２００７年１２月１９日

第24回　OpenGL によるグラフィックス（13）

情報処理論（応用）

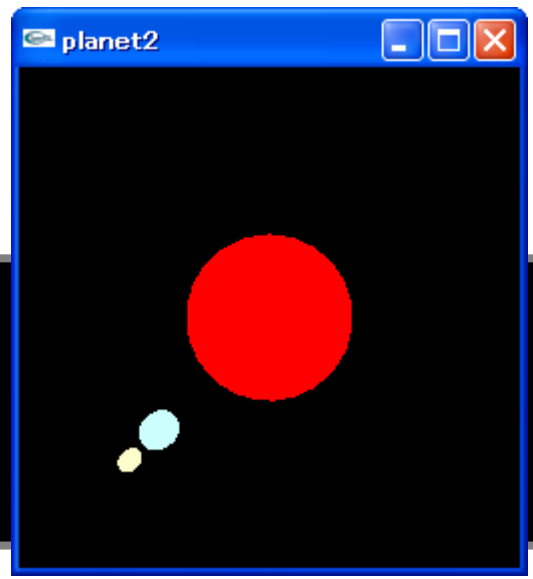
松山大学 経営学部
檀　裕也

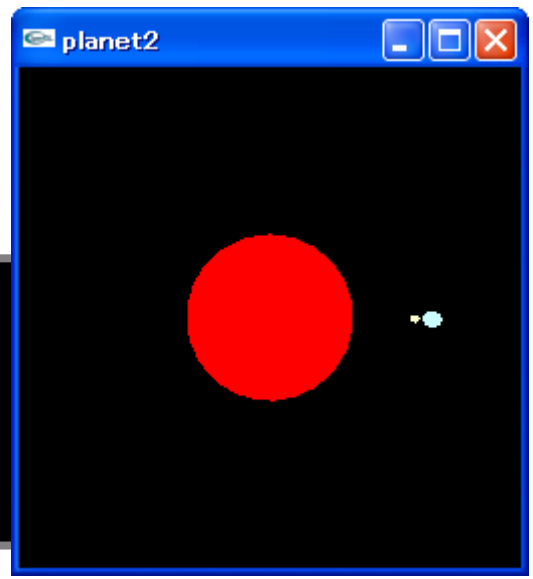
http://www.cc.matsuyama-u.ac.jp/~dan/education/application/

前回の実習課題

- 太陽の周りを地球が公転し、さらに地球の周りを月が公転する様子をＣＧで表現せよ。
  - 宛先：dan@cc.matsuyama-u.ac.jp
  - 件名：CG課題#12

```
C:\>planet2.exe
```

