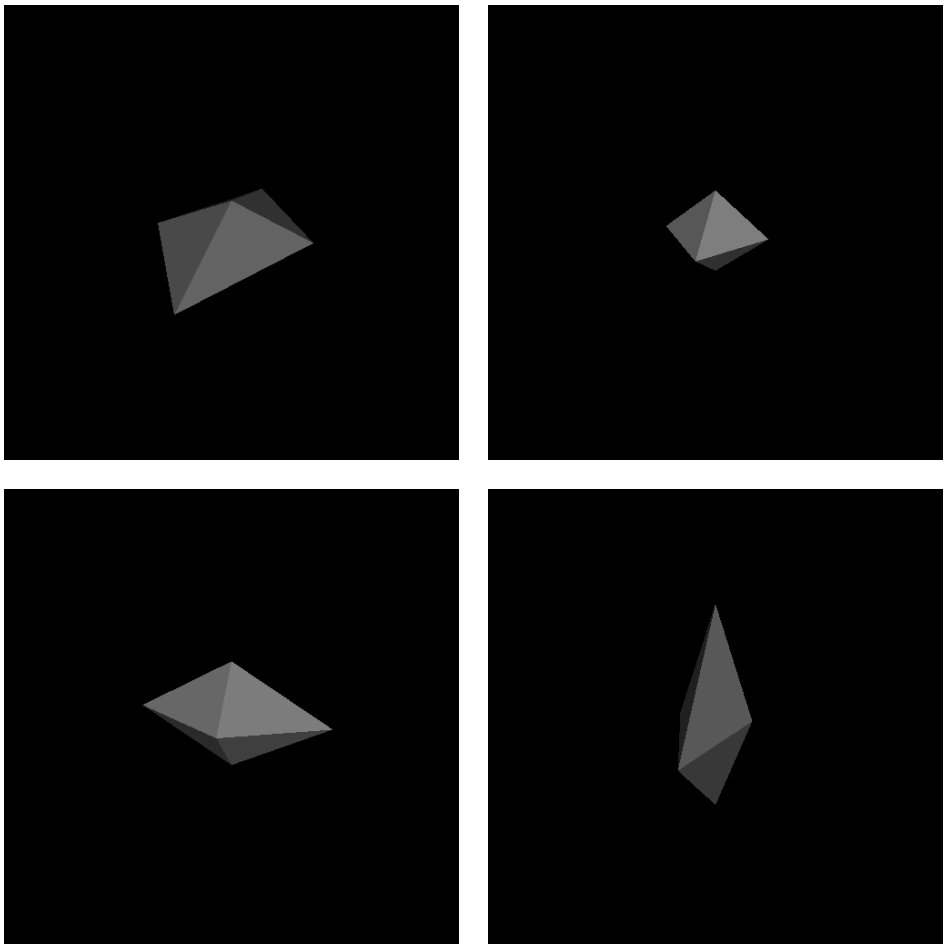
OpenGLのその他の機能

- さらにもっとOpenGLの機能を活用したい人のためにサンプルプログラムを用意しました
- 解説はしないので，興味のある人は各自勉強してください

スケーリング

- 拡大縮小はglScaled関数を使います
- 私は個人的にはglScaled関数が好きじゃないです
- OpenGLの場合，ローカル座標系の考え方なので，「図形が拡大縮小している」のではなく「座標系が拡大縮小している」というのがバグを生みやすくするからです
- 空間が伸び縮みしている，空間がゆがむ，ということです
- glScaled関数を呼び出した後は，ゆがんだ空間での座標系で平行移動をするため，座標の指定がややこしくなります
- できるだけ，glScaled関数を呼び出した後は「図形を描画するだけ」で済むようにして，glScaled関数のあとで平行移動などをする必要がないような使い方を（可能であれば）したほうがいいと思います
- 少なくとも，glPushMatrix関数やglPopMatrix関数を使って，glScaled関数が不要となったらすぐに元のゆがんでない空間に戻したほうがいいと思います
- 法線ベクトルもglScaled関数の影響を受けるため，陰影も正確ではなくなります
- glScaled関数を使わずに済むならばそのほうがいいです
- 例1：拡大縮小した倍率に応じて，y軸に沿って図形が移動してしまう
    - glScaled(scalex, scaley, scalez);
    - glTranslated(0.0, -1.0, 0.0);
    - glutSolidOctahedron();
- 例2：y軸に-1移動したところを中心に図形が拡大縮小するので図形は移動しない
    - glTranslated(0.0, -1.0, 0.0);
    - glScaled(scalex, scaley, scalez);
    - glutSolidOctahedron();



サンプルソースコード

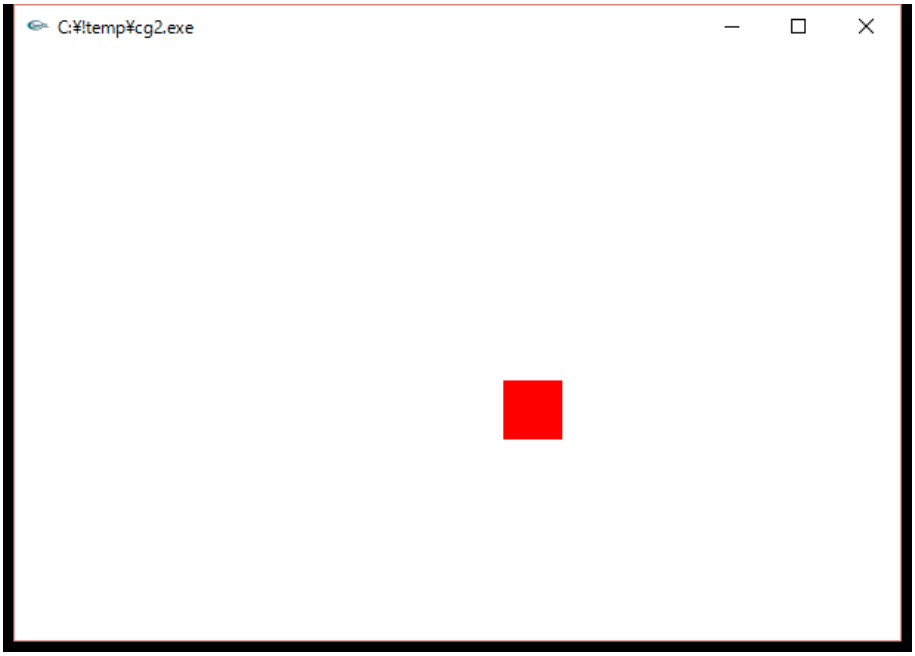
```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 100;
double scalex;
double scaley;
double scalez;
double scalevx;
double scalevy;
double scalevz;

void myDisplay()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(3.0, -10.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glPushMatrix();

    glScaled(scalex, scaley, scalez);
    //glTranslated(0.0, -1.0, 0.0);
    glutSolidOctahedron();

    glPopMatrix();
    glutSwapBuffers();
}

void myTimer(int value)
{
    if (value == 1) {
        scalex += scalevx;
        scaley += scalevy;
        scalez += scalevz;
        if (scalex > 5.0) {
            scalex = 5.0;
            scalevx = -scalevx;
        }
        if (scaley > 5.0) {
            scaley = 5.0;
            scalevy = -scalevy;
        }
        if (scalez > 5.0) {
            scalez = 5.0;
            scalevz = -scalevz;
        }
        if (scalex < 1.0) {
            scalex = 1.0;
            scalevx = -scalevx;
        }
        if (scaley < 1.0) {
            scaley = 1.0;
            scalevy = -scalevy;
        }
        if (scalez < 1.0) {
            scalez = 1.0;
            scalevz = -scalevz;
        }

        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}
```

サンプルソースコード

```
void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    scalex = scaley = scalez = 1.0;
    scalevx = 0.23;
    scalevy = 0.29;
    scalevz = 0.31;

    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

マウスの動き

- マウスボタンを押したか放したかはglutMouseFunc関数で指定するコールバック関数で調べることができますが，マウスをドラッグして動かした位置はglutMotionFunc関数で指定するコールバック関数で調べます
- サンプルプログラムを実行すると赤い四角形が表示されるので，それをマウスの左ボタンでドラッグすることができます



サンプルソースコード

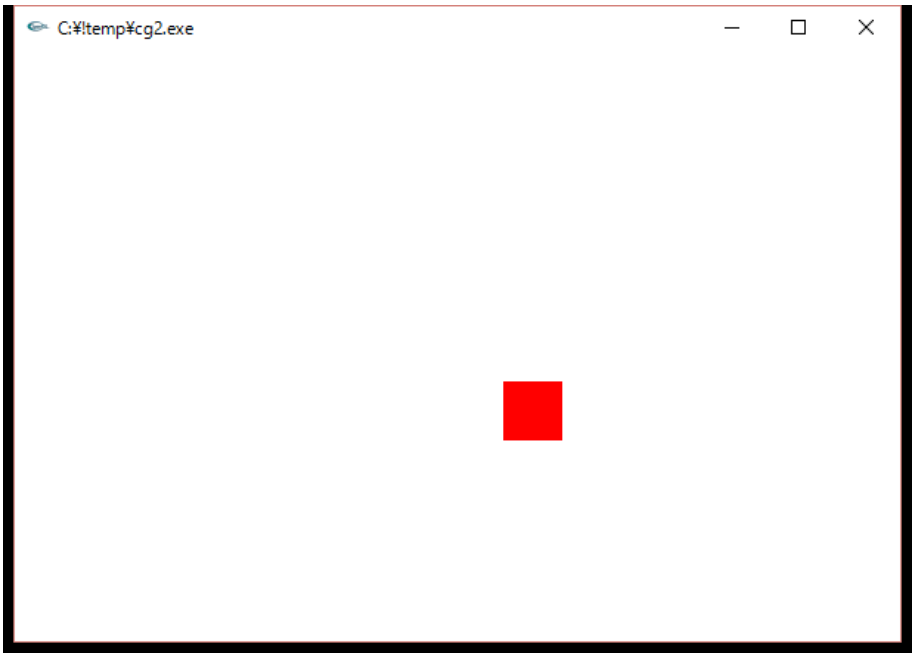
```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;
bool mousedrag;
int prevx;
int prevy;
int centerx;
int centery;
const int LENGTH = 20;

void myDisplay()
{
    double posr = 2.0 * (double)LENGTH / (double)winw;
    double posl = -posr;
    double posu = 2.0 * (double)LENGTH / (double)winh;
    double posd = -posu;
    double midx = 2.0 * (double)centerx / (double)winw - 1.0;
    double midy = -(2.0 * (double)centery / (double)winh - 1.0);

