多面体とフラクタル図形 (第 9 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9.1 - 12.3節 例 1~5: 行列スタックを利用したプログラム

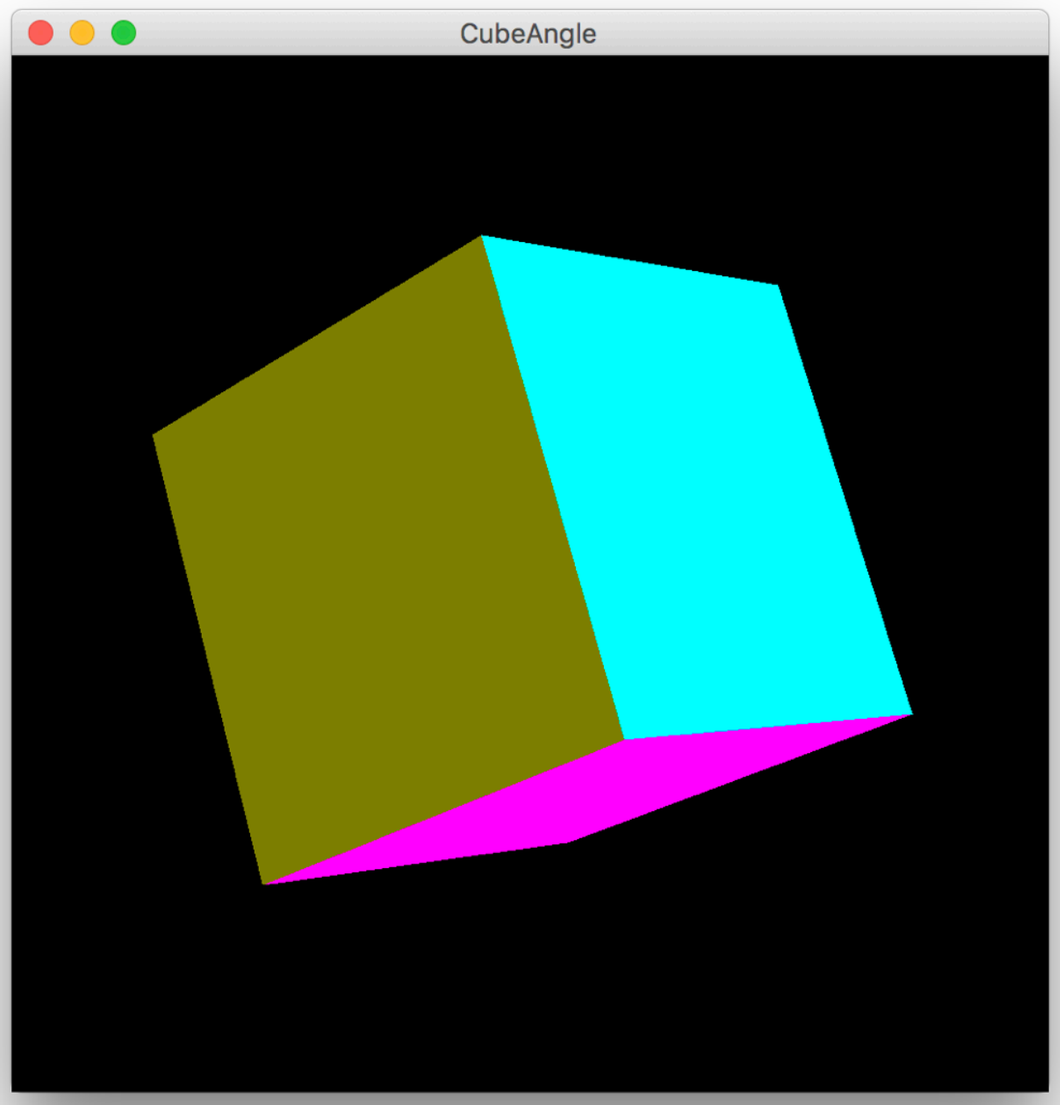
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMBP:Chap12 ochihideji$ java CubeAngle 180 40 -20
```

○実行結果



○考察

glPushMatrixとglPushMatrixがないときCubeAngleを実行すると、最初は立方体が傾いて表示され、ウィンドウサイズを変更すると元の視点角度に戻り、ウィンドウを最小化して再び開くと一定の角度だけ立方体が回転する。ウィンドウサイズ変更時の挙動はreshapeメソッドのpositioninit()によるもので、ウィンドウ最小化時の挙動はdisplayメソッド(glPushMatrixとglPopMatrixがない)によるものである。