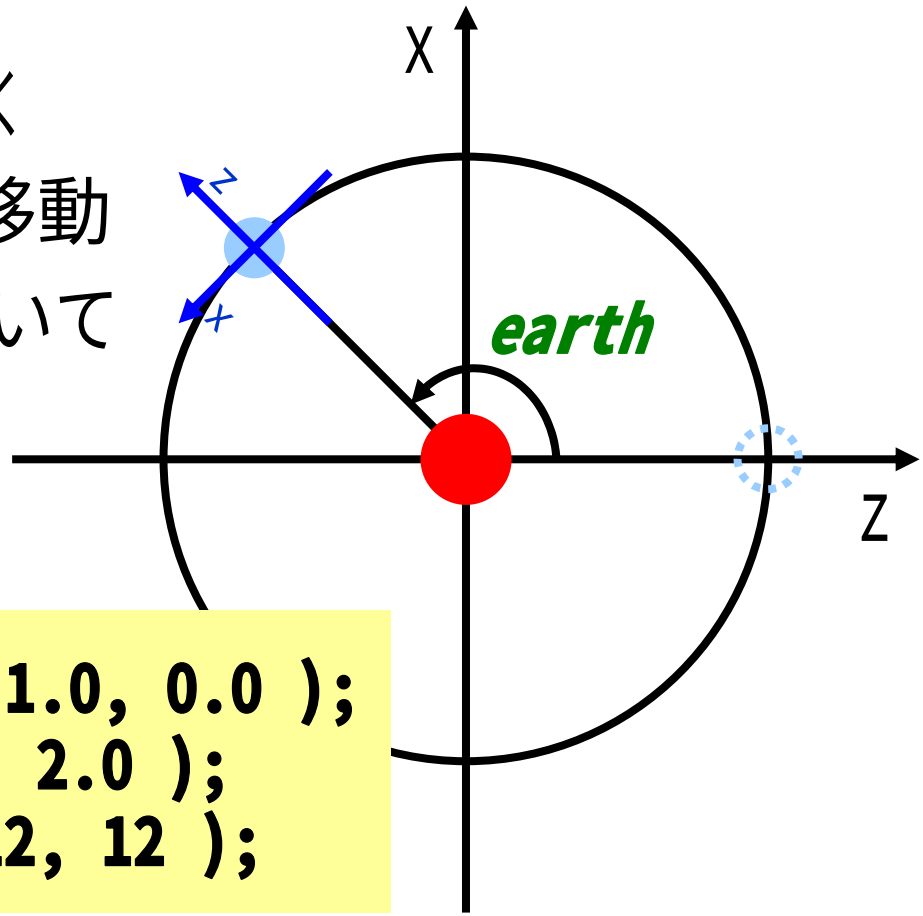




考えかた　(再)

- 地球の描画
  - 原点に球面を描く
  - z軸方向に平行移動
  - y軸のまわりについて回転させる

ローカル座標系

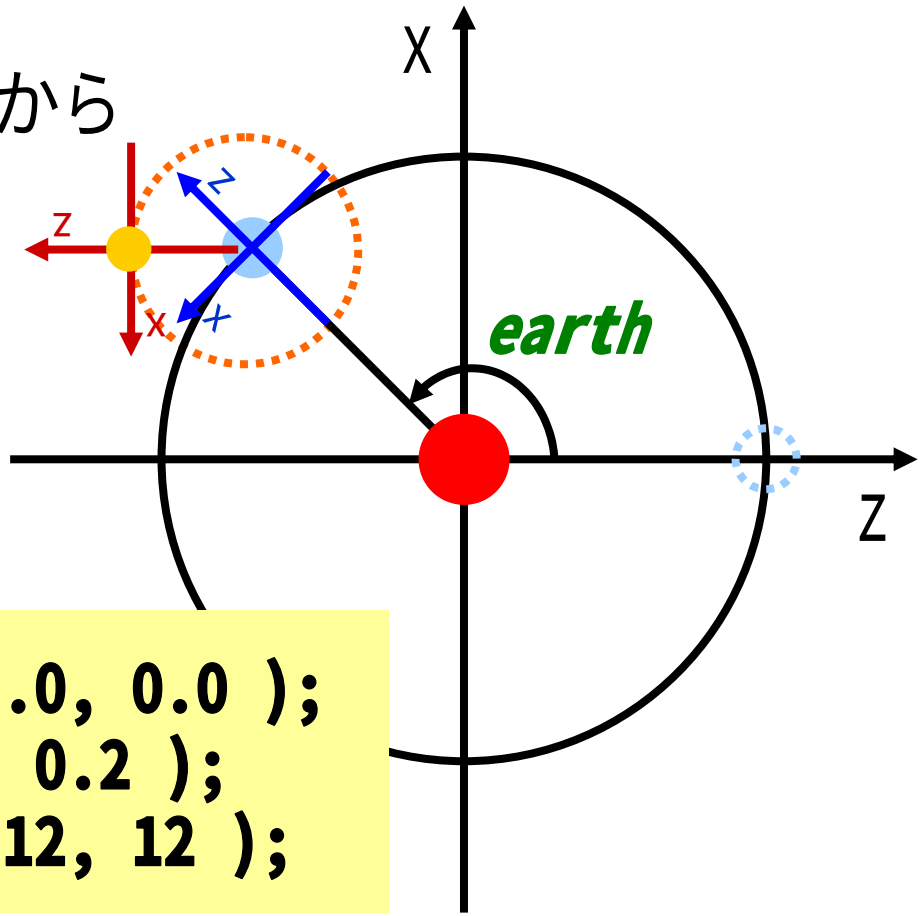



```
glRotatef( earth, 0.0, 1.0, 0.0 );
glTranslatef( 0.0, 0.0, 2.0 );
glutSolidSphere( 0.1, 12, 12 );
```

考えかた

- 月の描画
  - 地球の描画位置から相対的な場所に月を描く

ローカル座標系




```
glRotatef( moon, 0.0, 1.0, 0.0 );
glTranslatef( 0.0, 0.0, 0.2 );
glutSolidSphere( 0.08, 12, 12 );
```

