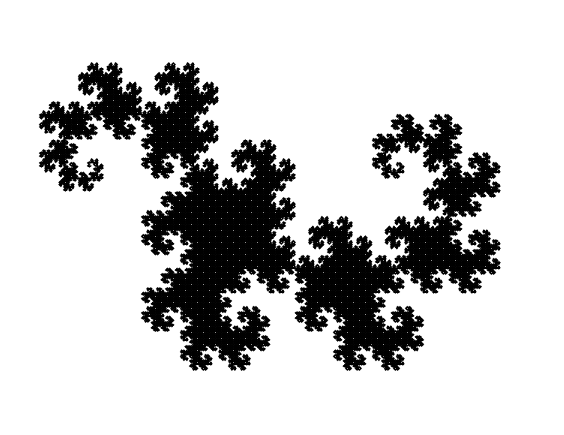
Grafische Darstellung von Fraktalen

Portfolioarbeit

Von Jonah Sebright

Klasse 6A



EF-Lehrer: PD Dr. Victor Yakhontov

Gymnasium Kirschgarten Basel 2021

## Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc71895427)

[1. Einleitung 3](#_Toc71895428)

[1.1 Was sind Fraktale? 3](#_Toc71895429)

[1.2 Ziele des Projekts 3](#_Toc71895430)

[1.3 Vorschau Resultate 3](#_Toc71895431)

[2. Hauptteil (Material und Methoden) 4](#_Toc71895432)

[2.1 Programmiersprache 4](#_Toc71895433)

[2.2 Bibliotheken 4](#_Toc71895434)

[2.2.1 StdDraw 4](#_Toc71895435)

[2.2.2 Flanagan 5](#_Toc71895436)

[2.3 Reduktion des Problems 5](#_Toc71895437)

[2.4 Entwicklung der Module 6](#_Toc71895438)

[2.5 Entwicklung der Algorithmen 6](#_Toc71895439)

[2.5.1 Schneeflocke 6](#_Toc71895440)

[2.5.2 Pfeilspitze 6](#_Toc71895441)

[2.5.3 Drachenkurve 6](#_Toc71895442)

[2.6 Testing 6](#_Toc71895443)

[3. Resultate 6](#_Toc71895444)

[3.1 Bilder 6](#_Toc71895445)

[3.2 Effizienz 6](#_Toc71895446)

[3.3 Features des Main-Programms? Oder zu einleitung?? 6](#_Toc71895447)

[4. Diskussion 6](#_Toc71895448)

[Flächedrachenkurve 6](#_Toc71895449)

[4.1 Optimierungsmöglichkeiten 6](#_Toc71895450)

[4.2 Rückblick auf das Problem 6](#_Toc71895451)

[5. Nachwort? 7](#_Toc71895452)

[6 Quellenverzeichnis 7](#_Toc71895453)

[7. Abbildungsverzeichnis 7](#_Toc71895454)

[8. Tabellenverzeichnis 7](#_Toc71895455)

[9. Quellcodeverzeichnis 7](#_Toc71895456)

[10. Anhang 7](#_Toc71895457)

[6.3 Glossar? 7](#_Toc71895458)

[6.6 Weitere Diagramme? 7](#_Toc71895459)

[6.7 Gesamter Quellcode 7](#_Toc71895460)

# 1. Einleitung

## 1.1 Was sind Fraktale?

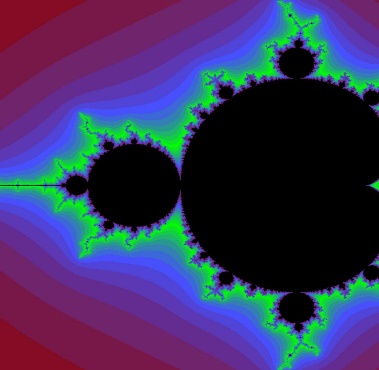


Abb. 1 Mandelbrot Fraktal ([Vater der Fraktale: Mandelbrot bewies die Schönheit der Mathematik - WELT](https://www.welt.de/wissenschaft/article10361079/Mandelbrot-bewies-die-Schoenheit-der-Mathematik.html), 2010)

Fraktale sind Geometrische Gebilde, die sich in nicht ganzzahligen („fraktalen“) Dimensionen befinden. B.B. Mandelbrot zeigte erstmals, dass sich Fraktale überall in der Naturbefinden. Fraktale werden beispielsweise verwendet um Längen von Küsten zu berechnen (vgl. [Fraktale - Lexikon der Physik (spektrum.de)](https://www.spektrum.de/lexikon/physik/fraktale/5252), kein Datum).

## 1.2 Ziele des Projekts

Das Hauptziel dieses Projekts ist, mit den gelernten Fähigkeiten und dem erworbenen Wissen ein konkretes Informatik-Problem zu lösen. Das Problem, das in diesem Projekt untersucht wurde, soll bestimmte Fraktale, wie die „Schneeflocke“, die „Pfeilspitze“ und die „Drachenkurve“ darstellen, bzw. zeichnen. Das Vorgehen beim Lösen des Problems wird ebenfalls in dieser Portfolioarbeit dar gestellt.

Zu diesem Problem soll ein Programm erstellt werden, das dem Benutzer eine Annäherung des ausgewählten Fraktals bis zu einer gewünschten Stufe grafisch wiedergibt. Zur Zeichnung der Kurve soll die Zeichnungs-Bibliothek „StdDraw“ verwendet werden. Sobald die Darstellung vollendet ist, wird der Benutzer gefragt, ob er die Zeichnung als PDF-Datei speichern möchte. Dabei kann der Benutzer den Dateinamen, selbst eingeben.

