表示空間(平面)と2次元座標変換 (第 5 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5.1 - 5.2節 例 3: Vector2とMatrix2X2の確認ーCheckVM.java

○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CheckVM
```

○実行結果

```
|(3.0, 4.0)| = 5.0
(1.0, 1.0) + (3.0, 4.0) = (4.0, 5.0)
(3.0, 4.0) - (1.0, 1.0) = (2.0, 3.0)
(3.0, 4.0) * 2.0 = (6.0, 8.0)
Normalize (0.0, 0.0) = (0.0, 0.0)
Normalize (3.0, 4.0) = (0.6000000000000001, 0.8)
Rotate    (1.0, 1.0) = (-0.9999999999999999, 1.0)
Scale&Rot (1.0, 1.0) = (-1.9999999999999998, 2.0)
Inv&Rot   (3.0, 4.0) = (3.0, 4.0)
```

▫課題5.2 - 5.2節 例 4: 仮想の表示空間をもつCanvasーCGCanvas.java

○プログラムリスト

略

○考察

Canvas上の位置から表示空間への変換およびその逆の変換の際には、Canvasと表示空間とでy軸正の向きが反対になっていることに気をつけなければならない。CGCanvas.javaの85行目と91行目で変換の際に-1という係数がつくのはこのためである。

▫課題5.3 - 5.3節 例 1 : 円の描画ーCGCircle.java

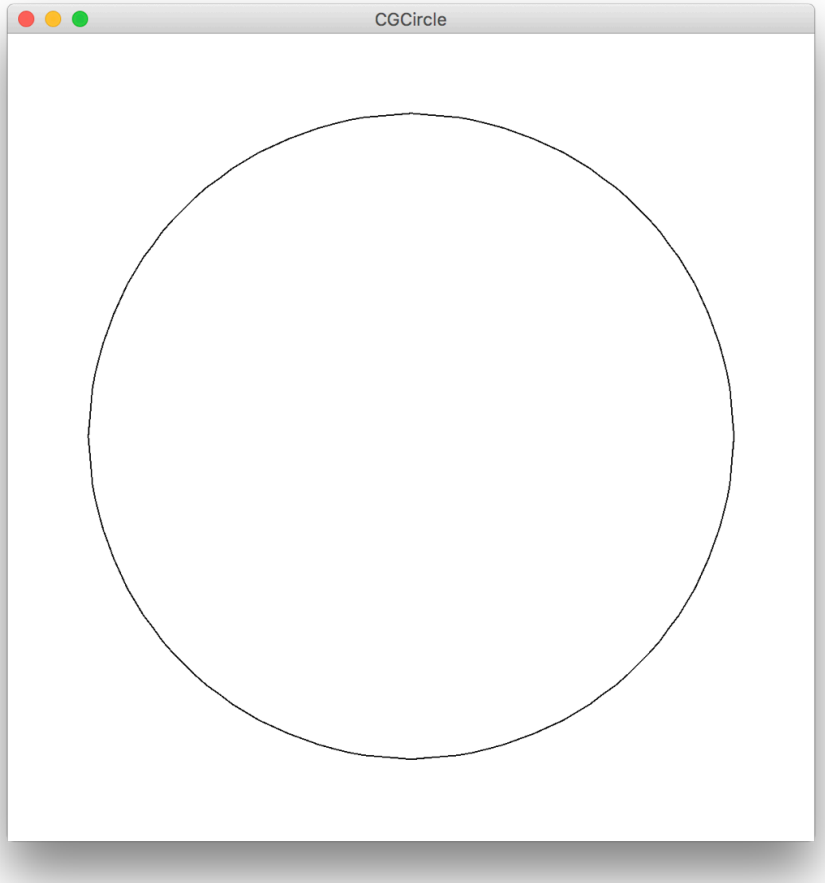
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGCircle 128
```

○実行結果



○考察

drawCircleなどのメソッドを使う際には、表示空間の広さ(range)の設定に気をつける必要がある。例えば、CGCircle.javaにおいて　radius = 0.8 * 2 とすると円が表示空間からはみ出でしまうが、setRangeメソッドでrangeを4.0に設定すると元の円と同じ大きさの円を描くことになる。

