アニメーションとマウス入力 (第 10 回)

氏名　越智 秀次
クラス　理 科 一 類 11 組
学生証番号 J4-170235

▫課題10.1 - 13.2節 例 1: 物体の回転プログラム - ObjectRotate.java

○プログラムリスト

略

○考察

このプログラムではy軸の向きを逆さまにする(暗いマゼンダの面が上に来るようにする)とマウスの左右の動きと実際の立体の動きが逆になる。

▫課題10.2 - 13.2節 例 2: 立方体の回転プログラム - CubeRotate.java

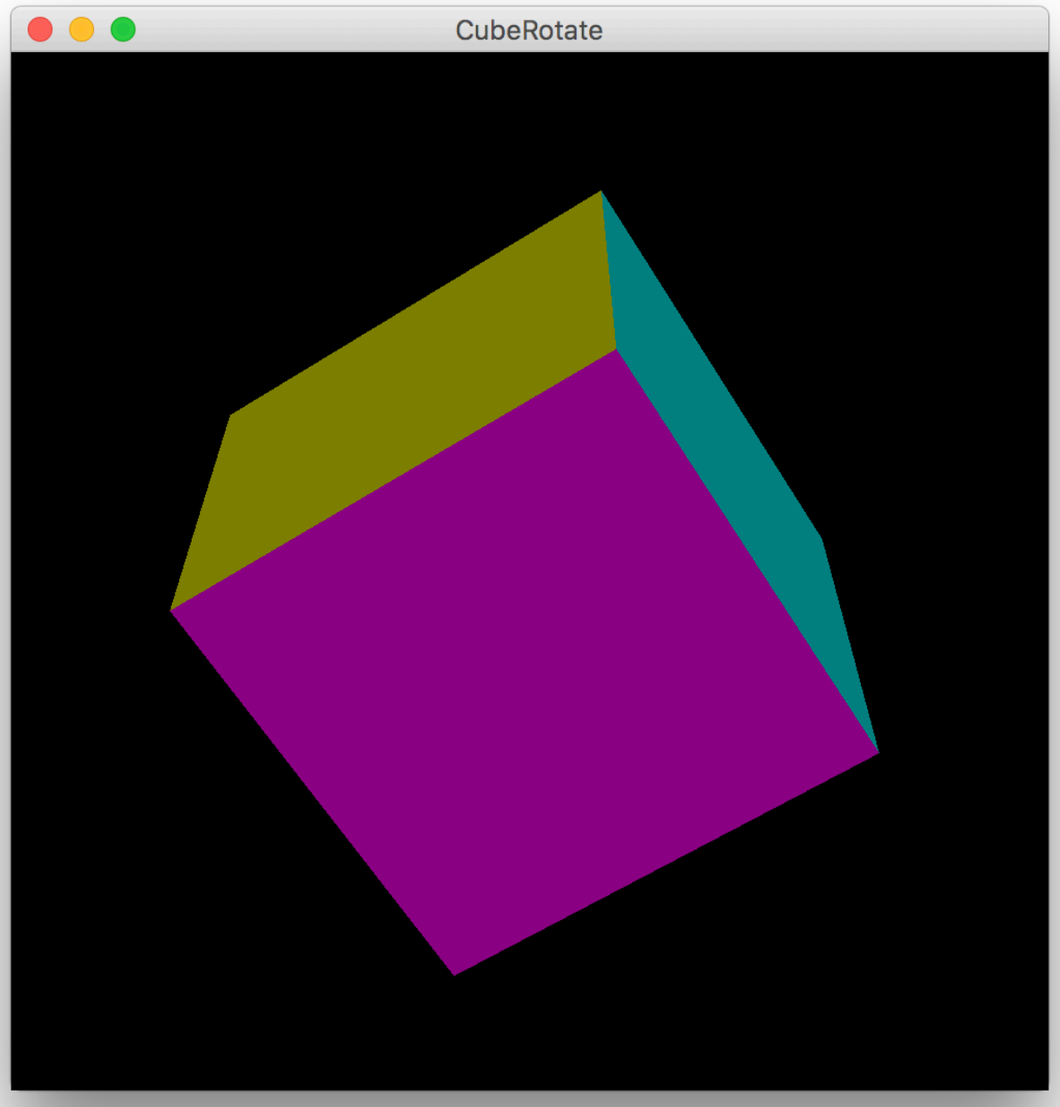
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMBP:Chap13 ochihideji$ java CubeRotate
```

○実行結果



○考察