    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glTranslated(midx, midy, 0.0);
    glBegin(GL_QUADS);
    glColor3d(1.0, 0.0, 0.0);
    glVertex2d(posl, posu);
    glVertex2d(posl, posd);
    glVertex2d(posr, posd);
    glVertex2d(posr, posu);
    glEnd();
    glPopMatrix();
    glutSwapBuffers();
}

void myMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON) {
        if (state == GLUT_DOWN) {
            if (__max(abs(x - centerx), abs(y - centery)) <= LENGTH) {
                mousedrag = true;
                prevx = x;
                prevy = y;
            }
        }
        else if (state == GLUT_UP) {
            mousedrag = false;
        }
    }
}

void myMotion(int x, int y)
{
    if (mousedrag) {
        centerx += x - prevx;
        centery += y - prevy;
        prevx = x;
        prevy = y;
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}
```

サンプルソースコード

```
void myReshape(int width, int height)
{
    glViewport(0, 0, width, height);
    winw = width;
    winh = height;
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(600, 400);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    mousedrag = false;
    centerx = 100;
    centery = 100;

    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutMouseFunc(myMouse);
    glutMotionFunc(myMotion);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

特殊キー

- アルファベットなどのASCII文字を押したかはglutKeyboardFunc関数で指定するコールバック関数で調べることができますが，矢印キーなどの特殊キーを押したかはglutSpecialFunc関数で指定するコールバック関数で調べます
- サンプルプログラムを実行すると赤い四角形が表示されるので，それを矢印キーで移動させることができます



サンプルソースコード

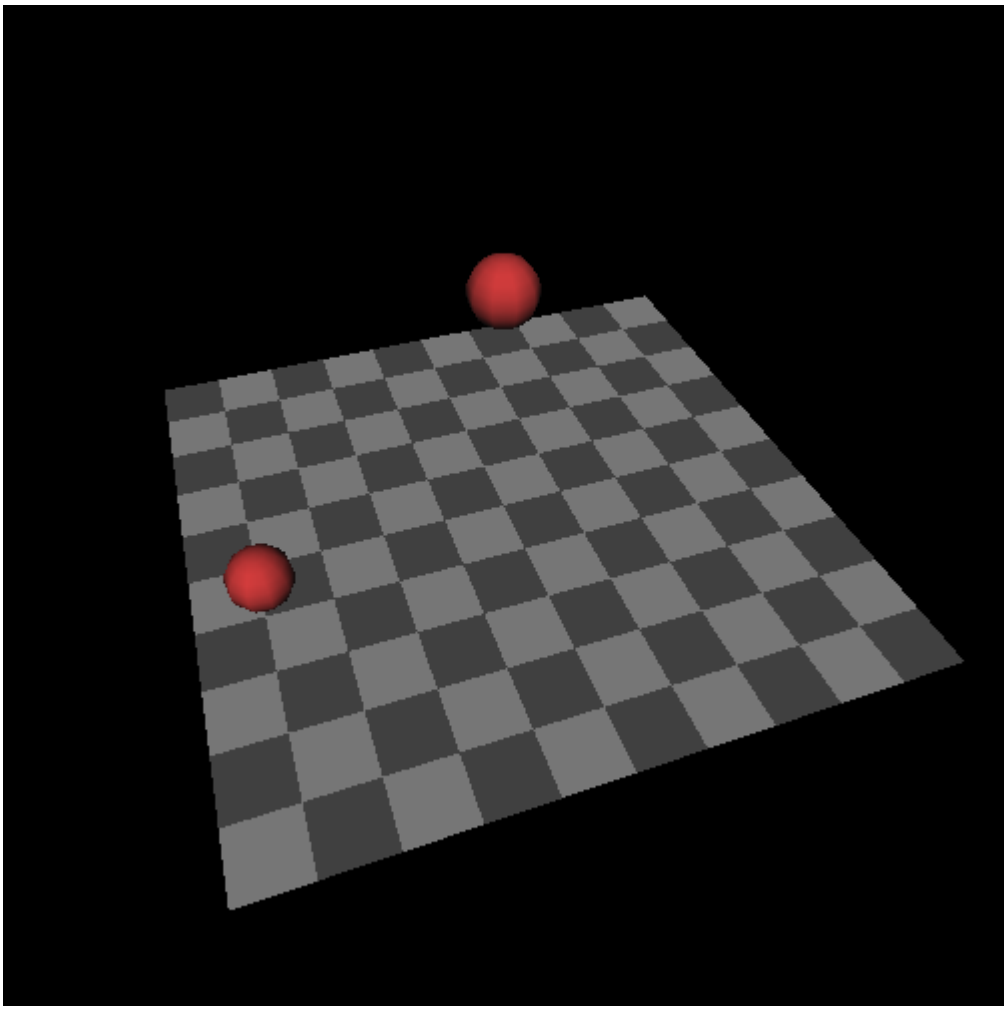
```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;
int centerx;
int centery;
const int LENGTH = 20;

void myDisplay()
{
    double posr = 2.0 * (double)LENGTH / (double)winw;
    double posl = -posr;
    double posu = 2.0 * (double)LENGTH / (double)winh;
    double posd = -posu;
    double midx = 2.0 * (double)centerx / (double)winw - 1.0;
    double midy = -(2.0 * (double)centery / (double)winh - 1.0);

    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glTranslated(midx, midy, 0.0);
    glBegin(GL_QUADS);
    glColor3d(1.0, 0.0, 0.0);
    glVertex2d(posl, posu);
    glVertex2d(posl, posd);
    glVertex2d(posr, posd);
    glVertex2d(posr, posu);
    glEnd();
    glPopMatrix();
    glutSwapBuffers();
}

void mySpecial(int key, int x, int y)
{
    if (key == GLUT_KEY_LEFT) centerx--;
    if (key == GLUT_KEY_UP) centery--;
    if (key == GLUT_KEY_RIGHT) centerx++;
    if (key == GLUT_KEY_DOWN) centery++;
    glutPostRedisplay();
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    glViewport(0, 0, width, height);
    winw = width;
    winh = height;
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(600, 400);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}
```

サンプルソースコード

```
int main(int argc, char* argv[])
{
    centerx = 100;
    centery = 100;

    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutSpecialFunc(mySpecial);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

カメラワーク

- gluLookAt関数の使い方は既に説明済みなので，ここでは特に何も解説しません



サンプルソースコード

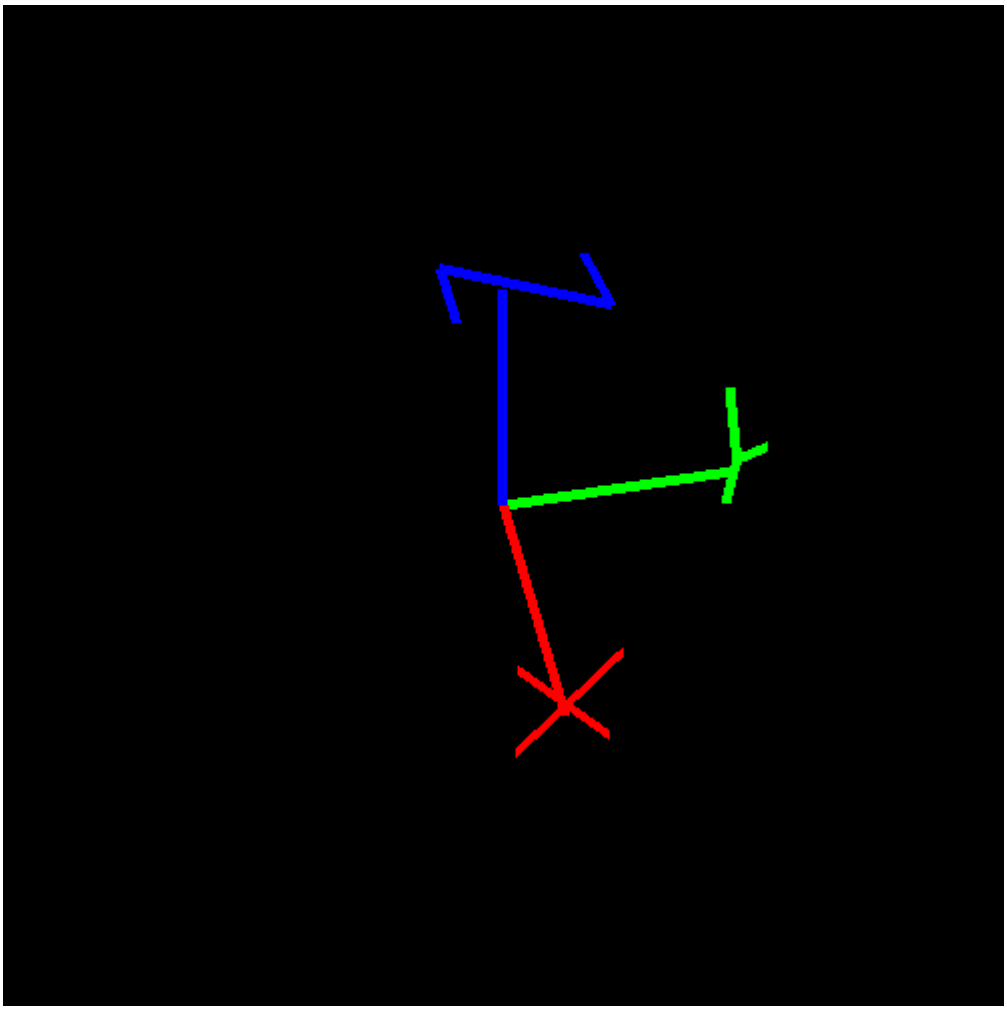
```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 50;
double dir = 0.0;
double angle = 0.0;
double height = 0.0;
const double heightvelocityinitial = 1.0;
double heightvelocity = heightvelocityinitial;
const double heightacceleration = -0.1;