▫課題5.4 - 5.3節 例 2: 円によるカージオイドの描画ーCGCardioidCircle.java

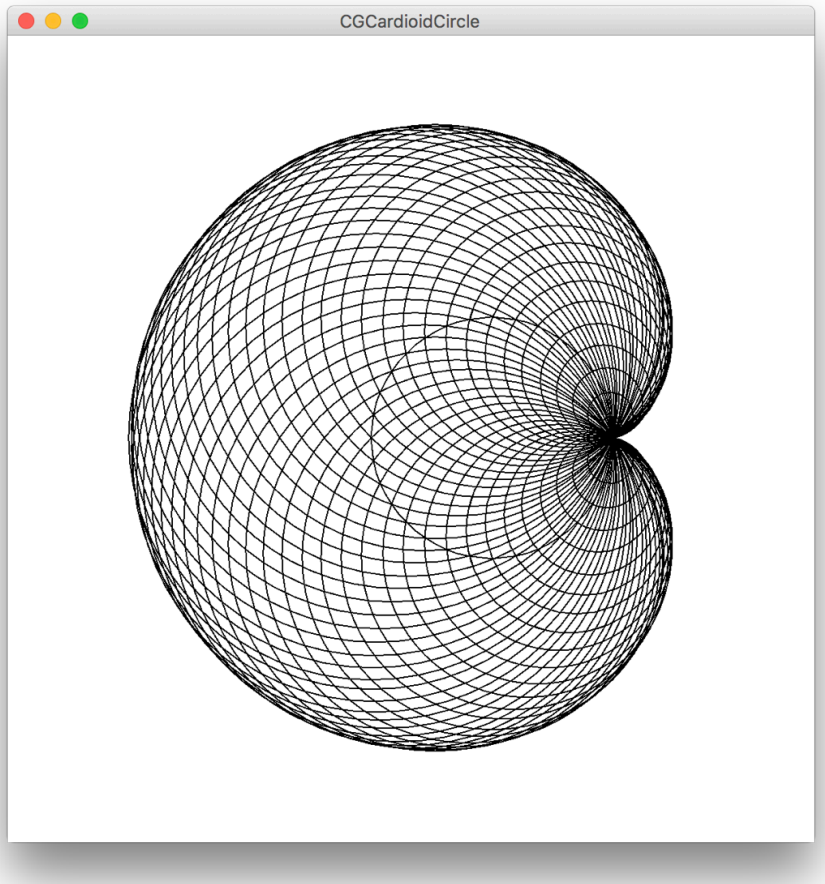
○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGCardioidCircle 64
```

○実行結果



○考察

第二章のCardioidOval.javaと比較すると、円の半径の記述が簡潔になっている。

□課題5.5 - 5.3節 例 3: マーカーの描画ーCGMarker.java

○プログラムリスト

略

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGMarker
```

○実行結果

○考察

このプログラムに限ったことではないが、スクリーン座標系とワールド座標系の区別を意識しながらプログラムを書くことが大切である。CGMarker.javaでは、マウスイベントによって取得したxやyはスクリーン座標である。そのためfillMarkerメソッドを適用する際にはpointメソッドによってワールド座標に変換する必要がある。

□課題5.6 - 章末問題: ダイヤモンドパターンやネフロイドの描画ー CGDiamondPattern.java & CGNephroidCircle.java

○プログラムリスト

-ダイヤモンドパターン-

```
import java.awt.*;
import java.awt.event.*;

public class CGDiamondPattern extends CGCircle {
    public static void main(String[] args) {
        if (args.length == 0)
          System.err.println("Usage: java CGDiamondPattern #");
        else
          radius = 0.8;
          (new CGDiamondPattern("CGDiamondPattern", Integer.parseInt(args[0]))).showFrame();
    }


    protected CGDiamondPattern(String name, int numberOfPoints) {
      super(name, numberOfPoints);
      setOrigin(360, 299);
  }

  public void paint(Graphics g) {
      super.paint(g);
```

```
      for (int i = 0; i < points.length; i++) {
        for (int j = i + 2; j < points.length; j++) {
          drawLine(g, points[i], points[j]);
        }
      }
  }
}
```

-ネフロイド-

```
import java.awt.*;

public class CGNephroidCircle extends CGCircle {
    public static void main(String[] args) {
        if (args.length == 0)
          System.err.println("Usage: java CGNephroidCircle #");
        else {
          radius = 0.5;
          (new CGNephroidCircle("CGNephroidCircle", Integer.parseInt(args[0]))).showFrame();
        }
    }

    protected CGNephroidCircle(String name, int numberOfPoints) {
        super(name, numberOfPoints);
        setOrigin(360, 350);
    }

    public void paint(Graphics g) {
        super.paint(g);
        for (int i = 0; i < points.length; i++) {
            int j = points.length - i - 1;
            drawCircle(g, points[i], points[i].subtract(points[j]).norm()/2);
        }
    }
}
```

