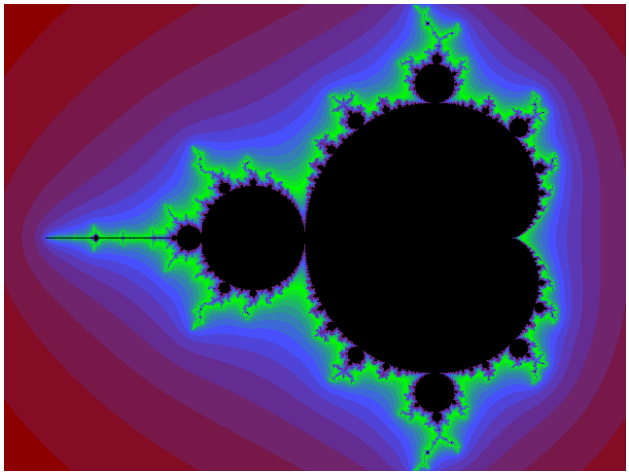
Programmieren 1 (PROG1-INF)
Übungsblatt 10



Inhaltlicher Schwerpunkt dieses Übungsblattes ist die Rekursion. Konkret geht es um die rekursive Berechnung von Fraktalen. Fraktale sind Figuren, die Selbstähnlichkeit aufweisen, d.h. bei denen Teilfiguren eine verkleinerte Kopie der Gesamtfigur sind. Sie haben sicher schon spektakuläre Beispiele wie die rechts abgebildete Mandelbrot-Menge gesehen. Die in diesem Übungsblatt zu implementierenden Fraktale sind Varianten des Pythagorasbaums und somit erheblich einfacher!

Aufgabe 1 (zum Warmwerden)

Legen Sie für dieses Übungsblatt wieder ein neues Paket an. Versuchen Sie dann aus dem Gedächtnis die Fakultät als rekursive Funktion in Java zu implementieren. Testen Sie die Funktion und verfolgen Sie auch im Debugger die Auswertung Schritt für Schritt nach. Wie oft wird die Funktion bei der Berechnung von 3! insgesamt aufgerufen?

Sie erinnern sich sicher an den Euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers. Implementieren Sie nun eine rekursive Variante. Die Methode soll die Signatur `int gcd(int x, int y)` aufweisen (für greatest common divisor) und darf *keine* Schleife zur Berechnung verwenden. Überlegen Sie, was als Abbruchbedingung zu prüfen ist, und wie Sie die Berechnung im nichttrivialen Fall auf einen erneuten Aufruf der Funktion (mit kleineren Argumenten) zurückführen können.

Aufgabe 2 (Rechenoperationen für einfache Fraktale bereitstellen)

Nun soll das Prinzip der Rekursion angewendet werden, um einfache Fraktale zu berechnen und grafisch auf dem Bildschirm darzustellen. Wir verwenden für die grafische Darstellung wieder das Paint Tool, sodass sich unsere Arbeit im Wesentlichen (aber nicht nur) darauf beschränkt, eine geeignete Controller-Implementierung bereit zu stellen.

Alle hier betrachteten Fraktale basieren auf der Konstruktion von Punkten im zweidimensionalen Raum und dem Zeichnen von grafischen Primitiven durch diese Punkte. Hier erweisen sich die früher schon einmal implementierten Vektoroperationen (insbesondere `rotate2d`) als hilfreich. Allerdings sind die Operationen dort nicht objektorientiert realisiert worden. Das können wir jetzt

noch schöner. Entwickeln Sie eine Klasse `Vector2D`, die einen zweidimensionalen Vektor von `double`-Zahlen repräsentiert. Nennen Sie die beiden dafür notwendigen Attribute `x` und `y`. Implementieren Sie neben einem Konstruktor (der zwei Parameter verlangt und damit die Attribute initialisiert) und den üblichen Getter-Methoden folgende Operationen:

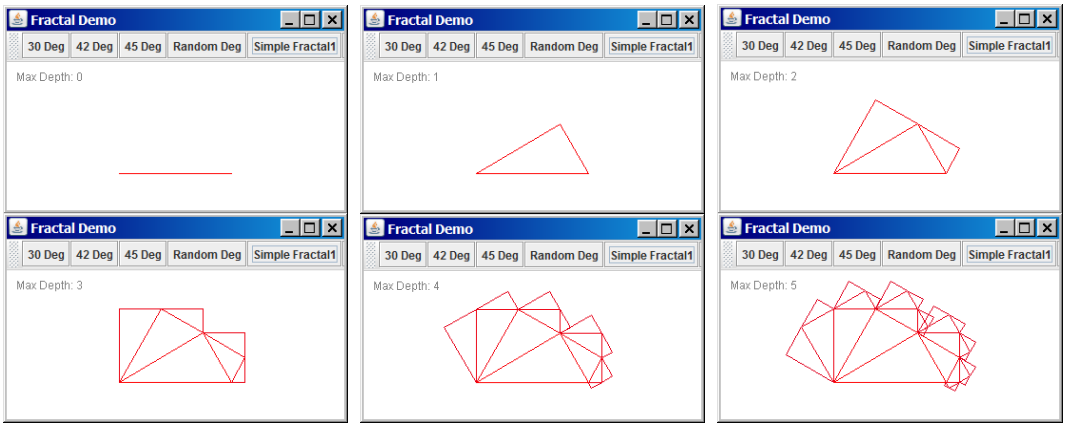
```
/**
 * Multiplies the elements of this vector with scalar factor and
 * returns the result. The vector is not changed by this operation!
 */
public Vector2D mult(double factor)

/**
 * Adds this vector and <code>vec</code> and returns the result.
 * Input vectors are not changed.
 */
public Vector2D plus(Vector2D vec)

/**
 * Subtracts <code>vec</code> from this vector and returns the
 * result. Input vectors are not changed.
 */
public Vector2D minus(Vector2D vec)

/**
 * Rotates this vector by <code>deg</code> degrees and returns the
 * result.
 */
public Vector2D rotate(int deg)

/** Returns the Euclidean norm of this vector. */
public double vlength()
```