void myGround()
{
    float ground[2][4] = {
        {0.6f, 0.6f, 0.6f, 1.0f},
        {0.3f, 0.3f, 0.3f, 1.0f}
    };
    int i, j;

    glBegin(GL_QUADS);
    glNormal3d(0.0, 0.0, 1.0);
    for (j = -5; j < 5; j++) {
        for (i = -5; i < 5; i++) {
            glMaterialfv(GL_FRONT, GL_DIFFUSE, ground[abs(i + j) % 2]);
            glVertex3d((double)i, (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)(j + 1), 0.0);
            glVertex3d((double)i, (double)(j + 1), 0.0);
        }
    }
    glEnd();
}

void myDisplay()
{
    const double RADIUS = 0.5;
    float red[4] = { 0.8f, 0.2f, 0.2f, 1.0f };
    double eyex = 10.0 * cos(angle);
    double eyey = 10.0 * sin(angle);
    double eyez = 10.0;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);
    myGround();

    glMaterialfv(GL_FRONT, GL_DIFFUSE, red);

    glPushMatrix();
    glTranslated(0.0, 0.0, height + RADIUS);
    glutSolidSphere(RADIUS, 10, 10);
    glPopMatrix();

    glPushMatrix();
    glRotated(dir, 0.0, 0.0, 1.0);
    glTranslated(4.0, 0.0, RADIUS);
    glutSolidSphere(RADIUS, 10, 10);
    glPopMatrix();

    glutSwapBuffers();
}
```

サンプルソースコード

```
void myTimer(int value)
{
    if (value == 1) {
        dir += 5.0;
        angle += 0.01;
        height += heightvelocity;
        heightvelocity += heightacceleration;
        if (height < 0.0) {
            height = heightvelocityinitial;
            heightvelocity = heightvelocityinitial + heightacceleration;
        }

        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

3Dテキスト

- 2Dテキストを描画するにはglutBitmapCharacter関数を使いましたが，3Dテキストを描画するにはglutStrokeCharacter関数を使います
- 3Dテキストは何かと使いづらいです
- 使わないほうがいいです



サンプルソースコード

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

int winw, winh;
const int delay = 50;
const double anglevelocity = 0.01;
double angle = 0.0;
```

サンプルソースコード

```
void myDisplay()
{
    double eyex = 5.0 * cos(angle);
    double eyey = 5.0 * sin(angle);
    double eyez = 5.0;
    double width;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glPushAttrib(GL_LINE_BIT);
    glLineWidth(5.0f);

    glBegin(GL_LINES);
    glColor3d(1.0, 0.0, 0.0);
    glVertex3d(0.0, 0.0, 0.0);
    glVertex3d(2.0, 0.0, 0.0);
    glColor3d(0.0, 1.0, 0.0);
    glVertex3d(0.0, 0.0, 0.0);
    glVertex3d(0.0, 2.0, 0.0);
    glColor3d(0.0, 0.0, 1.0);
    glVertex3d(0.0, 0.0, 0.0);
    glVertex3d(0.0, 0.0, 2.0);
    glEnd();

    glPushMatrix();
    glColor3d(1.0, 0.0, 0.0);
    glTranslated(2.0, 0.0, 0.0);
    glRotated(90.0, 0.0, 0.0, 1.0);
    glRotated(90.0, 1.0, 0.0, 0.0);
    glScaled(0.01, 0.01, 0.01);
    width = (double)glutStrokeWidth(GLUT_STROKE_ROMAN, 'X');
    glTranslated(-width / 2.0, -width / 2.0, 0.0);
    glutStrokeCharacter(GLUT_STROKE_ROMAN, 'X');
    glPopMatrix();

    glPushMatrix();
    glColor3d(0.0, 1.0, 0.0);
    glTranslated(0.0, 2.0, 0.0);
    glRotated(180.0, 0.0, 0.0, 1.0);
    glRotated(90.0, 1.0, 0.0, 0.0);
    glScaled(0.01, 0.01, 0.01);
    width = (double)glutStrokeWidth(GLUT_STROKE_ROMAN, 'Y');
    glTranslated(-width / 2.0, -width / 2.0, 0.0);
    glutStrokeCharacter(GLUT_STROKE_ROMAN, 'Y');
    glPopMatrix();

    glPushMatrix();
    glColor3d(0.0, 0.0, 1.0);
    glTranslated(0.0, 0.0, 2.0);
    glScaled(0.01, 0.01, 0.01);
    width = (double)glutStrokeWidth(GLUT_STROKE_ROMAN, 'Z');
    glTranslated(-width / 2.0, -width / 2.0, 0.0);
    glutStrokeCharacter(GLUT_STROKE_ROMAN, 'Z');
    glPopMatrix();

    glPopAttrib();

    glutSwapBuffers();
}
```

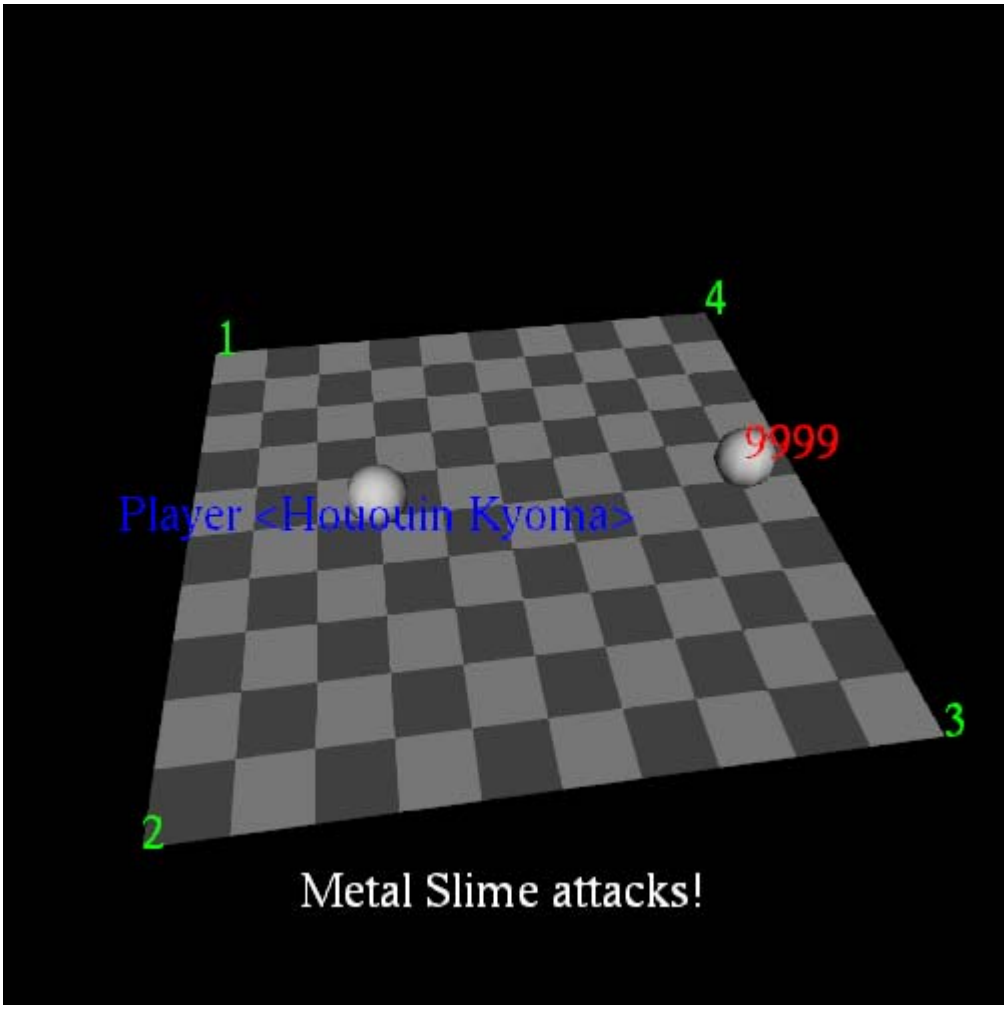
サンプルソースコード

```
void myTimer(int value)
{
    if (value == 1) {
        angle += anglevelocity;

        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

2Dテキスト

- 2Dテキストをどうやって描画するかということ自体は既に説明済みなので省略します
- みなさんが気になるのは「どうやって自分が思った通りの位置に文字を表示するか」ということだと思います
- そのためには「座標系」をきちんと理解する必要があります
- 「座標系」については最終的には，自分で考えて理解するしかありません

- サンプルプログラムではまず3DCGを描画してから，glDisable(GL_DEPTH_TEST)を実行してから文字を描画しています
  - これにより，3DCGの上に文字が描画されます
- projection行列とmodelview行列をglLoadIdentity関数により単位行列にすると，初期状態の座標系になります
  - 初期状態の座標系で文字を描画する座標を指定したい場合はこの方法を使うといいでしょう
- gluProject関数は3次元の座標を2次元の座標に変換する関数です