○実行コマンド

-ダイヤモンドパターン-

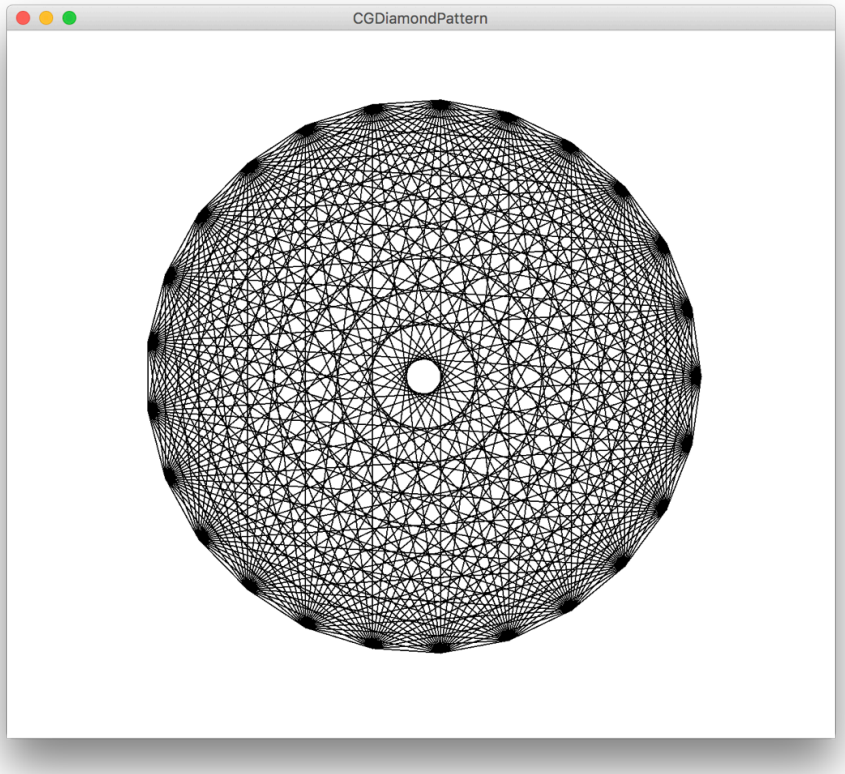
```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGDiamondPattern 25
```

-ネフロイド-

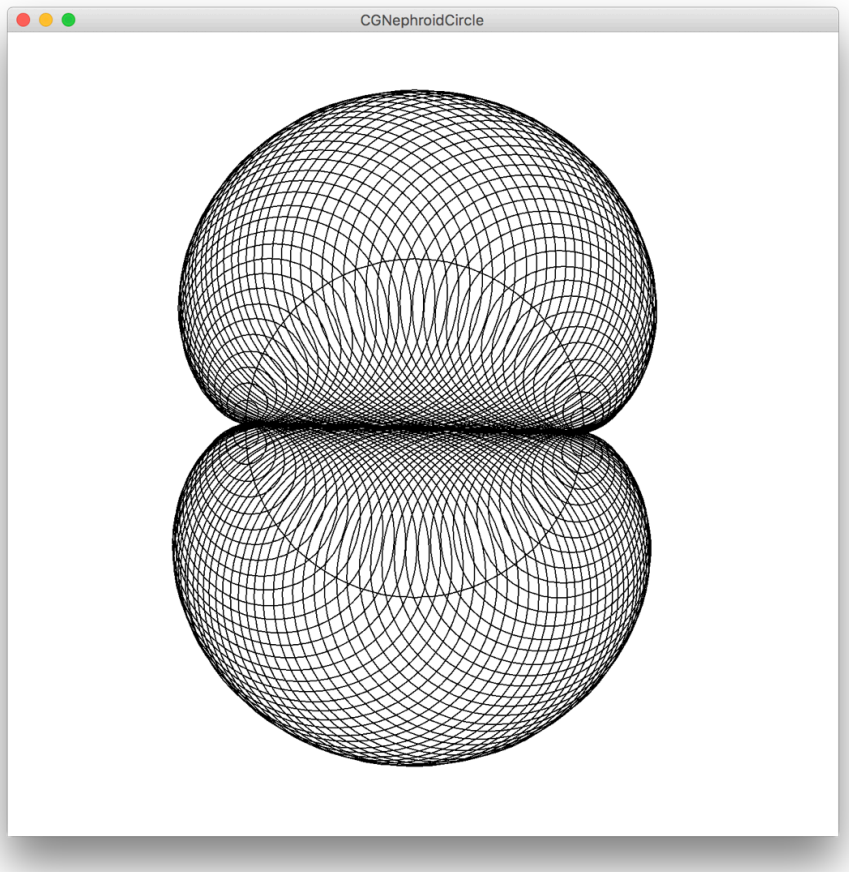
```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGNephroidCircle 128
```