□課題9.2 - 12.3節 例 6: フラクタル立体のプログラム - FractalObject.java

○プログラムリスト

略

○考察

フラクタル図形では 1 つ前のステップの図形が残らないように描画する必要があるので、第6章のFractal.java同様実際の描画をしているのは再帰レベル lが0になった時だけである。

□課題9.3 - 12.3節 例 7: メンガースポンジ - MengerSponge.java

○プログラムリスト

略

○考察

メンガースポンジでは頂点方向と稜線方向の両方に向かって縮小後の図形を移動させていることに注意する。

□課題9.4 - 12.3節 例 8: 角度指定によるメンガースポンジの描画 - MengerSpongeAngle.java

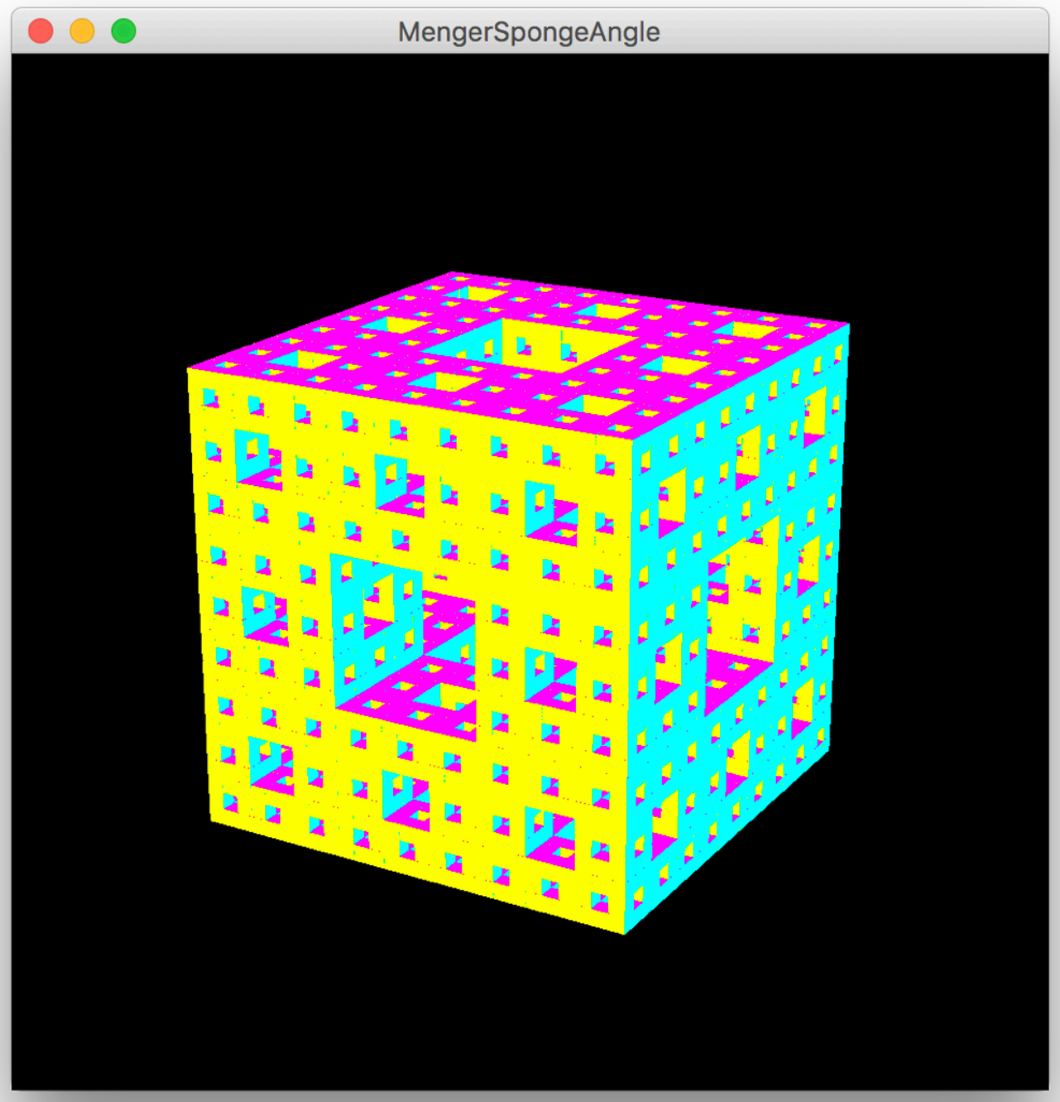
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMBP:Chap12 ochihideji$ java MengerSpongeAngle 3
```