サンプルソースコード

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

int winw, winh;
const int delay = 50;
double dir = 0.0;
double angle = 0.0;

void myGround()
{
    float ground[2][4] = {
        {0.6f, 0.6f, 0.6f, 1.0f},
        {0.3f, 0.3f, 0.3f, 1.0f}
    };
    int i, j;

    glBegin(GL_QUADS);
    glNormal3d(0.0, 0.0, 1.0);
    for (j = -5; j < 5; j++) {
        for (i = -5; i < 5; i++) {
            glMaterialfv(GL_FRONT, GL_DIFFUSE, ground[abs(i + j) % 2]);
            glVertex3d((double)i, (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)(j + 1), 0.0);
            glVertex3d((double)i, (double)(j + 1), 0.0);
        }
    }
    glEnd();
}

void myDisplay()
{
    const double RADIUS = 0.5;
    float gray[4] = { 0.8f, 0.8f, 0.8f, 1.0f };
    double eyex = 10.0 * cos(angle);
    double eyey = 10.0 * sin(angle);
    double eyez = 10.0;
    static char text1[] = "Metal Slime attacks!";
    static char text2[] = "Player <Hououin Kyoma>";
    char* p;
    int w;
    double cx, cy, cz;
    double vm[16];
    double pm[16];
    int vp[4];

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
```

サンプルソースコード

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

myGround();

glMaterialfv(GL_FRONT, GL_DIFFUSE, gray);

glPushMatrix();
glRotated(dir, 0.0, 0.0, 1.0);
glTranslated(4.0, 0.0, RADIUS);
glutSolidSphere(RADIUS, 10, 10);
glPopMatrix();

glPushMatrix();
glRotated(-dir, 0.0, 0.0, 1.0);
glTranslated(2.0, 0.0, RADIUS);
glutSolidSphere(RADIUS, 10, 10);
glGetDoublev(GL_MODELVIEW_MATRIX, vm);
glGetDoublev(GL_PROJECTION_MATRIX, pm);
glGetIntegerv(GL_VIEWPORT, vp);
gluProject(0.0, 0.0, 0.0, vm, pm, vp, &cx, &cy, &cz);
glPopMatrix();

glDisable(GL_DEPTH_TEST);
glDisable(GL_LIGHTING);
glDisable(GL_LIGHT0);

glColor3d(0.0, 1.0, 0.0);
glRasterPos3d(-5.0, -5.0, 0.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '1');
glRasterPos3d(5.0, -5.0, 0.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '2');
glRasterPos3d(5.0, 5.0, 0.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '3');
glRasterPos3d(-5.0, 5.0, 0.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '4');

glPushMatrix();
glRotated(dir, 0.0, 0.0, 1.0);
glTranslated(4.0, 0.0, RADIUS);
glColor3d(1.0, 0.0, 0.0);
glRasterPos3d(0.0, 0.0, 0.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '9');
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '9');
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '9');
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, '9');
glPopMatrix();
```

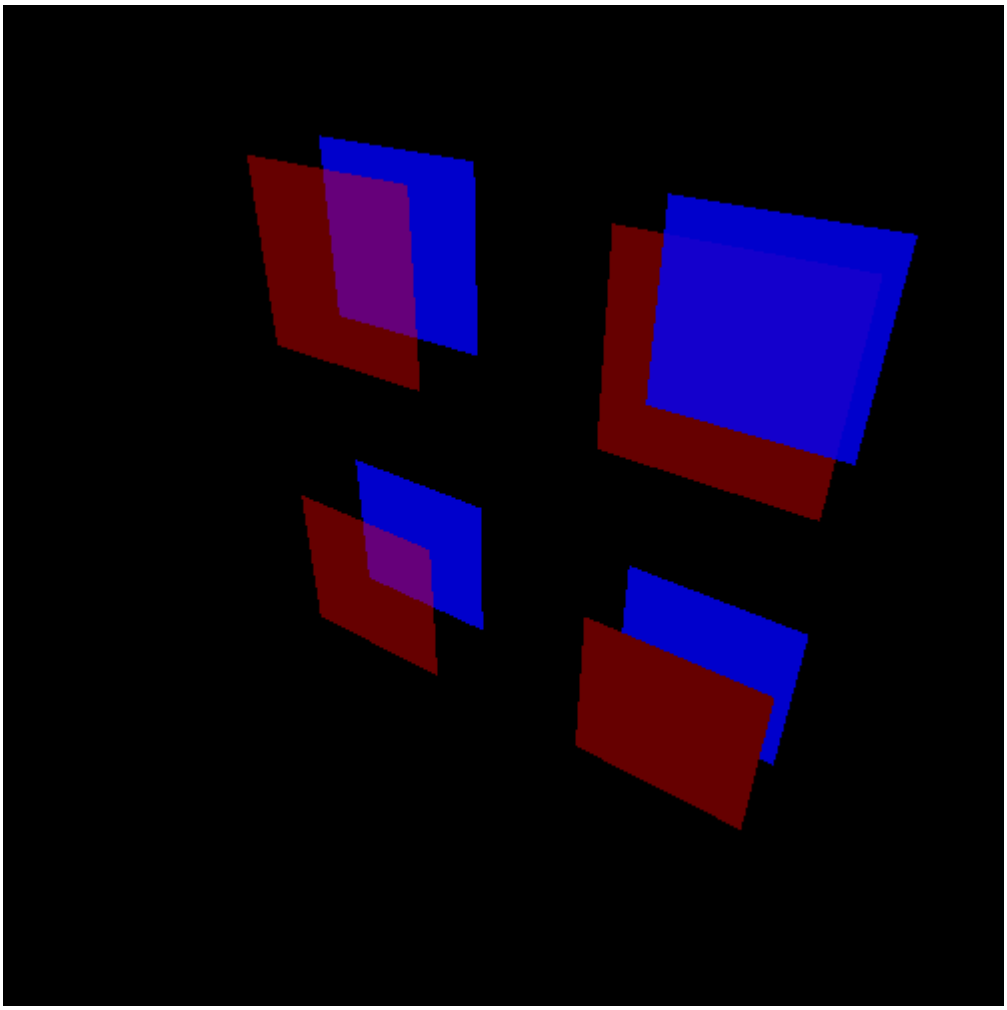
```
    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glLoadIdentity();
    w = 0;
    for (p = text1; *p; p++) {
        w += glutBitmapWidth(GLUT_BITMAP_TIMES_ROMAN_24, *p);
    }
    glColor3d(1.0, 1.0, 1.0);
    glRasterPos2d(0.0 - (double)w / (double)winw * 2.0 / 2.0, -0.8);
    for (p = text1; *p; p++) {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *p);
    }
    glMatrixMode(GL_PROJECTION);
    glPopMatrix();
    glMatrixMode(GL_MODELVIEW);
    glPopMatrix();

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glLoadIdentity();
    w = 0;
    for (p = text2; *p; p++) {
        w += glutBitmapWidth(GLUT_BITMAP_TIMES_ROMAN_24, *p);
    }
    glColor3d(0.0, 0.0, 1.0);
    glRasterPos2d((cx - w / 2) / (double)winw * 2.0 - 1.0, (cy / (double)winh * 2.0 - 1.0) - 0.07);
    for (p = text2; *p; p++) {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *p);
    }
    glMatrixMode(GL_PROJECTION);
    glPopMatrix();
    glMatrixMode(GL_MODELVIEW);
    glPopMatrix();

    glutSwapBuffers();
}
```

サンプルソースコード

```
void myTimer(int value)
{
    if (value == 1) {
        dir += 5.0;
        angle += 0.01;

        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, width, height);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

半透明

- ここで示しているような半透明の処理をアルファブレンディングと言います
- glDisable(GL_DEPTH_TEST)の状態で，奥にある図形から手前にある図形に向かって順番に描画します



サンプルソースコード

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;

void myPlane()
{
    glBegin(GL_QUADS);
    glVertex3d(-1.0, 0.0, -1.0);
    glVertex3d(1.0, 0.0, -1.0);
    glVertex3d(1.0, 0.0, 1.0);
    glVertex3d(-1.0, 0.0, 1.0);
    glEnd();
}
```

サンプルソースコード

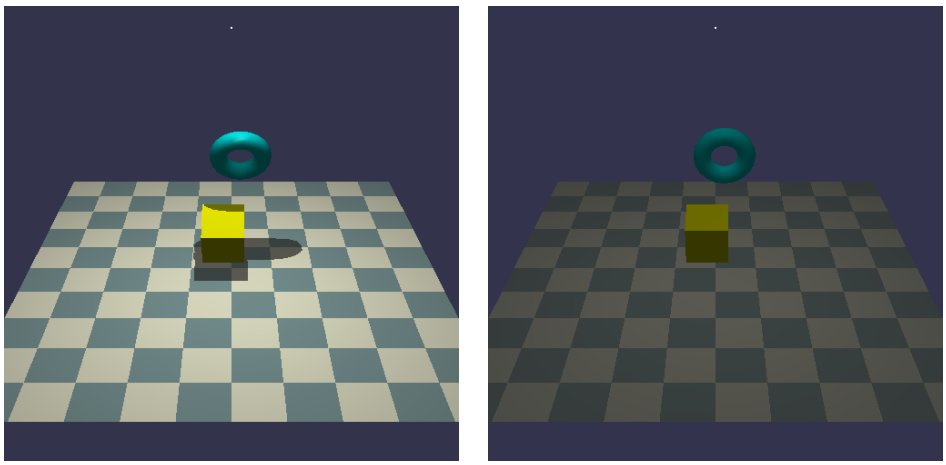
```
void myDisplay()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(4.0, -6.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_BLEND);

    glDisable(GL_DEPTH_TEST);

    glPushMatrix();
    glTranslated(-2.0, 1.0, 2.0);
    glColor4d(0.0, 0.0, 1.0, 0.8);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-2.0, 0.0, 2.0);
    glColor4d(1.0, 0.0, 0.0, 0.4);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 0.0, 2.0);
    glColor4d(1.0, 0.0, 0.0, 0.4);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 1.0, 2.0);
    glColor4d(0.0, 0.0, 1.0, 0.8);
    myPlane();
    glPopMatrix();
    glEnable(GL_DEPTH_TEST);

    glPushMatrix();
    glTranslated(-2.0, 1.0, -2.0);
    glColor4d(0.0, 0.0, 1.0, 0.8);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-2.0, 0.0, -2.0);
    glColor4d(1.0, 0.0, 0.0, 0.4);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 0.0, -2.0);
    glColor4d(1.0, 0.0, 0.0, 0.4);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 1.0, -2.0);
    glColor4d(0.0, 0.0, 1.0, 0.8);
    myPlane();
    glPopMatrix();

    glDisable(GL_BLEND);
    glutSwapBuffers();
}
```