objectとして別の立体を登録することで、立方体以外の回転プログラムも作成することができる。(フラクタル立体の場合は再帰回数の指定が必要)

▫課題10.3 - 13.2節 例 3: 物体の回転アニメーション - ObjectSpin.java

○プログラムリスト

略

○考察

○考察

回転角rotateRatioはマウスの相対移動量と正に相関しているので、より大きくマウスドラッグすればより早く立体が回転するようになる。

□課題10.4 - 13.2節 例 4: 立方体の回転アニメーション - CubeSpin.java

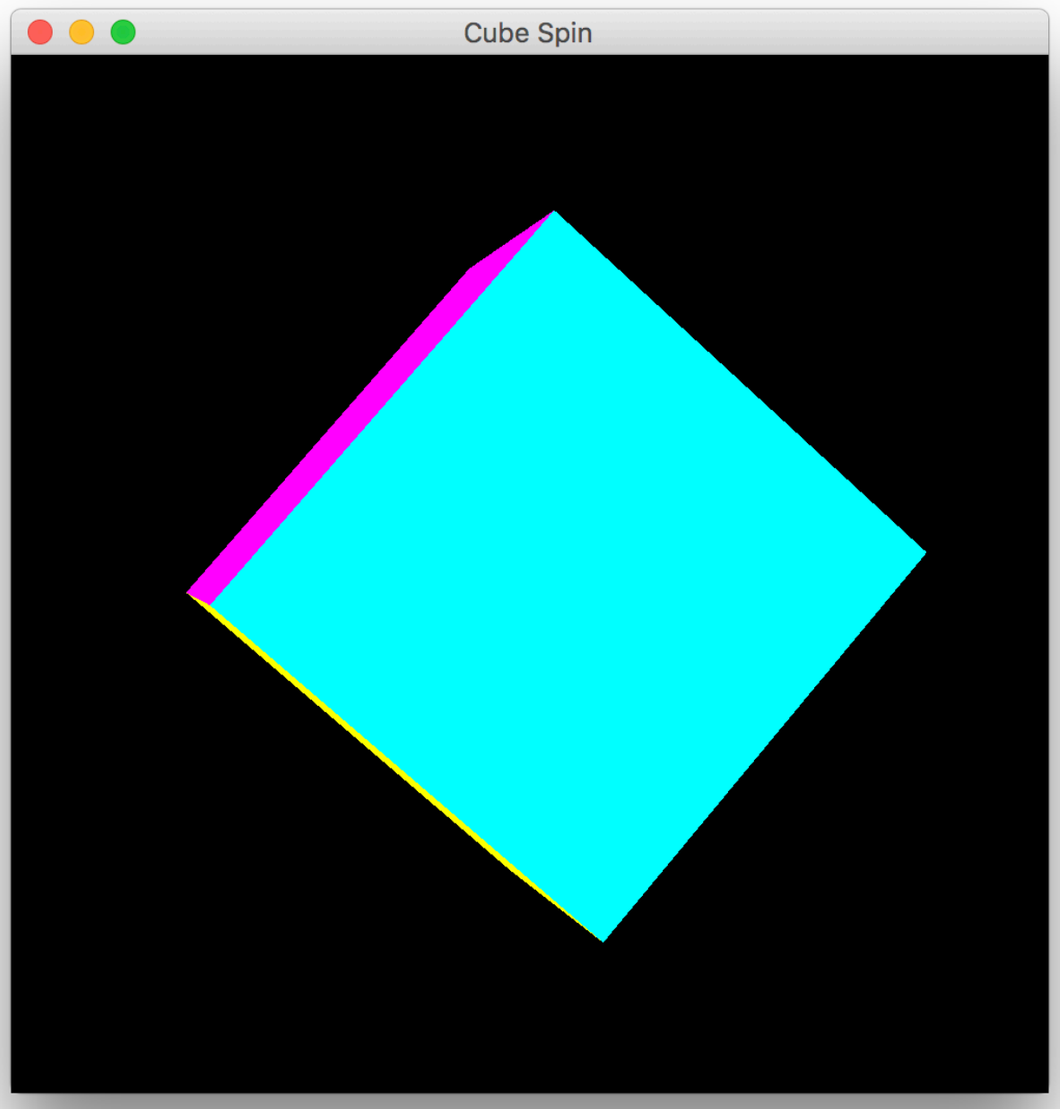
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMBP:Chap13 ochihideji$ java CubeSpin
```