解答例（1/5）

planet2.c

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

double ey = 0.0;
double ez = 5.0;
double theta = 0.3;
double earth = 0.0;
double moon = 0.0;

void display( void )
{
        glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

        glLoadIdentity();
        gluLookAt( ez * sin( theta ), ey, ez * cos( theta ),
                   0.0, 0.0, 0.0, 0.0, 1.0, 0.0 );

        glEnable( GL_DEPTH_TEST );
```

解答例（2/5）

```
        glColor3f( 1.0, 0.0, 0.0 );
        glutSolidSphere( 1.0, 24, 24 );

        glRotatef( earth, 0.0, 1.0, 0.0 );
        glTranslatef( 0.0, 0.0, 2.0 );
        glColor3f( 0.8, 1.0, 1.0 );
        glutSolidSphere( 0.1, 12, 12 );

        glRotatef( moon, 0.0, 1.0, 0.0 );
        glTranslatef( 0.0, 0.0, 0.2 );
        glColor3f( 1.0, 1.0, 0.8 );
        glutSolidSphere( 0.08, 12, 12 );

        glDisable( GL_DEPTH_TEST );

        glFlush();
}

void myKeyBoard( unsigned char key, int x, int y )
{
        switch( key ){
        case 0x1B:
```

解答例（3/5）

```
            exit( 0 );
        case 'y':
            ey -= 1.0;
            break;
        case 'Y':
            ey += 1.0;
            break;
        case 'z':
            ez -= 0.2;
            break;
        case 'Z':
            ez += 0.2;
            break;
        case 'r':
            theta -= 0.1;
            break;
        case 'R':
            theta += 0.1;
            break;
        case 'e':
            earth -= 5.0;
            break;
```

解答例（4/5）

```
        case 'E':
                earth += 5.0;
                break;
        case 'm':
                moon -= 5.0;
                break;
        case 'M':
                moon += 5.0;
                break;
        }
        glutPostRedisplay();
}

int main( int argc, char *argv[] )
{
        glutInit( &argc, argv );
        glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH );
        glutInitWindowSize( 250, 250 );
        glutCreateWindow( argv[0] );

        glClearColor( 0.0, 0.0, 0.0, 0.0 );
        glMatrixMode( GL_PROJECTION );
```

解答例（5/5）

```
        glLoadIdentity();
        glFrustum( -1.0, 1.0, -1.0, 1.0, 1.0, 20.0 );
        glMatrixMode( GL_MODELVIEW );

        glutDisplayFunc( display );
        glutKeyboardFunc( myKeyBoard );
        glutMainLoop();

        return 0;
}
```

今回の授業内容

- OpenGL によるグラフィックス（13）
  - アニメーションの技法

アニメーション

- ひとコマひとコマの静止画を連続してみせることで、動きを表す表現手法
  - 物体の位置を少しずつ変化させる

実装方法

- ＣＰＵの待ち時間にパラメータを変化させ、描画処理を実行する。

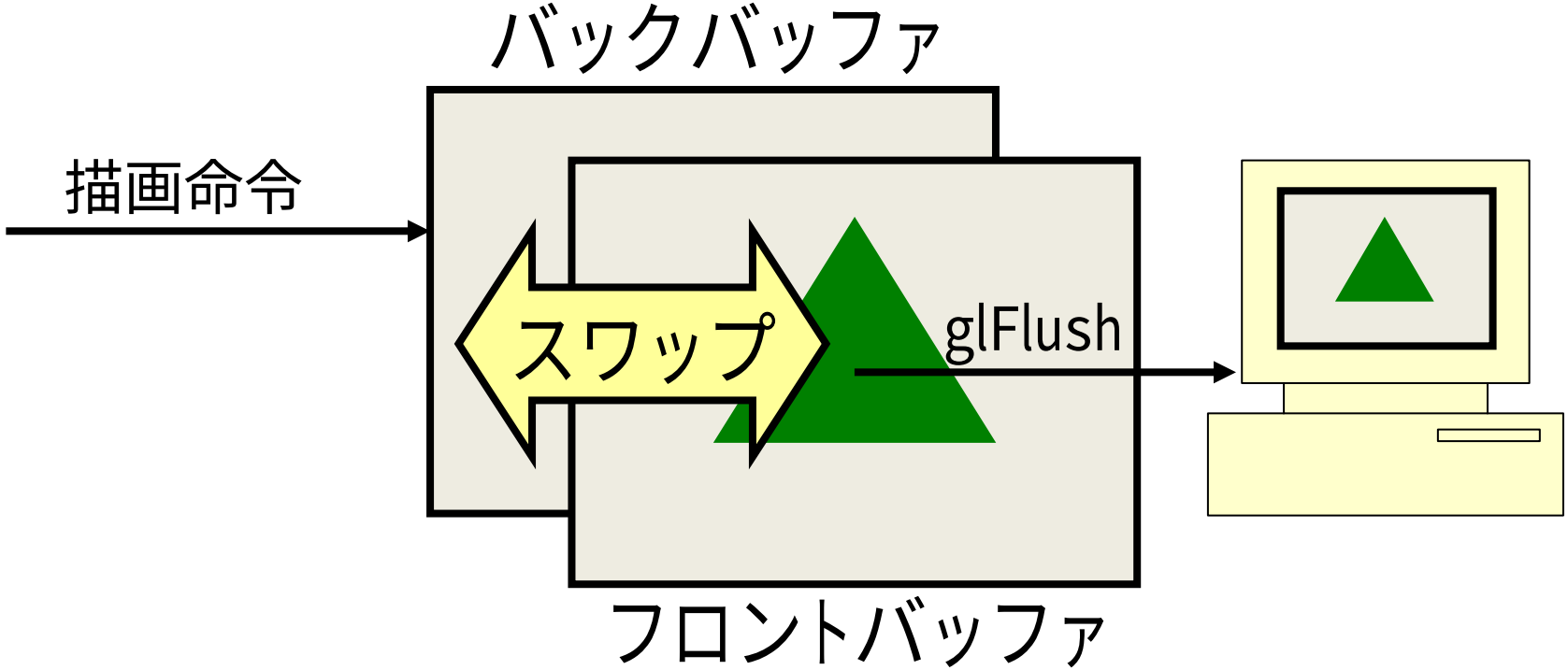
```
void idle( void )
{
        moon += 5.0;
        earth += 5.0;
        glutPostRedisplay();
}
```

コールバック関数を設定する