サンプルソースコード

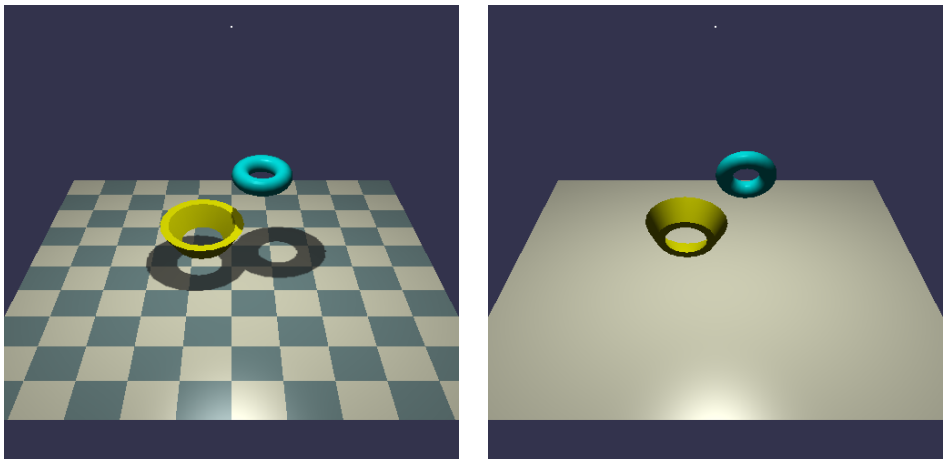
```
void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glutKeyboardFunc(myKeyboard);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

影

- 陰をシェード，影をシャドウと言います
- シェーディングはglEnable(GL_LIGHTING)によりOpenGLが勝手にやってくれます
- シャドウイングはプログラマ側で実装しないといけません
- シャドウを計算する方法の一つとしてシャドウマッピング法というものがあります
- 酒井幸市「OpenGL+GLSLによる3D-CGアニメーション」（工学社）にサンプルプログラムがあります
- このサンプルプログラムの実行結果は以下の通り
- 左の画像はあるパソコンで動作させて，正しく影が計算された例です
- 右の画像は別のパソコンで動作させて，影の計算に失敗した例です
- 動作環境によってはうまくいかないので，今回はやらないほうがいいでしょう



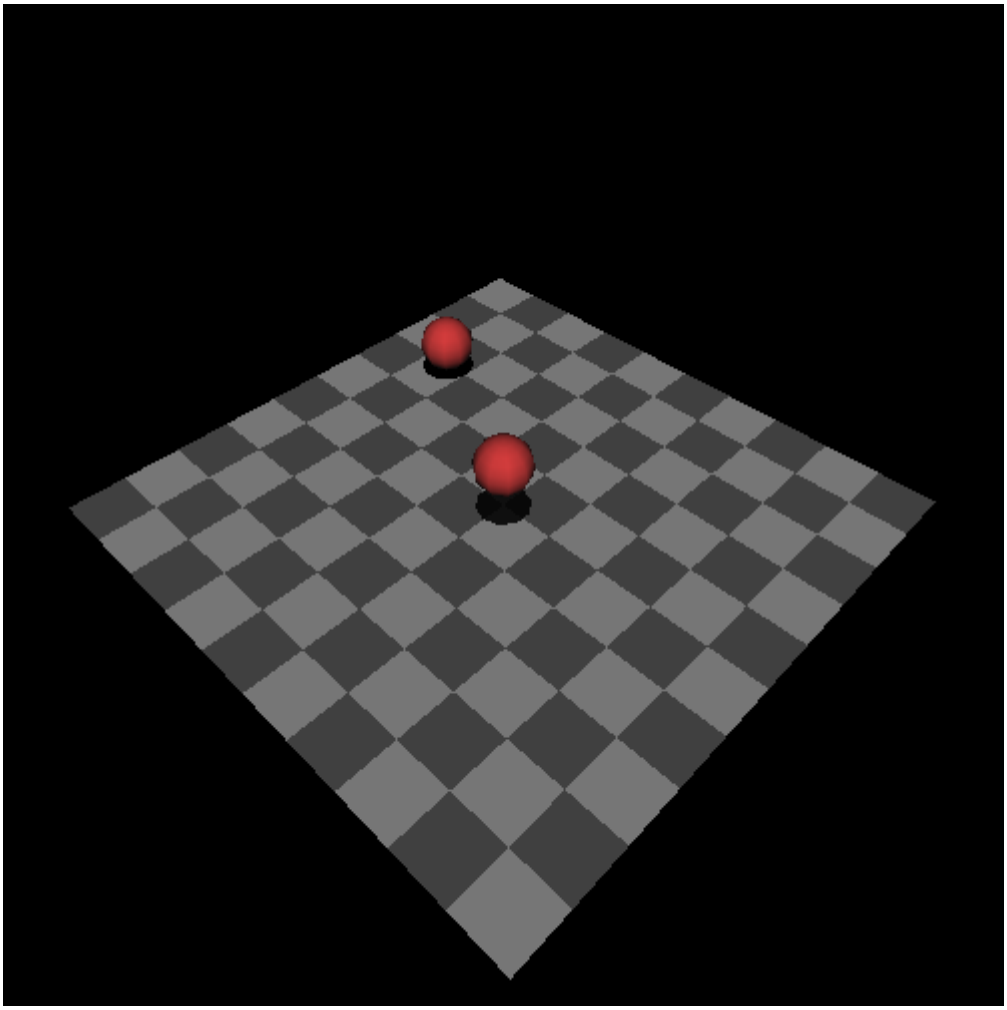
GLSLを使わないサンプルプログラムの動作結果



GLSLを使ったサンプルプログラムの動作結果

影

- いっそのこと, 影を円で表現してしまう, というのでいいのではないか



サンプルソースコード

```
#include <stdlib.h>
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 50;
double dir = 0.0;
double angle = 0.0;
double height = 0.0;
const double heightvelocityinitial = 1.0;
double heightvelocity = heightvelocityinitial;
const double heightacceleration = -0.1;

void myDisk(double r, int n)
{
    int i;
    double a;

    glBegin(GL_POLYGON);
    glNormal3d(0.0, 0.0, 1.0);
    for (i = 0; i < n; i++) {
        a = (double)i / (double)n * 2.0 * M_PI;
        glVertex3d(r * cos(a), r * sin(a), 0);
    }
    glEnd();
}

void myGround()
{
    float ground[2][4] = {
        {0.6f, 0.6f, 0.6f, 1.0f},
        {0.3f, 0.3f, 0.3f, 1.0f}
    };
    int i, j;

    glBegin(GL_QUADS);
    glNormal3d(0.0, 0.0, 1.0);
    for (j = -5; j < 5; j++) {
        for (i = -5; i < 5; i++) {
            glMaterialfv(GL_FRONT, GL_DIFFUSE, ground[abs(i + j) % 2]);
            glVertex3d((double)i, (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)j, 0.0);
            glVertex3d((double)(i + 1), (double)(j + 1), 0.0);
            glVertex3d((double)i, (double)(j + 1), 0.0);
        }
    }
    glEnd();
}
```

サンプルソースコード

```
void myDisplay()
{
    const double RADIUS = 0.5;
    float red[4] = { 0.8f, 0.2f, 0.2f, 1.0f };
    double alpha = (height > 3.0) ? 0.0 : ((3.0 - height) / 3.0);
    double eyex = 10.0 * cos(angle);
    double eyey = 10.0 * sin(angle);
    double eyez = 10.0;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    myGround();

    glPushMatrix();

    glPushMatrix();
    glTranslated(0.0, 0.0, 0.01);
    glDisable(GL_LIGHTING);
    glEnable(GL_BLEND);
    glDisable(GL_DEPTH_TEST);
    glColor4d(0.0, 0.0, 0.0, alpha);
    myDisk(RADIUS * (alpha * 0.5 + 0.5), 10);
    glEnable(GL_DEPTH_TEST);
    glDisable(GL_BLEND);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glTranslated(0.0, 0.0, height + RADIUS);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, red);
    glutSolidSphere(RADIUS, 10, 10);

    glPopMatrix();

    glPushMatrix();

    glRotated(dir, 0.0, 0.0, 1.0);
    glTranslated(4.0, 0.0, 0.0);

    glPushMatrix();
    glTranslated(0.0, 0.0, 0.01);
    glDisable(GL_LIGHTING);
    glColor3d(0.0, 0.0, 0.0);
    myDisk(RADIUS, 10);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glTranslated(0.0, 0.0, RADIUS);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, red);
    glutSolidSphere(RADIUS, 10, 10);

    glPopMatrix();

    glutSwapBuffers();
}
```

サンプルソースコード

```
void myTimer(int value)
{
    if (value == 1) {
        dir += 5.0;
        angle += 0.01;
        height += heightvelocity;
        heightvelocity += heightacceleration;
        if (height < 0.0) {
            height = heightvelocityinitial;
            heightvelocity = heightvelocityinitial + heightacceleration;
        }

        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

キャラクタ身体動作アニメーション

- ロボットのような形状と動き程度でいいのであれば，「歩行動作」の課題を参考にして実装できる
- 本格的なキャラクターアニメーションをOpenGLで実装するぐらいなら，Unityなどのゲーム開発環境を使ったほうがいい
- OpenGLで実装する場合は以下のページが参考になるかも
  http://www.cg.ces.kyutech.ac.jp/lecture/cg2/

