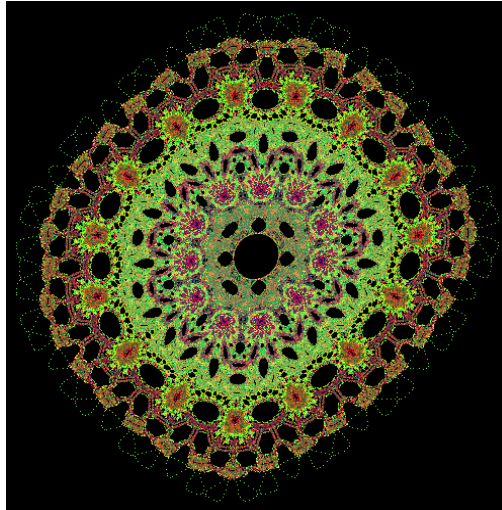



Praktikum Computergrafik, Blatt 1

* **Aufgabe 1** (Umrechnen logische – Gerätekoordinaten)

Unser Programm soll halluzinogene Muster wie dieses hier zeichnen¹:



In Moodle  finden Sie dazu die Datei `huepfer.zip` mit den Rahmenprogrammen:

HuepferFrame.java: JFrame-Klasse zur Darstellung, muss nicht editiert werden.

Huepfer.java: Klasse, in der Sie den Zeichenalgorithmus implementieren sollen.

HuepferTest.java: JUnit4-Test, muss nicht editiert werden. Identisch mit dem vom APA-Server ausgeführten Test.

In der Klasse `HuepferFrame` wird mit `width` und `height` die Größe des Gerätekoordinatensystems (GKOS) festgelegt:

(0,0) bis (width, height)

Es übergibt diese Koordinaten und die des logischen Koordinatensystems (LKOS) von

(xMin, yMin) bis (xMax, yMax)

per Konstruktor an die Klasse `Huepfer`, die für den Zeichenalgorithmus zuständig ist. Eine Methode `setPixel` zum Zeichnen einzelner Punkte existiert in `Huepfer` bereits.

a) Implementieren Sie zunächst in `Huepfer` die Methode

```
int transformX(double x)
```

die ein x_{LKOS} in ein x_{GKOS} umrechnet (s. Skript, Variante „Runden zur nächsten Ganzzahl“).

¹Quelle: A.K. Dewdney, *Scientific American*, Computer-Kurzweil, November 1986; Erfinder: Barry Marton (en. *Hopalong*).

Implementieren Sie gleichermaßen die Methode `int transformY(double y)`, die natürlich y_{LKOS} auf y_{GKOS} abbildet.

- b) In der Methode `render` der Klasse `Huepfer` ist der folgende iterative „Hüpfer“-Algorithmus umzusetzen:

```


 $x \leftarrow 0$ 
 $y \leftarrow 0$ 
for  $i \leftarrow 0, \text{num}$  do
    Zeichne  $(x, y)$ 
     $xx \leftarrow y - \text{sgn}(x) \cdot \sqrt{|b \cdot x - c|}$ 
     $yy \leftarrow a - x$ 
     $x \leftarrow xx$ 
     $y \leftarrow yy$ 
end for

```

hierbei sind die Muster-beeinflussenden Parameter a, b und c vom Konstruktor gesetzte Attribute der Klasse, genauso wie num , das die Anzahl der gezeichneten Punkte angibt. Die mathematischen Funktionen (inkl. der Signum- oder Vorzeichenfunktion sgn) finden Sie im `Math`-Package.

- c) Experimentieren Sie etwas mit den Parametern herum. Hübsche Bilder ergeben z.B.:

a	b	c	minX	maxX	minY	maxY
-3.14	0.3	0.3	-14	12	-15	11
-200	0.1	-80	-425	235	-425	235
0.4	1	0	-4	5	-4	5

Weitere Parameter finden sich hier .


- d) Färben Sie die Psychotapete ein! Wechseln Sie dazu regelmäßig (z.B. alle 100 Punkte) die Zeichenfarbe mit dem API-Methodenaufruf

```
graphics.setColor(new Color(red, green, blue));
```

mit `red`, `green` und `blue` RGB-Werten aus dem Bereich 0 bis 255.