Aufgabe 3 (Einfaches Fraktal berechnen)

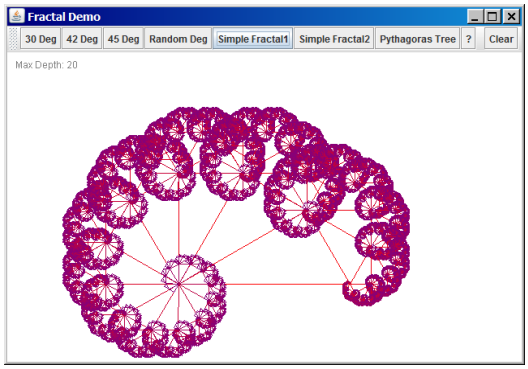
Ein einfaches Fraktal entsteht durch folgende Vorschrift:

- Zeichne eine Grundlinie zwischen zwei Ausgangspunkten.
- Zeichne auf der Grundlinie ein rechtwinkliges Dreieck, wobei der „linke“ Winkel 30 Grad beträgt.
- Benutze die beiden Katheten als Grundlinie für weitere gleichartige rechtwinklige Dreiecke.
- Zeichne auch auf deren Katheten wieder Dreiecke und fahre fort, bis ein bestimmtes Abbruchkriterium erfüllt ist.

Wir wählen als Abbruchkriterium eine maximale Rekursionstiefe (`maxDepth`). Die folgenden Bilder veranschaulichen den Konstruktionsvorgang. Das erste Bild zeigt das Ergebnis für `maxDepth=0`, das letzte für `maxDepth=5`.



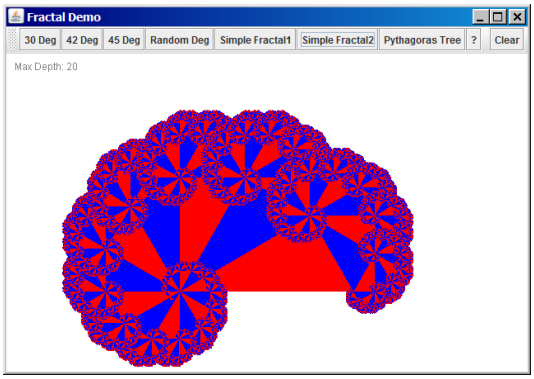
Ihre Aufgabe ist nun, neben der üblichen `main`-Methode zum Start der Applikation eine Controller-Implementierung bereit zu stellen, die bei Drücken



eines Knopfes („`Simple Fractal1`“) ein solches Fraktal mit fest vorgegebenem `maxDepth`-Wert berechnet und auf der Zeichenfläche dargestellt. Für `maxDepth=20` könnte die Ausgabe wie links abgebildet aussehen.

Tipps zur Realisierung der Applikation finden Sie bei Bedarf wie üblich im Anhang.

Aufgabe 4 (Variante des Fraktals zeichnen)