```
        glutIdleFunc( idle );
```

ダブルバッファ

- 2つのフレームバッファを使い、画面のちらつきを低減する。



実装方法

- ダブルバッファの宣言

```
glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
```

- フレームバッファのスワップ処理

```
glutSwapBuffers();
```

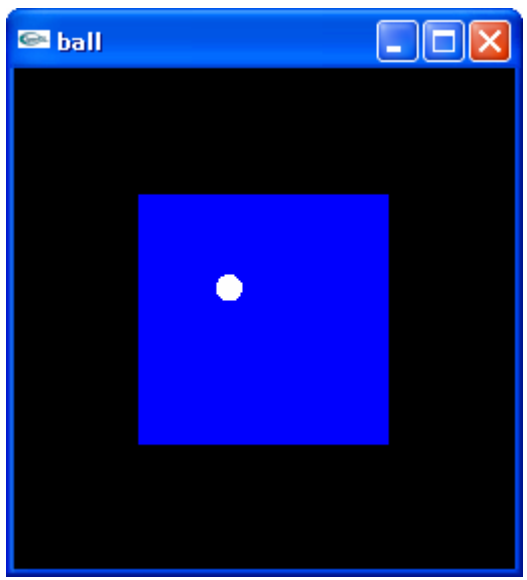
  - 描画コールバック関数の中で使用する
  - glFlush 関数の代わりに使用する
    (この関数の中で glFlush が呼び出されている)

練習

- 前回の実習課題にアニメーション機能を実装せよ。

例題

- 壁で囲まれた摩擦のない面でボールが滑る様子を表現せよ。



ソースコード（1/4）

ball.c

```
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>

double r = 0.1;
double x1 = 0.0, y1 = 0.5;
double dx1 = 0.01, dy1 = 0.01;

void display( void )
{
        glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

        glLoadIdentity();
        gluLookAt( 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 );

        glEnable( GL_DEPTH_TEST );

        glColor3f( 0.0, 0.0, 1.0 );
        glBegin( GL_POLYGON );
                glVertex3f( +1.0, +1.0, 0.0 );
                glVertex3f( -1.0, +1.0, 0.0 );
```

ソースコード（2/4）

```
                glVertex3f( -1.0, -1.0, 0.0 );
                glVertex3f( +1.0, -1.0, 0.0 );
        glEnd();

        glColor3f( 1.0, 1.0, 1.0 );
        glTranslatef( x1, y1, r );
        glutSolidSphere( r, 24, 24 );

        glDisable( GL_DEPTH_TEST );

        glutSwapBuffers();
}

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

void idle( void )
{
```

ソースコード（3/4）

```
        if( x1 + dx1 < -1.0 + r || 1.0 - r < x1 + dx1 ){
                dx1 = -dx1;
        }
        if( y1 + dy1 < -1.0 + r || 1.0 - r < y1 + dy1 ){
                dy1 = -dy1;
        }
        x1 += dx1;
        y1 += dy1;
        glutPostRedisplay();
}

int main( int argc, char *argv[] )
{
        glutInit( &argc, argv );
        glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE );
        glutInitWindowSize( 250, 250 );
        glutCreateWindow( argv[0] );