○実行結果



○考察

こちらのプログラムもCubeRotate.java同様に、描画立体として立方体以外の立体を登録できる。そのプログラムは課題10.7に記載。

□課題10.5 - 章末課題: 回転速度の変更 - DumpingSpin.java / DumpingCubeSpin.java

○プログラムリスト

DumpingSpin.java

```
import java.awt.*;
import java.awt.event.*;
import javax.media.opengl.*;
import javax.media.opengl.awt.*;

public abstract class DumpingSpin extends ObjectRotate {
    protected float rotateRatio = 10.0f;
    protected int startX;
    protected int startY;
    protected double angle = 0.0;
    protected double[] axis = {0.0, 0.0, 1.0, 1.0};
    protected double[] origin = new double[4];
    protected double[] modelMatrix = new double[16];
    protected double[] projMatrix = new double[16];
    protected int[] viewport = new int[4];

    protected DumpingSpin(String name) {
        super(name);
    }

    protected void mouseInit(GLAutoDrawable drawable) {
```

```
    ((GLCanvas)drawable).addMouseListener(new MouseAdapter() {
        public void mousePressed(MouseEvent e) {
            prevX = startX = e.getX();
            prevY = startY = e.getY();
        }
        public void mouseReleased(MouseEvent e) {
            int x = e.getX();
            int y = e.getY();
            if ((x == startX) && (y == startY))
              angle = 0.0;
        }
    });
}

protected void motionInit(GLAutoDrawable drawable) {
    ((GLCanvas)drawable).addMouseMotionListener(new MouseMotionAdapter() {
        public void mouseDragged(MouseEvent e) {
            int x = e.getX();
            int y = e.getY();
            Dimension size = e.getComponent().getSize();
            float deltaX = x - prevX;
            float deltaY = y - prevY;
            angle = Math.sqrt(deltaX * deltaX + deltaY * deltaY) * rotateRatio / Math.min(size.width, size.height);
            glu.gluUnProject(origin[0] + deltaY, origin[1] + deltaX, origin[2], modelMatrix, 0, projMatrix, 0, viewport, 0, axis, 0);
            prevX = x;
            prevY = y;
        }
    });
}



protected void positionInit() {
    gl.glTranslatef(0.0f, 0.0f, depth);
    gl.glRotatef(rotx, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(roty, 0.0f, 1.0f, 0.0f);
    gl.glRotatef(rotz, 0.0f, 0.0f, 1.0f);
    gl.glGetDoublev(GL2.GL_PROJECTION_MATRIX, projMatrix, 0);
    gl.glGetIntegerv(GL2.GL_VIEWPORT, viewport, 0);

}

protected void setMatrix() {
    angle = angle * 0.99f;
    gl.glRotated(angle, axis[0], axis[1], axis[2]);
    gl.glGetDoublev(GL2.GL_MODELVIEW_MATRIX, modelMatrix, 0);
    glu.gluProject(0.0, 0.0, 0.0, modelMatrix, 0, projMatrix, 0, viewport, 0, origin, 0);
    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
}
```

```
    protected void resetMatrix() {
    }
}
```