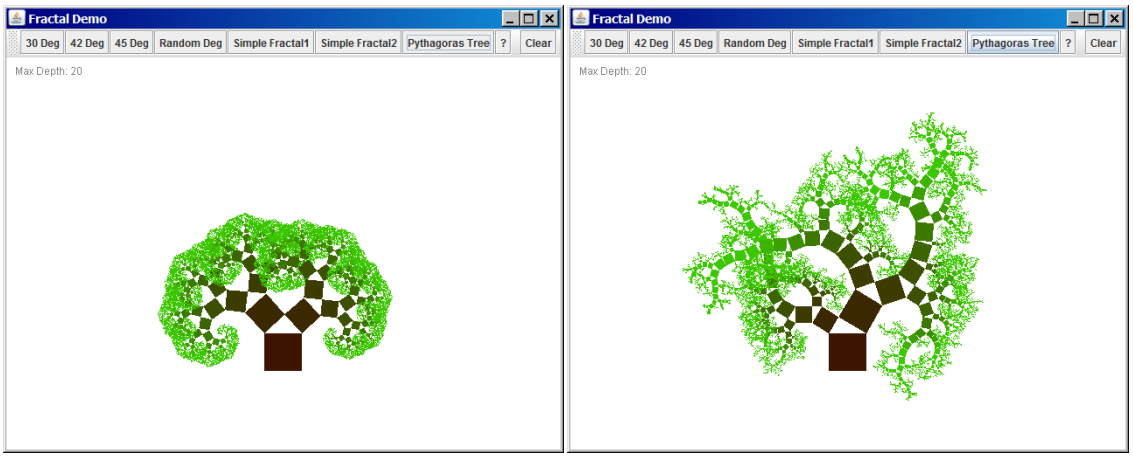


Ein hübsches Bild ergibt sich auch, wenn man die Dreiecke ausmalt, am besten mit wechselnden Farben in Abhängigkeit von der Rekursionstiefe. Fügen Sie einen zusätzlichen Knopf „`Simple Fraktal2`“ zu Ihrem Controller hinzu, der das Zeichnen dieser Fraktal-Variante auslöst.

Aufgabe 5 (Pythagoras-Baum berechnen)

Noch hübscher wird das Fraktal, wenn Sie über der Grundlinie zusätzlich zu dem Dreieck noch ein Quadrat einfügen. Ausgemalt wird dabei nur das Quadrat. Sie erhalten dann einen Pythagoras-Baum. Das Aussehen des Baumes kann über den Winkel und die Rekursionstiefe variiert werden. Links ist ein Baum mit 42-Grad-Winkeln und `maxDepth=20` dargestellt, rechts ein Baum mit zufällig gewählten Winkeln und ebenfalls `maxDepth=20`. Die Farbe der

Verbindungslinien wurde in Abhängigkeit von der Rekursionstiefe eingestellt (mit zunehmender Rekursionstiefe immer grüner!).



Implementieren Sie eine Methode zur Erzeugung von Pythagoras-Bäumen mit vorgegebenen Winkeln und fügen Sie auch entsprechende Köpfe in die Werkzeugleiste ein. Was passiert, wenn Sie die Fenstergröße verändern?

Aufgabe 6 (Benutzungskomfort steigern)

Schön wäre es, wenn man bei der schrittweisen Konstruktion der Fraktale zuschauen könnte, ähnlich wie in Aufgabe 3 dargestellt. Dazu wäre es praktisch, wenn der Benutzer die maximale Rekursionstiefe über die Aktionsknöpfe steuern könnte. Auch soll dem Benutzer die Wahl der verwendeten Winkel ermöglicht werden. Entsprechende Experimente führen bei allen drei Fraktalen zu interessanten Ergebnissen. Überlegen Sie sich ein geeignetes Bedienungskonzept und realisieren Sie es mit Hilfe zusätzlicher Knöpfe und Attribute.

Anhang

Sie benötigen für die Berechnung eines Fraktals mindestens eine Methode, die sich rekursiv aufruft. Als Parameter bieten sich neben der `PaintTool`-Instanz (zum Zeichnen benötigt) die Begrenzungspunkte der jeweiligen Grundlinie sowie die aktuelle Rekursionstiefe an.

Sinnvoll ist darüber hinaus, die Konstruktion des dritten Eckpunkts vom rechtwinkligen Dreieck als eigene Methode auszulagern. Die Konstruktion des Eckpunktes zwischen den beiden Katheten ist die eigentliche spannende Aufgabe. Wie lässt sich das machen? Tipp: Differenzvektor aus den beiden anderen Eckpunkten bilden, den um den vorgegebenen Winkel drehen und seine Länge skalieren (mit Cosinus des Winkels), und schließlich Stützvektor addieren. Bei der Drehung ist noch zu beachten, dass das Bildschirmkoordinatensystem linkshändig ist weil die x-Achse zwar wie gewohnt nach rechts, die y-Achse aber nach unten aufgetragen wird. Um bei der Drehung um den Ursprung das gleiche Ergebnis wie im rechtshändigen Koordinatensystem zu erhalten muss der Winkel daher mit -1 multipliziert werden.

Es ist sinnvoll, die Positionsberechnungen mit **`double`**-Genauigkeit durchzuführen. Deshalb wurde die `Vector2D`-Klasse auch so konzipiert. Beim Zeichnen mit den `addX`-Operationen des Paint Tools müssen diese Werte natürlich nach **`int`** gecastet werden.

Ein netter Effekt ergibt sich, wenn man die Farbe mit der Rekursionstiefe verändert. Der Konstruktor **`new`** `Color(int r, int g, int b)` liefert Ihnen einen beliebigen Farbwert, wobei die Farbanteile rot, grün und blau über Zahlen zwischen 0 und 255 frei gewählt werden können.

Sollte der Speicher knapp werden (kann passieren!), bietet der Eintrag `-Xmx500M` bzw. `-Xmx1G` im VM arguments-Feld der Run Configuration Ihrer Klasse Abhilfe. Er setzt für Ihre Applikation den Heap-Space auf 500 MB bzw. 1GB hoch.