ジョイスティック・ゲームパッド

- ジョイスティックやゲームパッドに本格的に対応するなら各自で調べて実装してください
- ほとんどのジョイスティックやゲームパッドは，それぞれのボタンや十字キーがキーボードの文字と対応していることが多いです．例え製品側が対応していなくても，フリーソフトを使えば対応させることが可能です．例えば，ゲームパッドの十字キーの上を押したらキーボードのWを押したことと同じになる，とか，左ならA，右ならD，下ならS，など．そういったキーバインド設定は，製品側の機能や，フリーソフトの機能で，任意のキーに割り当てることが可能なことが多いです．
- つまり，とりあえずはglutKeyboardFuncを使えばジョイスティックやゲームパッドでも操作できます

音楽の再生

- OpenGLには音楽ファイルを再生する機能はありません

Win32 API

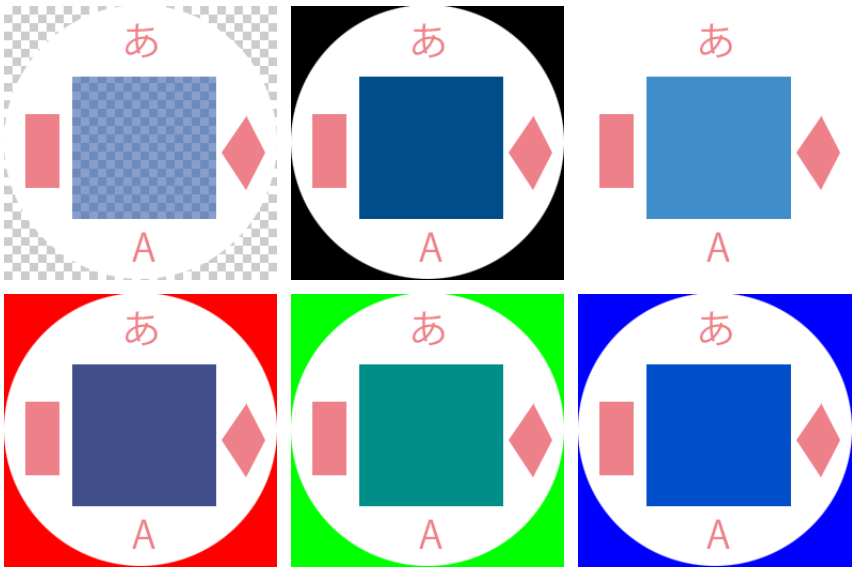
- Windowsの標準機能でwavファイルを再生するサンプルプログラムを載せます
- bgm.wavを無限ループで再生して，スペースキーを押すたびにbell.wavを再生するプログラムです
    - ファイルは配布しませんので，各自で好きな.wavファイルを用意してください

サンプルソースコード

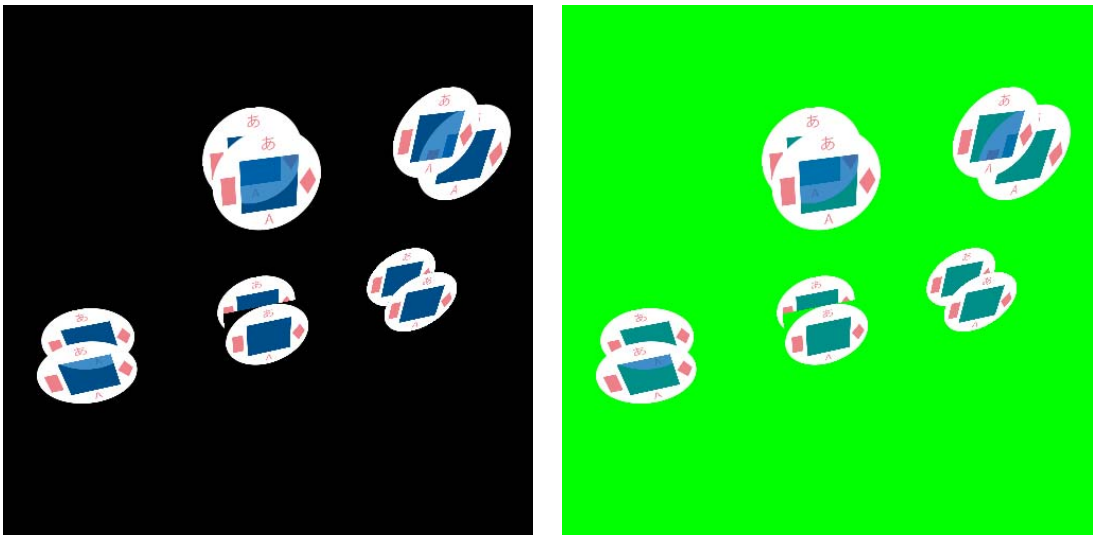
```
#include <stdio.h>
#include <conio.h>
#include <Windows.h>
#pragma comment(lib, "winmm.lib")

int main(int argc, char* argv[])
{
    char bgmfilename[] = "bgm.wav";
    char bellfilename[] = "bell.wav";
    char ch;

    PlaySound(bgmfilename, NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);

    MCI_OPEN_PARMS bellopen;
    bellopen.lpstrDeviceType = "WaveAudio";
    bellopen.lpstrElementName = bellfilename;
    mciSendCommand(NULL, MCI_OPEN, MCI_OPEN_TYPE | MCI_OPEN_ELEMENT, (DWORD)&bellopen);

    printf("[ESC] End, [SPACE] Bell¥n");

    for (;;) {
        ch = _getch();
        if (ch == 0x1B) break;
        if (ch == ' ') {
            mciSendCommand(bellopen.wDeviceID, MCI_SEEK, MCI_SEEK_TO_START, 0);
            mciSendCommand(bellopen.wDeviceID, MCI_PLAY, 0, 0);
        }
    }
    mciSendCommand(bellopen.wDeviceID, MCI_CLOSE, 0, 0);
    PlaySound(NULL, NULL, 0);

    printf("Press any key¥n");
    _getch();

    return 0;
}
```

半透明テクスチャ

- 描画する順番に注意する(デプステストをオフにした状態)



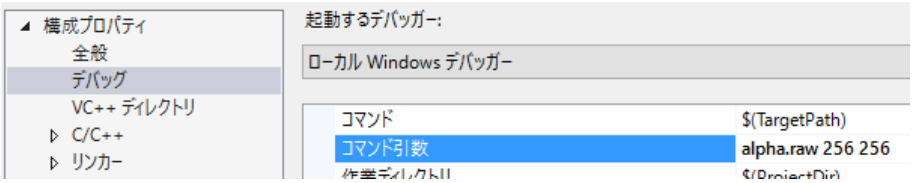
幅256, 高さ256, ヘッダなし, 8bit, rawデータ
RGBAがスキャンラインオーダーで左上から右下に並べられたファイル1枚
(上図は半透明の様子が分かりやすいように異なる背景で6枚載せているだけ)



実行はcg2.exe alpha.raw 256 256のように引数を3つ与える

複数の引数

- コマンドライン引数に複数の項目を指定するときはスペースで区切る



サンプルソースコード

```
#pragma warning(disable:4996)
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;
int imagew, imageh;
unsigned char* imagedata;

void myPlane()
{
    glBegin(GL_QUADS);
    glTexCoord2d(0.0, 1.0);
    glVertex3d(-1.0, 0.0, -1.0);
    glTexCoord2d(1.0, 1.0);
    glVertex3d(1.0, 0.0, -1.0);
    glTexCoord2d(1.0, 0.0);
    glVertex3d(1.0, 0.0, 1.0);
    glTexCoord2d(0.0, 0.0);
    glVertex3d(-1.0, 0.0, 1.0);
    glEnd();
}

void myDisplay()
{
    unsigned int texName;

    glGenTextures(1, &texName);
    glBindTexture(GL_TEXTURE_2D, texName);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imagew, imageh, 0, GL_RGBA, GL_UNSIGNED_BYTE, imagedata);
    glEnable(GL_TEXTURE_2D);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(-4.0, -7.0, 6.0, -2.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glEnable(GL_BLEND);
    glDisable(GL_DEPTH_TEST);
```

```
    glColor4d(1.0, 1.0, 1.0, 1.0);
    glPushMatrix();
    glTranslated(-2.0, 1.0, 2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-2.0, 0.0, 2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 0.0, 2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 1.0, 2.0);
    myPlane();
    glPopMatrix();

    glEnable(GL_DEPTH_TEST);

    glPushMatrix();
    glTranslated(-6.0, 1.0, -2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-6.0, 0.0, -2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-2.0, 0.0, -2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(-2.0, 1.0, -2.0);
    myPlane();
    glPopMatrix();

    glAlphaFunc(GL_GREATER, 0.5f);
    glEnable(GL_ALPHA_TEST);

    glPushMatrix();
    glTranslated(2.0, 0.0, -2.0);
    myPlane();
    glPopMatrix();

    glPushMatrix();
    glTranslated(2.0, 1.0, -2.0);
    myPlane();
    glPopMatrix();

    glAlphaFunc(GL_ALWAYS, 0.0f);
    glDisable(GL_ALPHA_TEST);

    glDisable(GL_BLEND);
    glDeleteTextures(1, &texName);
    glDisable(GL_TEXTURE_2D);

    glutSwapBuffers();
}
```