○実行結果



○考察

コマンド実行時に与える数字のうち、一番最初の数字は再帰回数を与えるので11行目でoffsetの値が1になっている。このプログラムではコマンド実行時に5つ以上の数字を与えると、標準の投影法で描画されることになる。(再帰回数は正しく認識される。)

□課題9.5 - 章末課題：正多面体の表示 - Tetrahedron.java / TetrahedronAngle.java

○プログラムリスト

Tetrahedron.java

```
public class Tetrahedron extends Polyhedron {
    private static final float[][] tetraVertices = { {-1.0f, -1.0f, -1.0f}, {1.0f, 1.0f, -1.0f}, {1.0f, -1.0f, 1.0f}, {-1.0f, 1.0f, 1.0f} };
    private static final int[][] tetraFaces = { {0, 1, 2}, {1, 3, 2}, {0, 2, 3}, {0, 3, 1} };
    private static final int[][] tetraEdges = {{1, 3}, {3, 2}, {2, 1}, {3, 4}, {4, 2}, {4, 1}};
    private static final float[][] tetraFaceColors = { {0.0f, 1.0f, 1.0f}, {1.0f, 0.0f, 1.0f}, {1.0f, 1.0f, 0.0f}, {1.0f, 0.0f, 0.0f} };
    protected Tetrahedron() {
        vertices   = tetraVertices;
        faces      = tetraFaces;
        edges      = tetraEdges;
        faceColors = tetraFaceColors;
    }
}
```

TetrahedronAngle.java

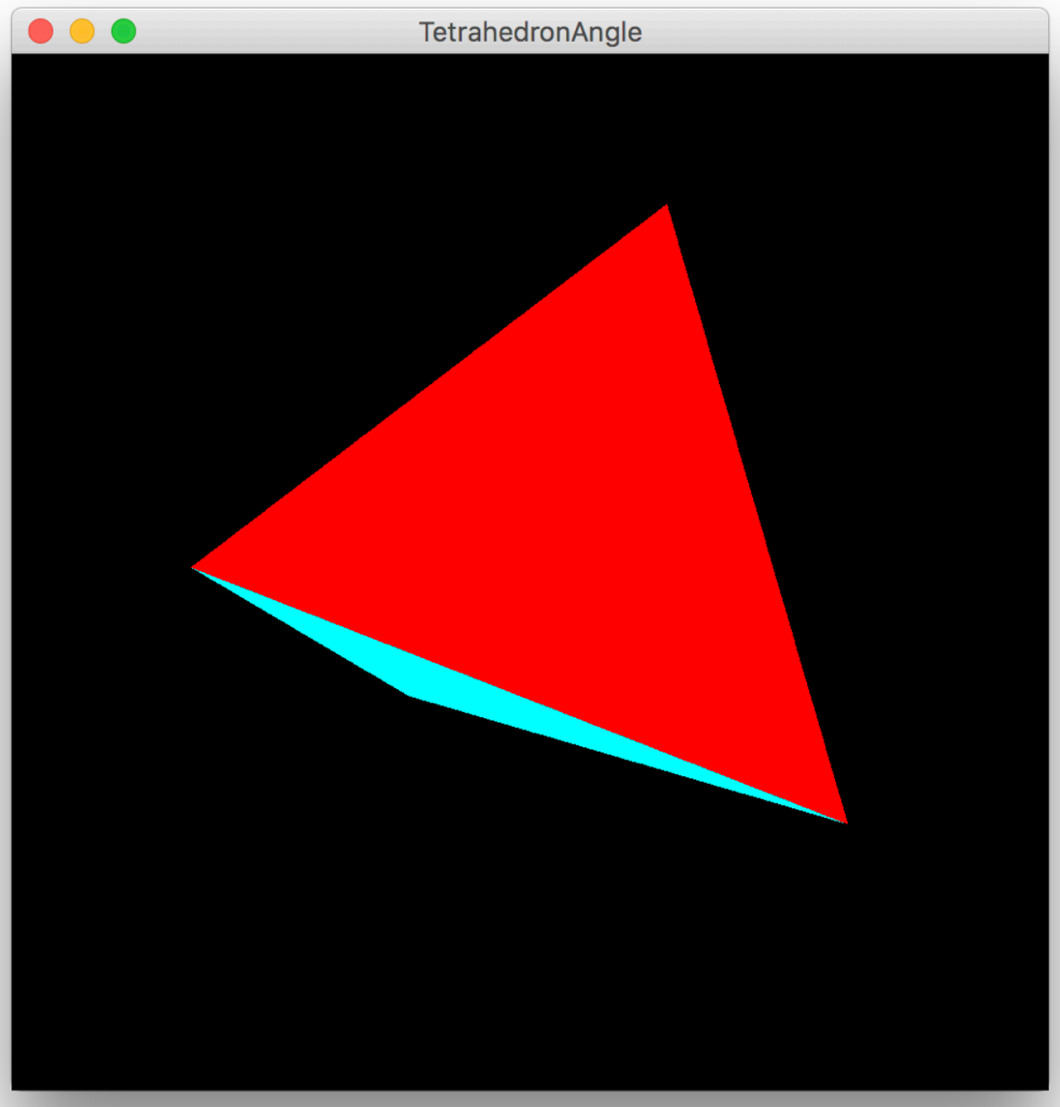
```
public class TetrahedronAngle extends ObjectAngle {
    public static void main(String[] args) {
      if (args.length > 0 && args.length < 4)
```

```
      (new TetrahedronAngle("TetrahedronAngle", args, 0)).showFrame();
    else
      (new TetrahedronAngle("TetrahedronAngle")).showFrame();
  }

  protected TetrahedronAngle(String name) {
    super(name);
    object = new Tetrahedron();
  }

  protected TetrahedronAngle(String name, String[] args, int offset) {
    super(name, args, offset);
    object = new Tetrahedron();
  }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap12 ochihideji$ java TetrahedronAngle 30 180 -20
```