DumpingCubeSpin.java

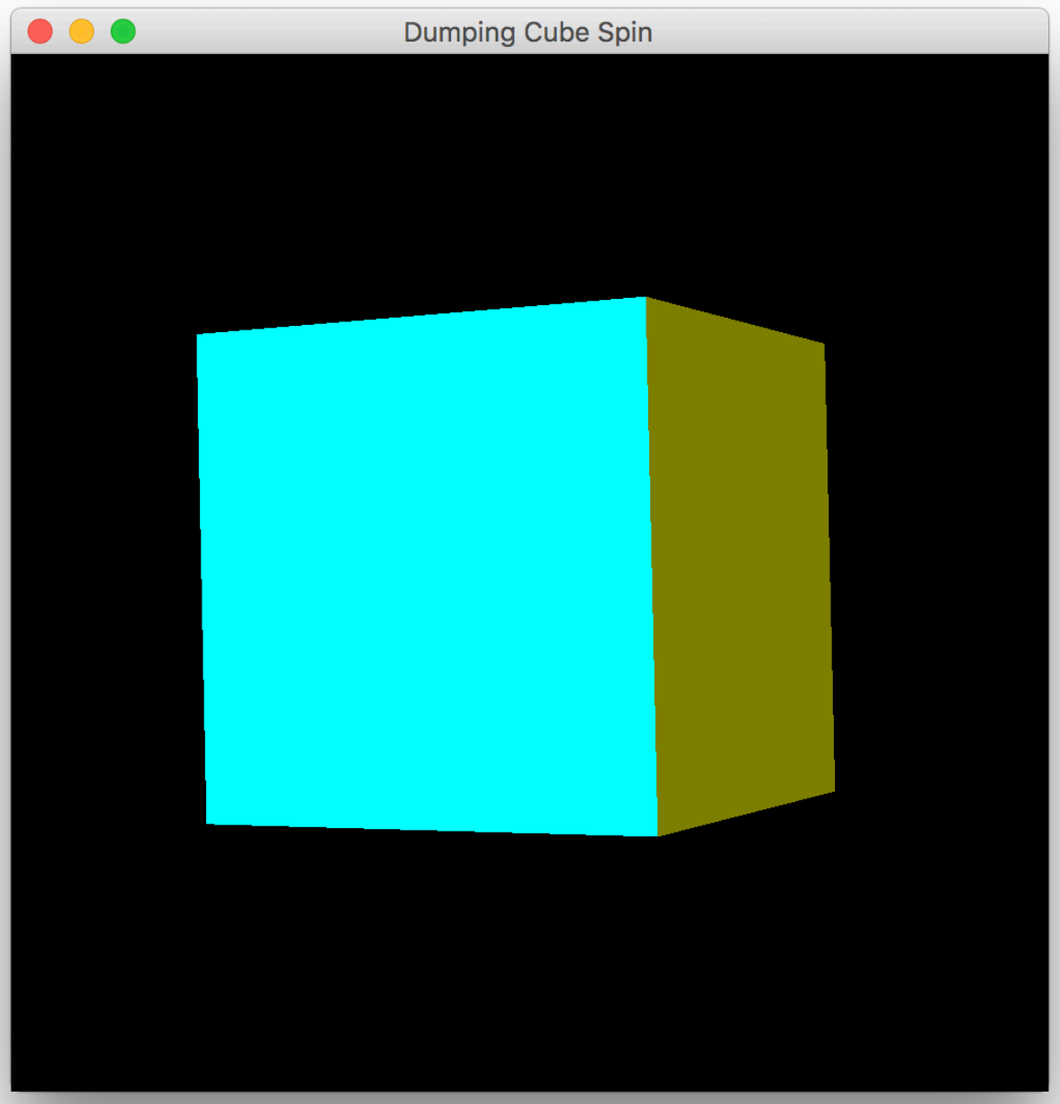
```
public class DumpingCubeSpin extends DumpingSpin {
    public static void main(String[] args) {
        (new DumpingCubeSpin("Dumping Cube Spin")).showFrame();
    }

    protected DumpingCubeSpin(String name) {
        super(name);
        object = new Cube();
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap13 ochihideji$ java DumpingCubeSpin
```

○実行結果



○考察

行列が設定される毎にangleを0.99倍にすることによって、回転の速度を指数関数的に減少させている。ちなみにangleを0.99倍ではなく、0.9倍にすると減衰速度が速く瞬く間に回転が停止する。連続的なアニメーションを生成するために非常に短い時間間隔で何度もdisplayメソッドの起動が行われているということがわかる。