サンプルソースコード

```
int readRaw(char filename[], int width, int height)
{
    FILE* fp;
    int x, y;
    int i;

    fp = fopen(filename, "rb");
    if (fp == NULL) {
        printf("file '%s' not exist¥n", filename);
        return 1;
    }

    imagew = width;
    imageh = height;

    if (imagedata != NULL) free(imagedata);
    imagedata = (unsigned char*)malloc(sizeof(unsigned char) * imagew * imageh * 4);
    if (imagedata == NULL) {
        printf("memory error¥n");
        return 1;
    }

    for (y = 0; y < imageh; y++) {
        for (x = 0; x < imagew; x++) {
            if (feof(fp)) {
                printf("lack of data¥n");
                return 1;
            }
            i = y * imagew * 4 + x * 4;
            fread(&imagedata[i + 0], 1, 1, fp);
            fread(&imagedata[i + 1], 1, 1, fp);
            fread(&imagedata[i + 2], 1, 1, fp);
            fread(&imagedata[i + 3], 1, 1, fp);
        }
    }

    fclose(fp);
    return 0;
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}
```

サンプルソースコード

```
int main(int argc, char* argv[])
{
    char line[256];
    imagedata = NULL;
    if (argc <= 3) {
        printf("filename not specified¥n");
        printf("push any key and push enter¥n");
        scanf("%s", line);
        return 1;
    }
    if (readRaw(argv[1], atoi(argv[2]), atoi(argv[3]))) {
        printf("push any key and push enter¥n");
        scanf("%s", line);
        return 1;
    }

    glutInit(&argc, argv);
    myInit(argv[0]);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glutKeyboardFunc(myKeyboard);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

背景

- 背景にテクスチャを貼りたい場合
- サンプルプログラムはcg2.exe background.bmpのように実行している

サンプルソースコード

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 50;
double angle = 0.0;
int imagew, imageh;
unsigned char* imagedata;
unsigned int texName;
```

サンプルソースコード

```
int readbmp(char* filename, unsigned char** image, int* sizex, int* sizey)
{
    FILE* fp;
    int row;
    unsigned char* ras;
    char header1, header2;
    int imagew, imageh;
    unsigned short bitcount;

    fopen_s(&fp, filename, "rb");
    if (fp == NULL) {
        fprintf(stderr, "ファイル%sを開けません¥n", filename);
        return 1;
    }

    fread(&header1, 1, 1, fp);
    fread(&header2, 1, 1, fp);
    if (header1 != 'B' || header2 != 'M') {
        fprintf(stderr, "ファイル%sはBMPファイルではありません¥n", filename);
        return 1;
    }
    fseek(fp, 12, 1);

    fseek(fp, 4, 1);
    fread(&imagew, 4, 1, fp);
    fread(&imageh, 4, 1, fp);
    *sizex = imagew;
    *sizey = imageh;

    fseek(fp, 2, 1);
    fread(&bitcount, 2, 1, fp);
    if (bitcount != 24) {
        fprintf(stderr, "ファイル%sは24ビットフルカラーではありません¥n", filename);
        return 1;
    }
    fseek(fp, 24, 1);

    row = imagew * 3;
    row = ((row + 3) / 4) * 4;
    ras = (unsigned char*)malloc(sizeof(unsigned char) * row * imageh);
    if (ras == NULL) {
        fprintf(stderr, "メモリを確保できません¥n");
        return 1;
    }
    fread(ras, sizeof(unsigned char), row * imageh, fp);

    *image = (unsigned char*)malloc(sizeof(unsigned char) * imagew * imageh * 3);
    if (*image == NULL) {
        fprintf(stderr, "メモリを確保できません¥n");
        return 1;
    }
    for (int y = 0; y < imageh; y++) {
        for (int x = 0; x < imagew; x++) {
            int cr, cg, cb;
            int i;
            i = (imageh - 1 - y) * row + x * 3;
            cb = ras[i + 0];
            cg = ras[i + 1];
            cr = ras[i + 2];
            (*image)[y * imagew * 3 + x * 3 + 0] = cr;
            (*image)[y * imagew * 3 + x * 3 + 1] = cg;
            (*image)[y * imagew * 3 + x * 3 + 2] = cb;
        }
    }
    free(ras);
    fclose(fp);
    return 0;
}
```

サンプルソースコード

```
int readTexture(int argc, char* argv[]) {
    if (argc == 1) {
        printf("file name not specified¥n");
        return 1;
    }

    glGenTextures(1, &texName);
    imagedata = NULL;
    glBindTexture(GL_TEXTURE_2D, texName);

    if (readbmp(argv[1], &imagedata, &imagew, &imageh)) return 1;
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imagew, imageh, 0, GL_RGB, GL_UNSIGNED_BYTE, imagedata);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

    return 0;
}

void myPlane(double length) {
    double l = length * 0.5;

    glBegin(GL_QUADS);
    glNormal3d(0.0, 0.0, 1.0);
    glTexCoord2d(0.0, 0.0);
    glVertex3d(-l, l, 0.0);
    glTexCoord2d(0.0, 1.0);
    glVertex3d(-l, -l, 0.0);
    glTexCoord2d(1.0, 1.0);
    glVertex3d(l, -l, 0.0);
    glTexCoord2d(1.0, 0.0);
    glVertex3d(l, l, 0.0);
    glEnd();
}
```

サンプルソースコード

```
void myDisplay()
{
    double eyex = 10.0 * cos(angle);
    double eyey = 10.0 * sin(angle);
    double eyez = 10.0;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glEnable(GL_TEXTURE_2D);

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glLoadIdentity();

    glBindTexture(GL_TEXTURE_2D, texName);
    myPlane(2.0);
    glBindTexture(GL_TEXTURE_2D, 0);

    glMatrixMode(GL_PROJECTION);
    glPopMatrix();
    glMatrixMode(GL_MODELVIEW);
    glPopMatrix();

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    myPlane(10.0);

    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHTING);
    glDisable(GL_DEPTH_TEST);
    glDisable(GL_TEXTURE_2D);

    glutSwapBuffers();
}

void myTimer(int value)
{
    if (value == 1) {
        angle += 0.01;
        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}
```

サンプルソースコード

```
void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
    if (readTexture(argc, argv)) {
        printf("push any key¥n");
        _getch();
        return 1;
    }
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

複数のテクスチャ

- 複数のテクスチャを使いたい場合
- サンプルプログラムはcg2.exe ichiko.bmp dice.bmpのように実行している

サンプルソースコード

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 50;
double dir = 0.0;
double angle = 0.0;
const int TEXNUM = 2;
int imagew[TEXNUM], imageh[TEXNUM];
unsigned char* imagedata[TEXNUM];
unsigned int texName[TEXNUM];
```

サンプルソースコード

```
int readbmp(char* filename, unsigned char** image, int* sizex, int* sizey)
{
    FILE* fp;
    int row;
    unsigned char* ras;
    char header1, header2;
    int imagew, imageh;
    unsigned short bitcount;

    fopen_s(&fp, filename, "rb");
    if (fp == NULL) {
        fprintf(stderr, "ファイル%sを開けません¥n", filename);
        return 1;
    }

    fread(&header1, 1, 1, fp);
    fread(&header2, 1, 1, fp);
    if (header1 != 'B' || header2 != 'M') {
        fprintf(stderr, "ファイル%sはBMPファイルではありません¥n", filename);
        return 1;
    }
    fseek(fp, 12, 1);

    fseek(fp, 4, 1);
    fread(&imagew, 4, 1, fp);
    fread(&imageh, 4, 1, fp);
    *sizex = imagew;
    *sizey = imageh;

    fseek(fp, 2, 1);
    fread(&bitcount, 2, 1, fp);
    if (bitcount != 24) {
        fprintf(stderr, "ファイル%sは24ビットフルカラーではありません¥n", filename);
        return 1;
    }
    fseek(fp, 24, 1);

    row = imagew * 3;
    row = ((row + 3) / 4) * 4;
    ras = (unsigned char*)malloc(sizeof(unsigned char) * row * imageh);
    if (ras == NULL) {
        fprintf(stderr, "メモリを確保できません¥n");
        return 1;
    }
    fread(ras, sizeof(unsigned char), row * imageh, fp);

    *image = (unsigned char*)malloc(sizeof(unsigned char) * imagew * imageh * 3);
    if (*image == NULL) {
        fprintf(stderr, "メモリを確保できません¥n");
        return 1;
    }
    for (int y = 0; y < imageh; y++) {
        for (int x = 0; x < imagew; x++) {
            int cr, cg, cb;
            int i;
            i = (imageh - 1 - y) * row + x * 3;
            cb = ras[i + 0];
            cg = ras[i + 1];
            cr = ras[i + 2];
            (*image)[y * imagew * 3 + x * 3 + 0] = cr;
            (*image)[y * imagew * 3 + x * 3 + 1] = cg;
            (*image)[y * imagew * 3 + x * 3 + 2] = cb;
        }
    }
    free(ras);
    fclose(fp);
    return 0;
}
```

サンプルソースコード

```
int readTexture(int argc, char* argv[]) {
    int n;
    int index;
    char* args;

    glGenTextures(TEXNUM, texName);

    for (index = 0; index < TEXNUM; index++) {
        imagedata[index] = NULL;
        glBindTexture(GL_TEXTURE_2D, texName[index]);

        if (index + 1 >= argc) {
            printf("%d files specified, while %d expected¥n", index, TEXNUM);
            return 1;
        }

        args = argv[index + 1];
        n = strlen(args);

        if (n < 4) {
            printf("invalid file name '%s'¥n", args);
            return 1;
        }

        if (strcmp(&args[n - 4], ".bmp") == 0 || strcmp(&args[n - 4], ".BMP") == 0) {
            if (readbmp(args, &imagedata[index], &imagew[index], &imageh[index])) return 1;
            glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imagew[index], imageh[index], 0, GL_RGB, GL_UNSIGNED_BYTE,
imagedata[index]);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        }
        else {
            printf("file '%s' not supported¥n", args);
            return 1;
        }
    }

    return 0;
}
```