○実行結果



○考察

正四面体については前々回の課題で頂点や面に関する情報を求めていたので、稜線情報を考えるだけで済んだ。

□課題9.6 - 章末課題: フラクタル立体の表示 - Sierpinski.java / SierpinskiAngle.java

○プログラムリスト

Sierpinski.java

```
public class Sierpinski extends FractalObject {
    protected Sierpinski(int times) {
        super(times);

        primitive = new Tetrahedron();
        int nv = primitive.vertices();
        s = 1.0 / 2.0;
        v = new double[nv][3];
        for (int i = 0; i < nv; i ++)
          for (int j = 0; j < 3; j++)
            v[i][j] = primitive.vertex(i, j) * (1.0 - s);
    }
}
```

SierpinskiAngle.java

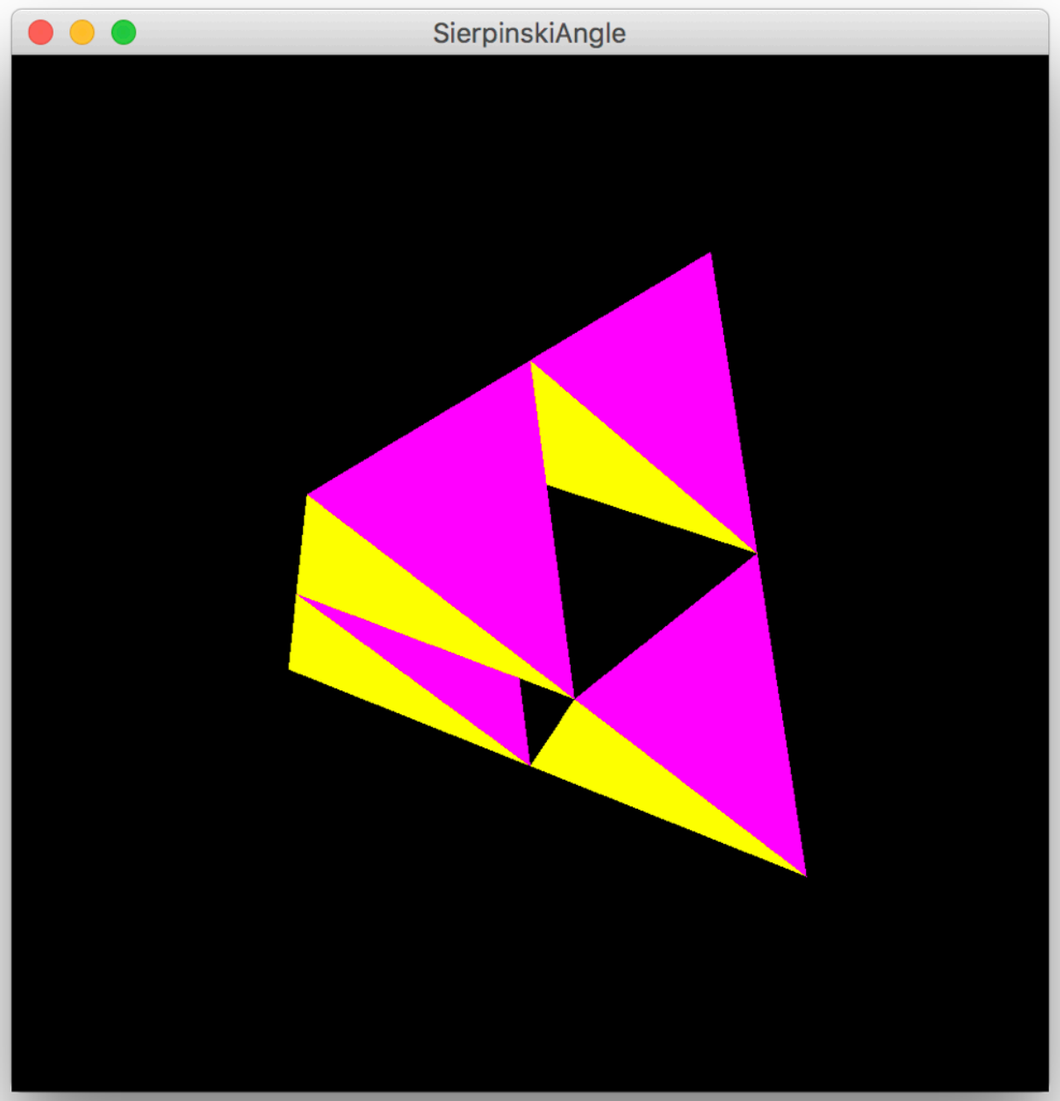
```
public class SierpinskiAngle extends ObjectAngle {
    static int times = 0;
```

```
    public static void main(String[] args) {
        if (args.length == 0) {
            System.err.println("Usage: SierpinskiAngle #iteration[ ... ]");
        }
        else {
            times = Integer.parseInt(args[0]);
            if (args.length > 1 && args.length < 5) {
                (new SierpinskiAngle("SierpinskiAngle", args, 1)).showFrame();