□課題10.6 - 章末課題: 回転速度の対話的な調整 - WheelSpin.java / WheelCubeSpin.java

○プログラムリスト

WheelSpin.java

```
import java.awt.*;
import java.awt.event.*;
import javax.media.opengl.*;
import javax.media.opengl.awt.*;

public abstract class WheelSpin extends ObjectRotate {
    protected int startX;
    protected int startY;
    protected float rotateRatio = 10.0f;
    protected double angle = 0.0;
    protected double[] axis = {0.0, 0.0, 1.0, 1.0};
    protected double[] origin = new double[4];
    protected double[] modelMatrix = new double[16];
    protected double[] projMatrix = new double[16];
    protected int[] viewport = new int[4];
    protected float deltaX;
    protected float deltaY;
    Dimension size;

    protected WheelSpin(String name) {
        super(name);
    }

    protected void mouseInit(GLAutoDrawable drawable) {
        ((GLCanvas)drawable).addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                prevX = startX = e.getX();
                prevY = startY = e.getY();
            }
            public void mouseReleased(MouseEvent e) {
                int x = e.getX();
                int y = e.getY();
                if ((x == startX) && (y == startY))
                  angle = 0.0;
            }
        });
    }

    protected void motionInit(GLAutoDrawable drawable) {
        ((GLCanvas)drawable).addMouseMotionListener(new MouseMotionAdapter() {
            public void mouseDragged(MouseEvent e) {
                int x = e.getX();
                int y = e.getY();
```

```
            size = e.getComponent().getSize();
            deltaX = x - prevX;
            deltaY = y - prevY;
            angle = Math.sqrt(deltaX * deltaX + deltaY * deltaY) * rotateRatio / Math.min(size.width, size.height);
            glu.gluUnProject(origin[0] + deltaY, origin[1] + deltaX, origin[2], modelMatrix, 0, projMatrix, 0, viewport, 0, axis, 0);

            prevX = x;
            prevY = y;
        }
    });
    ((GLCanvas)drawable).addMouseWheelListener(new MouseAdapter() {
        public void mouseWheelMoved(MouseWheelEvent mwe) {
            float r = (float)mwe.getWheelRotation();
            float s = (float)Math.sqrt(deltaX * deltaX + deltaY * deltaY) * rotateRatio / Math.min(size.width, size.height) + r;
            angle = (s > 0.1f) ? s : 0.1f;
        }
    });
}



protected void positionInit() {
    gl.glTranslatef(0.0f, 0.0f, depth);
    gl.glRotatef(rotx, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(roty, 0.0f, 1.0f, 0.0f);
    gl.glRotatef(rotz, 0.0f, 0.0f, 1.0f);
    gl.glGetDoublev(GL2.GL_PROJECTION_MATRIX, projMatrix, 0);
    gl.glGetIntegerv(GL2.GL_VIEWPORT, viewport, 0);

}

protected void setMatrix() {
    gl.glRotated(angle, axis[0], axis[1], axis[2]);
    gl.glGetDoublev(GL2.GL_MODELVIEW_MATRIX, modelMatrix, 0);
    glu.gluProject(0.0, 0.0, 0.0, modelMatrix, 0, projMatrix, 0, viewport, 0, origin, 0);
    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
}

protected void resetMatrix() {
}
}
```

WheelCubeSpin.java

```
public class WheelCubeSpin extends ObjectSpin {
    public static void main(String[] args) {
        (new WheelCubeSpin("Wheel Cube Spin")).showFrame();
    }

    protected WheelCubeSpin(String name) {
        super(name);
        object = new Cube();
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap13 ochihideji$ java WheelCubeSpin
```

○実行結果

○考察

mwe.getWheelRotation()はホイールの回転方向によって正の値を返すか負の値を返すかが決まる。angleが負の値になると立体が最初の回転方向と逆に回転するようになるので、ここではangleが0.1より小さな値を取らないように設定した。