サンプルソースコード

```
void myCube(double length)
{
    double l = length * 0.5;
    double vlist[8][3] = {
        { l, l, l },
        { -l, l, l },
        { -l, -l, l },
        { l, -l, l },
        { l, l, -l },
        { -l, l, -l },
        { -l, -l, -l },
        { l, -l, -l }
    };
    double nlist[6][3] = {
        { 0.0, 0.0, 1.0 },
        { 1.0, 0.0, 0.0 },
        { 0.0, 1.0, 0.0 },
        { 0.0, -1.0, 0.0 },
        { -1.0, 0.0, 0.0 },
        { 0.0, 0.0, -1.0 }
    };
    int flist[6][4] = {
        { 2, 3, 0, 1 },
        { 3, 7, 4, 0 },
        { 0, 4, 5, 1 },
        { 2, 6, 7, 3 },
        { 1, 5, 6, 2 },
        { 7, 6, 5, 4 }
    };
    double tlist[6][4][2] = {
        {
            { 0.0, 0.5 },
            { 0.25, 0.5 },
            { 0.25, 0.25 },
            { 0.0, 0.25 }
        },
        {
            { 0.25, 0.5 },
            { 0.5, 0.5 },
            { 0.5, 0.25 },
            { 0.25, 0.25 }
        },
        {
            { 0.25, 0.25 },
            { 0.5, 0.25 },
            { 0.5, 0.0 },
            { 0.25, 0.0 }
        },
        {
            { 0.25, 0.75 },
            { 0.5, 0.75 },
            { 0.5, 0.5 },
            { 0.25, 0.5 }
        },
        {
            { 0.25, 1.0 },
            { 0.5, 1.0 },
            { 0.5, 0.75 },
            { 0.25, 0.75 }
        },
        {
            { 0.5, 0.5 },
            { 0.75, 0.5 },
            { 0.75, 0.25 },
            { 0.5, 0.25 }
        }
    };
```

サンプルソースコード

```
    int i;
    int i0, i1, i2, i3;

    glBegin(GL_QUADS);

    for (i = 0; i < 6; i++) {
        i0 = flist[i][0];
        i1 = flist[i][1];
        i2 = flist[i][2];
        i3 = flist[i][3];
        glNormal3dv(nlist[i]);
        glTexCoord2dv(tlist[i][0]);
        glVertex3dv(vlist[i0]);
        glTexCoord2dv(tlist[i][1]);
        glVertex3dv(vlist[i1]);
        glTexCoord2dv(tlist[i][2]);
        glVertex3dv(vlist[i2]);
        glTexCoord2dv(tlist[i][3]);
        glVertex3dv(vlist[i3]);
    }
    glEnd();
}

void myPlane(double length) {
    double l = length * 0.5;

    glBegin(GL_QUADS);
    glNormal3d(0.0, 0.0, 1.0);
    glTexCoord2d(0.0, 0.0);
    glVertex3d(-l, l, 0.0);
    glTexCoord2d(0.0, 1.0);
    glVertex3d(-l, -l, 0.0);
    glTexCoord2d(1.0, 1.0);
    glVertex3d(l, -l, 0.0);
    glTexCoord2d(1.0, 0.0);
    glVertex3d(l, l, 0.0);
    glEnd();
}
```

サンプルソースコード

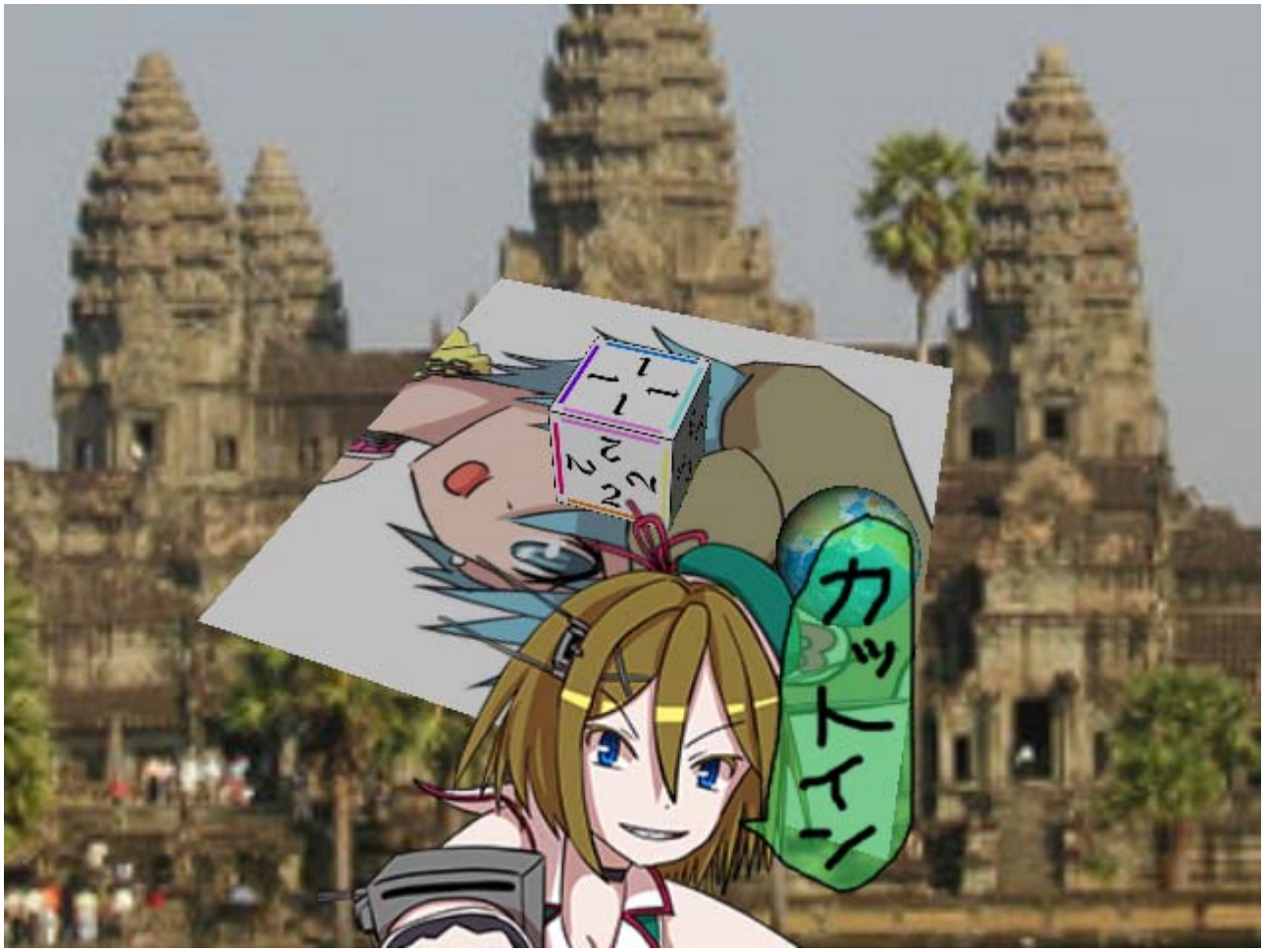
```
void myDisplay()
{
    double eyex = 10.0 * cos(angle);
    double eyey = 10.0 * sin(angle);
    double eyez = 10.0;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glEnable(GL_TEXTURE_2D);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glBindTexture(GL_TEXTURE_2D, texName[0]);
    myPlane(10.0);
    glBindTexture(GL_TEXTURE_2D, 0);

    glPushMatrix();
    glTranslated(0.0, 0.0, 1.0);
    glBindTexture(GL_TEXTURE_2D, texName[1]);
    myCube(2.0);
    glBindTexture(GL_TEXTURE_2D, 0);
    glPopMatrix();

    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHTING);
    glDisable(GL_DEPTH_TEST);
    glDisable(GL_TEXTURE_2D);

    glutSwapBuffers();
}

void myTimer(int value)
{
    if (value == 1) {
        angle += 0.01;
        glutTimerFunc(delay, myTimer, 1);
        glutPostRedisplay();
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}
```

サンプルソースコード

```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
    if (readTexture(argc, argv)) {
        printf("push any key¥n");
        _getch();
        return 1;
    }
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

前景

- α値のあるテクスチャを前景として貼りたい場合



サンプルプログラム

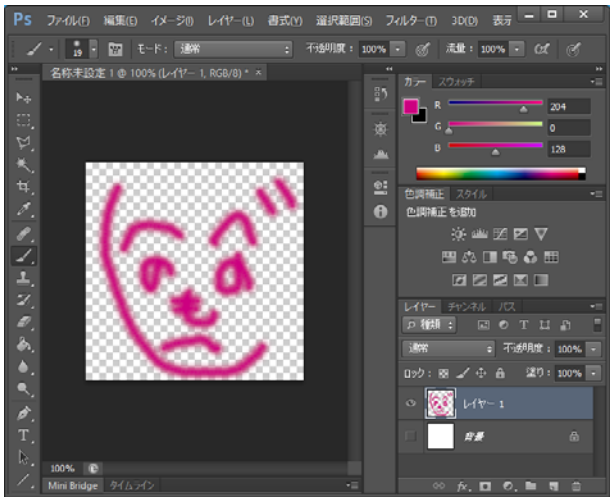
- 省略
- 私の実装では500行ぐらい

Photoshop

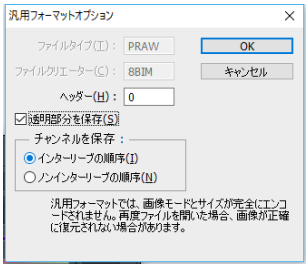
- 半透明raw画像の作り方を解説します
- Adobe Photoshopという市販ソフトで説明します

手順

- Photoshopを起動，256×256などのサイズの新規画像を作成，絵を描く



- メニューの[ファイル]-[別名で保存]をクリック
- [ファイル形式]から[汎用フォーマット(*.RAW)]を選択し，ファイル名を入力し[保存]
- [ヘッダー]は[0]，[透明部分を保存]は[チェックあり]，[チャンネルを保存]は[インターリーブの順序]



- あとは実装したプログラムで読み込んで表示して，正しく読み込めたか確認すればOK