○実行結果

-ダイヤモンドパターン-



-ネフロイド-



○考察

両者とも5.3の例2のpaintメソッドを書き換えるだけなのでさほど難しくはない。pointsは第2章の場合とは異なりワールド座標であることに注意する。

□課題5.6 - 章末問題: マウスによる円の描画

○プログラムリスト

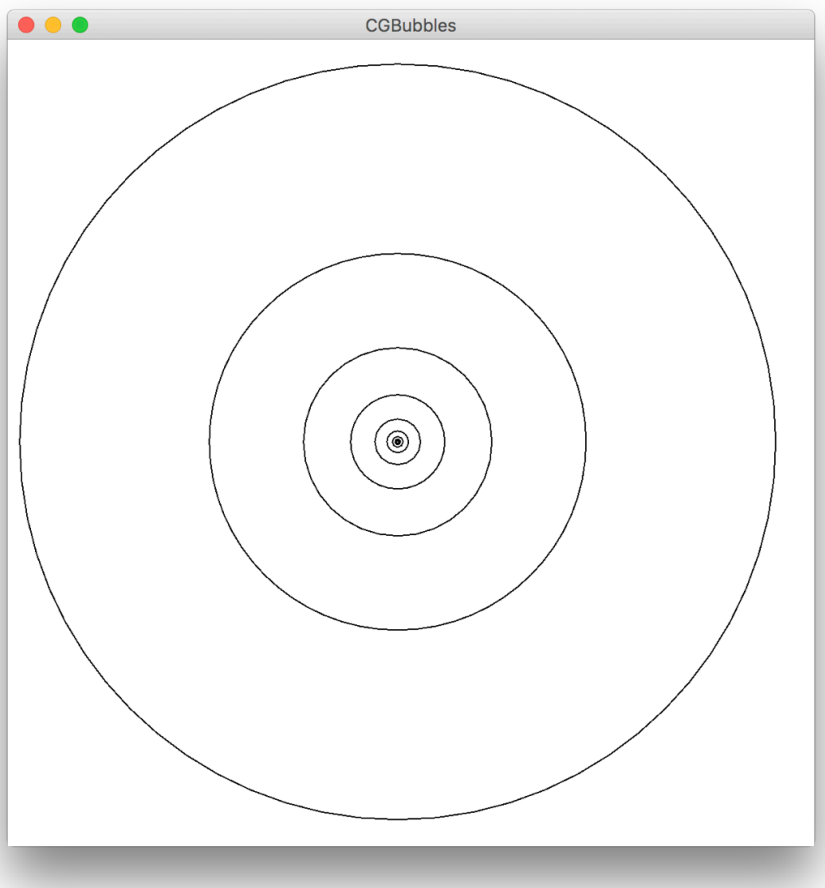
```
import java.awt.*;
import java.awt.event.*;
```

```
public class CGBubbles extends CGCanvas {
    public static void main(String[] args) {
        new CGBubbles("CGBubbles").showFrame();
    }

    protected int centerX;
    protected int centerY;

    protected CGBubbles(String name) {
        super(name);
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent me)  {
                centerX = me.getX();
                centerY = me.getY();
                setOrigin(centerX, centerY);
            }
        });

        addMouseMotionListener(new MouseMotionAdapter() {
            public void mouseDragged(MouseEvent me) {
                Graphics g = getGraphics();
                update(g);
                int x = me.getX();
                int y = me.getY();
                double radius = point(x ,y).norm();
                for (double i = 1; i > Vector2D.EPSILON; i = i * 0.5) {
                    drawCircle(g, point(centerX, centerY), i * radius);
                }
            }
        });
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGBubbles
```

○実行結果



○考察

マウスのクリック&ドラッグによって半径を指定した円および半径がその1/2倍、1/4倍、1/8倍...の円を同時に描くプログラムである。このような繰り返しの操作が行えることからもCGCanvasの有用性が感じられる。