□課題10.7 - 自習課題: フラクタル立体の回転アニメーション - SierpinskiSpin.java

○プログラムリスト

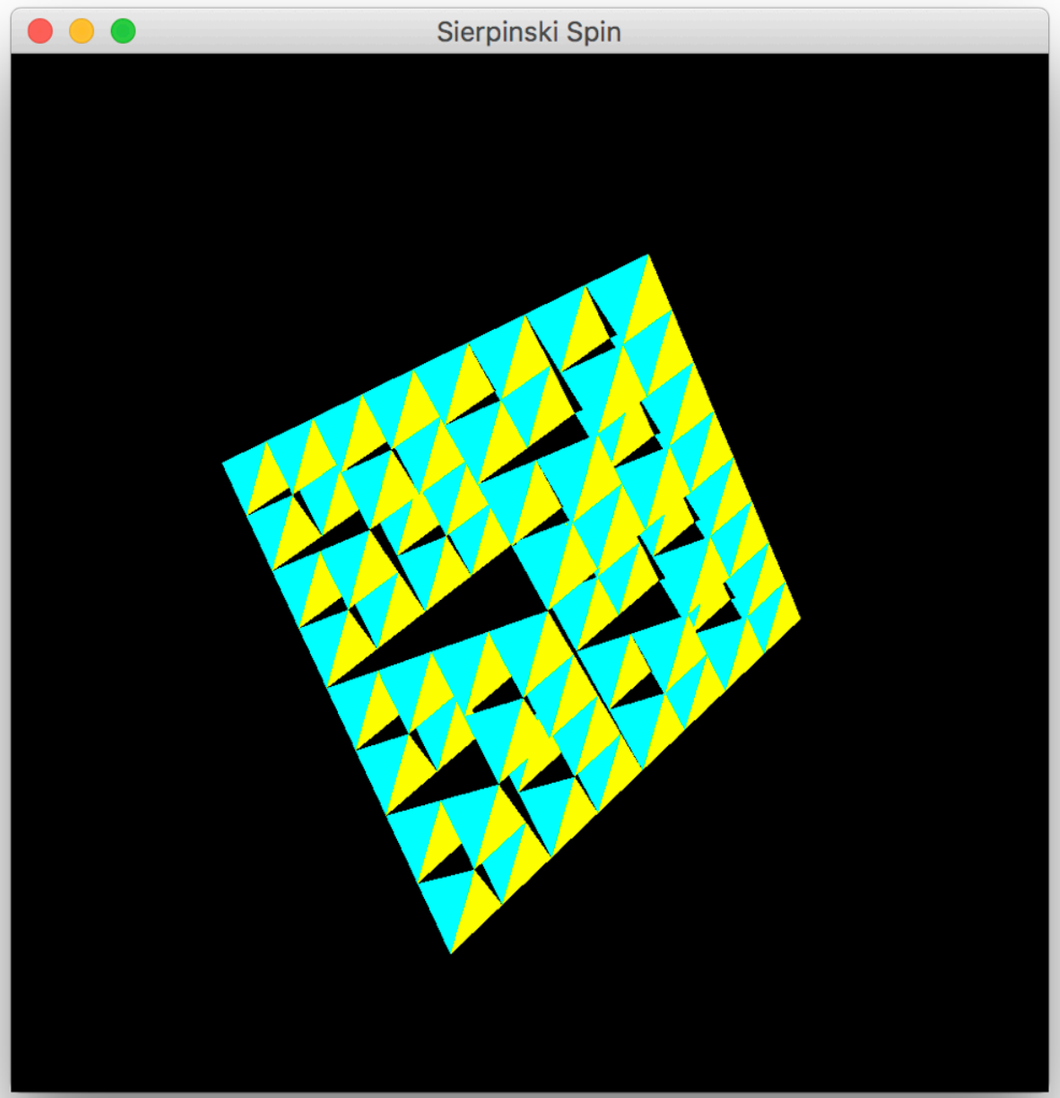
```
public class SierpinskiSpin extends ObjectSpin {
    public static void main(String[] args) {
        if (args.length == 0)
            System.err.println("Usage: java SierpinskiSpin #iteration");
        else
            (new SierpinskiSpin("Sierpinski Spin", Integer.parseInt(args[0]))).showFrame();
    }

    protected SierpinskiSpin(String name, int times) {
        super(name);
        object = new Sierpinski(times);
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap13 ochihideji$ java SierpinskiSpin 3
```

○実行結果



○考察

課題に指定はされていなかったが、フラクタル立体を様々な角度から観察してみたいと思いプログラムを作成した。

□課題や授業に関して

○レポート作成に要した時間

5時間以上

○特に苦労した点

今回も章末課題作成に当たって細かいミスを多数してしまったため、レポート作成に非常に時間を要してしまった。前回行えなかった多面体の作成なども今週行いたいと思っていたが、時間が足りず作成できなかった。

○授業についての感想や希望

今回の章末課題は非常に難しかったです。