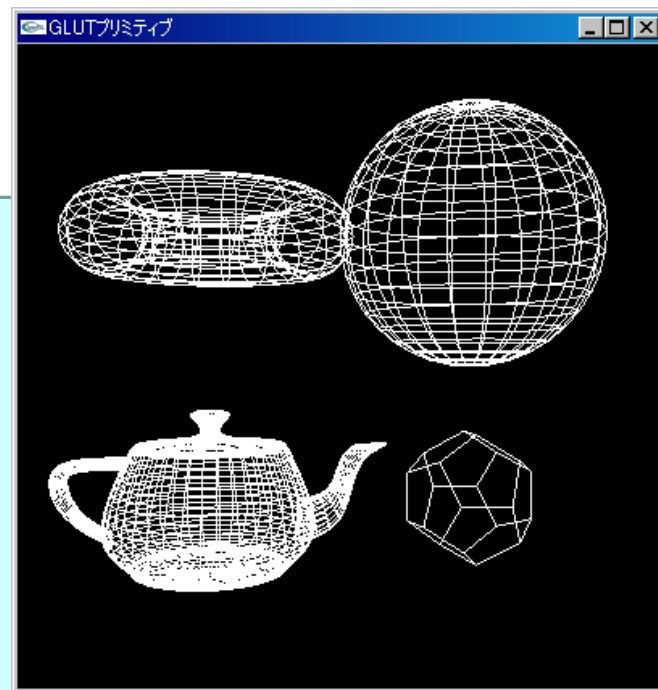
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("GLUTプリミティブ");

    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

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

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



詳細は

<http://opengl.jp/glut/section11.html>

GLUTの幾何オブジェクト(2)

Example13-6

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

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

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);

    glPushMatrix();
    glTranslatef(1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidSphere(1, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.3, 0.8, 20, 20);
    glPopMatrix();

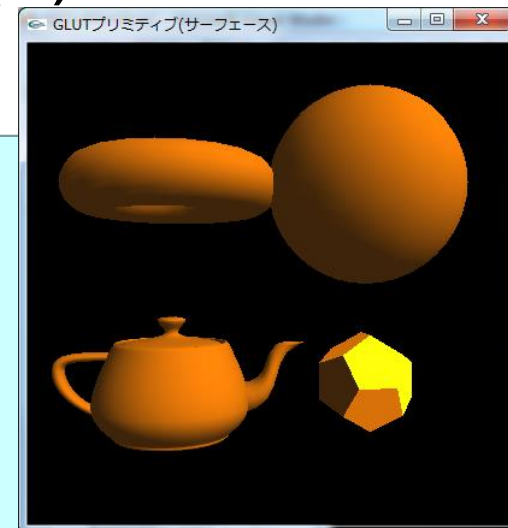
    glPushMatrix();
    glTranslatef(1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glScalef(0.3, 0.3, 0.3);
    glutSolidDodecahedron();
    glPopMatrix();
```

```
    glPushMatrix();
    glTranslatef(-1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glutSolidTeapot(0.8);
    glPopMatrix();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("GLUTプリミティブ(サーフェース)");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 10.0);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    float position[] = {1.0, 1.0, 1.0, 0.0};
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    float amber[] = {1.0, 0.5, 0.0, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
    glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
    glEnable(GL_DEPTH_TEST);

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



詳細は

<http://opengl.jp/glut/section11.html>

基本課題13

ウィンドウの名前を各自の学籍番号と氏名にすること

次のmain()関数をそのまま用い、実行例のように様々なレンダリングが混在する画像を作成しなさい。なお、物体はすべて内径0.2, 外径0.7で20×20に分割したトーラスである。

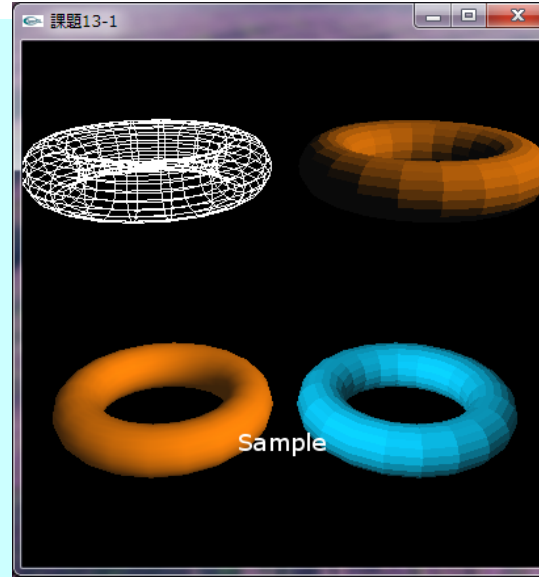
Report13-1

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
void display( void );
float blue[] = {0.0, 0.8, 1.0, 1.0}; //青
float amber[] = {1.0, 0.5, 0.0, 1.0}; //アンバー
float white[] = {1.0, 1.0, 1.0, 1.0}; //白
float black[] = {0.0, 0.0, 0.0, 1.0}; //黒
float positionR[] = {1.0, 1.0, 1.0, 0.0}; //右上位置
float positionL[] = {-1.0, 1.0, 1.0, 0.0}; //左上位置

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("課題13-1");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(35, 1.0, 1.0, 10.0);
    gluLookAt(0, 2.5, 0, 0, 1.2, -3, 0, 1, 0);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    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
 - glutウィンドウの画面コピー
- ② 実行プログラム (○○○.exe)

ヒント:シェーディングをオフにするには下記を用いる
glDisable(GL_LIGHTING)

左上 <中心 (-1, +1, -6)>
ワイヤフレーム
シェーディング無し
白色

右上 <中心 (+1, +1, -6)>
サーフェース
フラットシェーディング
色:アンバー(右上から照明)
環境光:なし

左下 <中心 (-1, -1, -6)>
サーフェース
スムーズシェーディング
色:アンバー(右上から照明)
環境光:アンバー

右下<中心 (+1, -1, -6)>
サーフェース
フラットシェーディング
色:青(左上から照明)
環境光:青

glutウィンドウを変化しても常に同じレンダリング結果になることを提出前に確認！