            }
            else {
                (new SierpinskiAngle("SierpinskiAngle")).showFrame();
            }
        }
    }

    protected SierpinskiAngle(String name) {
        super(name);
        object = new Sierpinski(times);
    }

    protected SierpinskiAngle(String name, String[] args, int offset) {
        super(name, args, offset);
        object = new Sierpinski(times);
    }
}
```

○実行コマンド

```
ochihidejinoMBP:Chap12 ochihideji$ java SierpinskiAngle 1 30 10 0
```

○実行結果



○考察

シェルピンスキー正四面体についてはメンガースポンジと異なり、稜線方向の平行移動は含まれないことに注意する。(至極当然のことであるが、私はそれに気づかずにMengerSponge.javaをそのまま書き換えようとしたため実行時に大量のエラーが生じた。)

▫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

正八面体、正二十面体についても描画プログラムを書いたが、何度見直してもjava.lang.ArrayIndexOutOfBoundsExceptionのエラーが大量に生じ、レポート作成に莫大な時間を要した上結局レポート期限までに修正することができなかった。おそらく頂点情報と面情報に齟齬があるため、もしくはシェルピンスキー四面体の場合のように何か簡単なミスに気づくことができていないためにこのようになっていると考えられる。

○授業についての感想や希望

多面体の面に関する情報については頂点番号列の指定方法にルールがありましたが、稜線に関しても頂点番号列の指定方法に何らかのルールがあるのでしょうか？