□課題5.6 - 自習課題：マウスによるダイヤモンドパターンの描画ー CGDiamondBubble.java

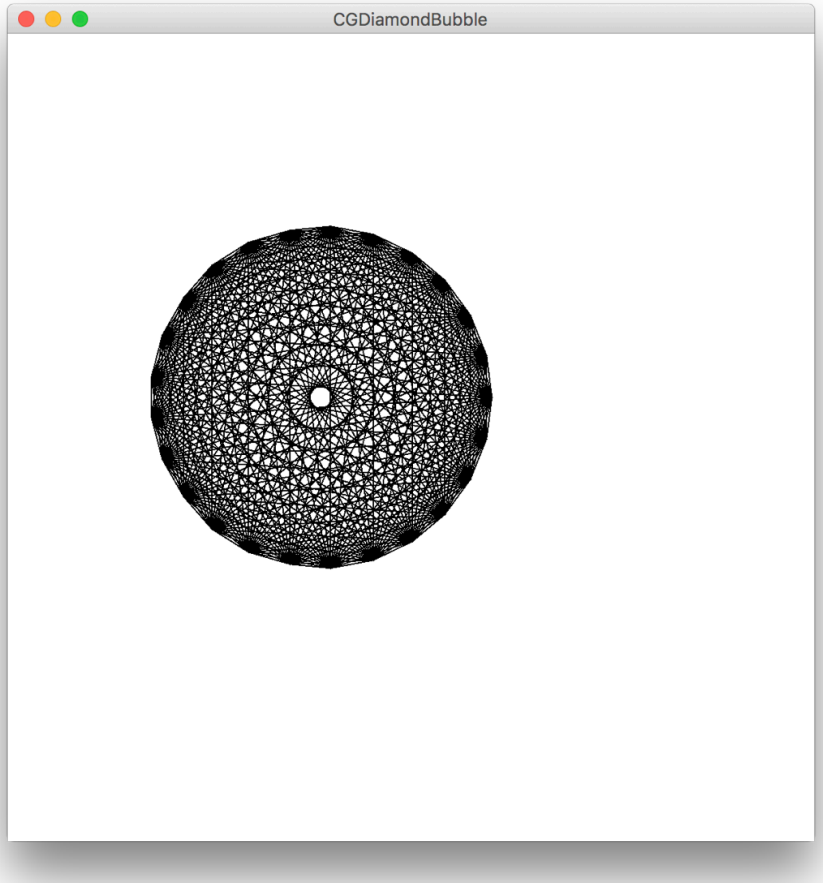
○プログラムリスト

```
import java.awt.*;
import java.awt.event.*;

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

    protected int centerX;
    protected int centerY;
    protected Vector2D[] points;

    protected CGDiamondBubble(String name, int numberOfPoints) {
        super(name);
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent me)  {
                centerX = me.getX();
                centerY = me.getY();
                setOrigin(centerX, centerY);
            }
        });

        addMouseMotionListener(new MouseMotionAdapter() {
            public void mouseDragged(MouseEvent me) {
                Graphics g = getGraphics();
                update(g);
                int x = me.getX();
                int y = me.getY();
                double radius = point(x ,y).norm();
                points = new Vector2D[numberOfPoints];
                for (int i = 0; i < numberOfPoints; i++) {
                   double theta = 2.0 * Math.PI * i / numberOfPoints;
                   points[i] = new Vector2D(radius*Math.cos(theta), radius*Math.sin(theta));
                }
                drawPolygon(g, points);
                for (int j = 0; j < numberOfPoints; j++) {
                    for (int k = j + 2; k < numberOfPoints; k++) {
                        drawLine(g, points[j], points[k]);
                    }
                }
            }
        });
    }
}
```

○実行コマンド

```
ochihidejinoMacBook-Pro:Chap05 ochihideji$ java CGDiamondBubble 25
```

○実行結果



○考察

自習課題として良い題材が思い浮かばなかったので、マウスによってダイアモンドパターンを描くプログラムを作成することにした。当初CGCircleを継承してプログラムを描こうとしたが、radiusの設定がうまくいかなかった。CGCanvasクラスを継承して書くと意外に煩雑になってしまった。クラス継承についてもう少し自習する必要があると感じた。

▫課題や授業に関して

○レポート作成に要した時間

4時間程度

○特に苦労した点

まだワールド座標系とスクリーン座標系について理解できていない部分が多く、プログラムがあまり整然としていないように感じる。この点に関してはこれ以降も継続的に自習していきたい。

○授業についての感想や希望

CGCardioidCircle.javaなどのプログラムで、mainメソッドでradiusの値を代入しているのは何故なのでしょうか。