* Aufgabe 2 (Zeichnen von Linien)

In dieser Aufgabe sollen die Algorithmen zum Zeichnen von Linien implementiert werden.

In Moodle  finden Sie dazu die Datei `lines.zip` mit den Rahmenprogrammen:

LinesFrame.java: `JFrame`-Klasse zur Darstellung.

Lines.java: Klasse, in der Sie die Zeichenalgorithmen implementieren sollen.

LinesTest.java: `JUnit4`-Test, muss nicht editiert werden. Identisch mit dem vom APA-Server ausgeführten Test.

a) Implementieren Sie zunächst in `Lines.java` die Methode

```
void drawLineEquation(int x0, int y0, int x1, int y1)
```

die die Punkte auf der Linie von (x_0, y_0) nach (x_1, y_1) nach der Formel $y = mx + b$ errechnet und zeichnet. Runden Sie die berechneten y -Werte zur nächsten Ganzzahl.

Bei richtiger Implementierung sollte die Ausführung von `LinesFrames` zwei rote Linien zeichnen.

b) Implementieren Sie dann

```
void drawDda(int x0, int y0, int x1, int y1)
```

die die Punkte auf der Linie von (x_0, y_0) nach (x_1, y_1) mit dem DDA-Algorithmus errechnet und zeichnet.

Zu beachten:

- Verwenden Sie Bitshift-Operationen für Multiplikation-/Division mit Zweierpotenzen.
- Die Konstante γ des Pseudo-Codes im Skript ist geeignet vordefiniert.

Bei richtiger Implementierung sollte die Ausführung von `LinesFrames` zwei grüne Linien zeichnen.

c) Implementieren Sie dann

```
void drawBresenham(int x0, int y0, int x1, int y1)
```

die die Punkte auf der Linie von (x_0, y_0) nach (x_1, y_1) mit Bresenham-Algorithmus errechnet und zeichnet. Sie dürfen davon ausgehen, dass die Steigung zwischen 0 und 1 liegt.

Bei richtiger Implementierung sollte die Ausführung von `LinesFrames` zwei grüne Linien zeichnen.

d) Kommentieren Sie in `LinesFrames.java` die Zeilen unter dem Kommentar „Überdeckungstests“ ein. So können Sie visuell überprüfen, ob die drei Algorithmen gleiche Ergebnisse liefern.

Fragen zur Übung (prüfungsrelevant)

- 1.) Mit einem Objekt welchen Typs zeichnet man in Java2D grafische Grundprimitive? Woher bekommt man dieses Objekt?

- 2.) Java2D hat keine Methode zum Zeichnen eines einzelnen Punktes. Wie behilft man sich?

- 3.) Welcher Punkt eines Fensters hat in Java2D die Gerätekoordinaten (0,0)?

- ☐ Punkt in der Mitte des Fensters: (width/2,height/2).
- ☐ Linker oberer Eckpunkt.
- ☐ Rechter oberer Eckpunkt.
- ☐ Linker unterer Eckpunkt.
- ☐ Rechter unterer Eckpunkt.

- 4.) Rastergrafik: Wie rechnet man bei gegebener Fensterbreite width und gegebener minimaler und maximaler Koordinate xMin und xMax ein x_{LKOS} in ein x_{GKOS} um?

- 5.) Wie spezifiziert man in Java2D RGB-Farbwerte?

- ☐ Mit drei Gleitpunktzahlen aus [0;1].
- ☐ Mit drei Ganzzahlen aus 1...100.
- ☐ Mit drei Ganzzahlen aus 0...255.
- ☐ Mit drei Strings aus #00,..., #FF, die Hexwerte spezifizieren.

- 6.) Der Shift b beim DDA-Algorithmus zum Zeichnen von Linien sollte so groß wie möglich gemacht werden.

Warum?

Dennoch kann b nicht beliebig groß gemacht werden. Warum?

Was ergibt sich als obere Grenze für b bei einer Viewport-Auflösung in y -Richtung von 1080 Pixeln, wenn die Daten mit `int` repräsentiert werden?