        glClearColor( 0.0, 0.0, 0.0, 0.0 );
        glMatrixMode( GL_PROJECTION );
        glLoadIdentity();
        glFrustum( -1.0, 1.0, -1.0, 1.0, 1.0, 20.0 );
```

ソースコード（4/4）

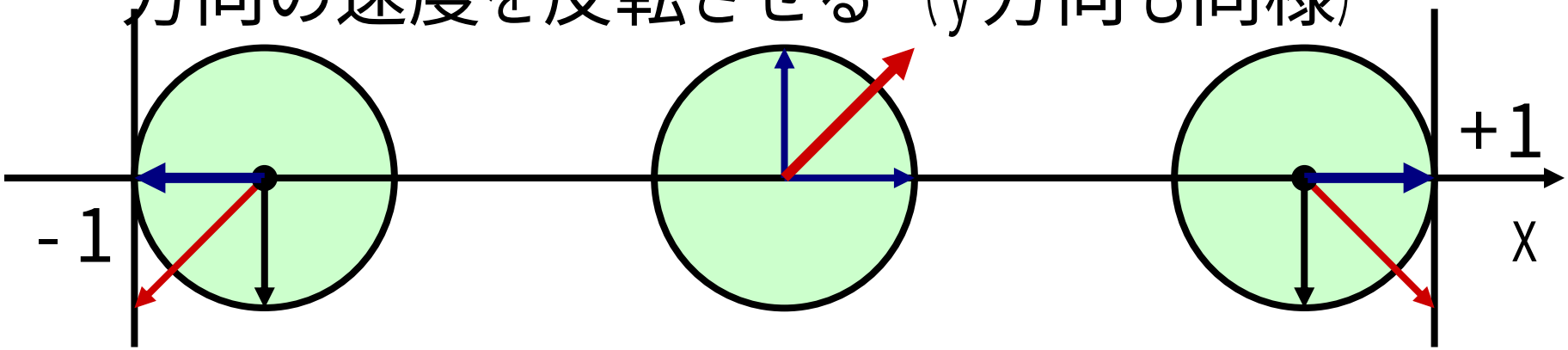
```
        glMatrixMode( GL_MODELVIEW );

        glutDisplayFunc( display );
        glutKeyboardFunc( myKeyBoard );
        glutIdleFunc( idle );
        glutMainLoop();

        return 0;
}
```

解説

- プログラムで利用するパラメータ
  - double r = 0.1;
    ボールの半径
  - double x1 = 0.0, y1 = 0.5;
    ボールの座標　(x座標、y座標)
  - double dx1 = 0.01, dy1 = 0.01;
    ボールの速度　(x方向、y方向)
    idle 関数の呼び出しごとに
    ボールの座標を変化させる

衝突判定

- 動ける範囲 -1 < x < +1 を超えるとき、x軸方向の速度を反転させる（y方向も同様）

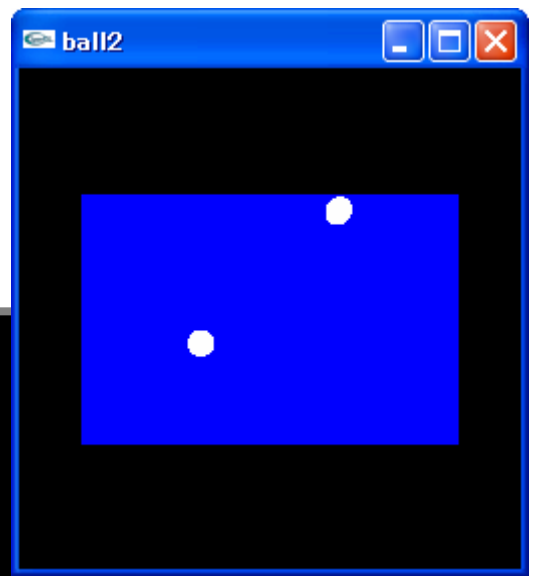



```
if( x1 + dx1 < -1.0 + r || 1.0 - r < x1 + dx1 ){
        dx1 = -dx1; // x 軸方向の速度を反転
}
x1 += dx1;  // 座標値を更新する
glutPostRedisplay();
```

実習課題

- 例題のプログラムで、ボールを２個に増やし、動く範囲を長方形にせよ。
  - 宛先：dan@cc.matsuyama-u.ac.jp
  - 件名：CG課題#13

```
C:\>ball2.exe
```



次回の予定

- 日時:　12月21日（金）
  4時限
- OpenGL によるグラフィックス
  - モデリング　(続き)