Für die grafische Darstellung der Fraktale müssen die Algorithmen der einzelnen Fraktale geschrieben werden, die alle das Basisprinzip der Rekursion implementieren. Zudem verwenden alle Algorithmen bestimmte Hilfs-Klassen, wie z.B. die *Schildkroete*.

Ausserdem werden in diesem Projekt die Resultate des Programms analysiert und mit mathematischen Erkenntnissen verglichen. Bei der Drachenkurve wird in der Diskussion bewiesen, dass die unendlich lange Kurve eine endliche Fläche bedeckt.

Struktur der Arbeit

## 1.3 Vorschau Resultate

Die Zeichnungen des Programms sind zum Staunen schön! Das Haupt-Programm läuft ohne Fehler und beinhaltet alle oben beschrieben Funktionalitäten und erfüllt somit dessen Ziele. Die einzelnen Algorithmen der Fraktale funktionieren ebenfalls einwandfrei bei beliebiger Stufe. Hier sind ein paar grafische Resultate der verschiedenen Fraktale:

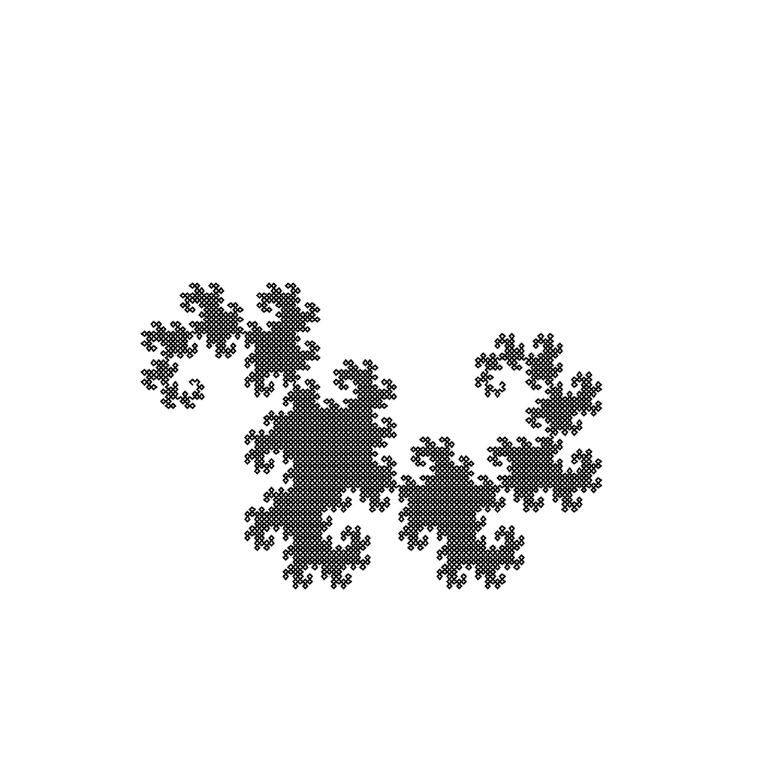


Abb. 2 Drachenkurve bei Stufe 13

Abb. 3 Schneeflocke bei Stufe 7

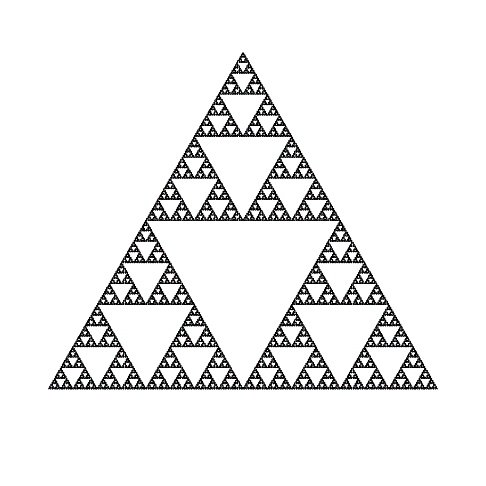
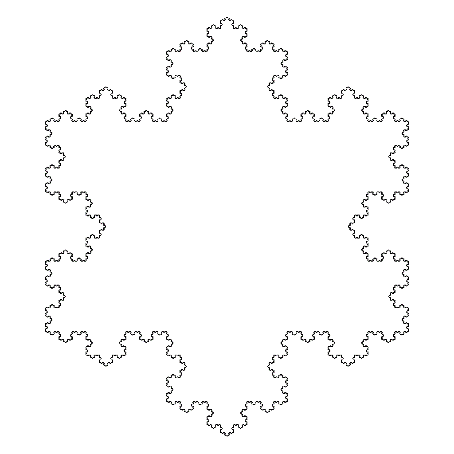


Abb. 4 Pfeilspitze bei Stufe 9

Was und wie gemacht und was wird vorgsestellt, Darstellung

Das Programm besteht aus den Modulen *Main, Schildkroete*, *Fraktal, Input* und den Bibliotheken *StdDraw* und *Flanagan* (siehe Abb. 2 Modularisierung des Programms).

Insgesamt wurden 13 Klassen erstellt mit einem Total von ca. 500 Linien Quellcode. Eine gesamte Übersicht der Abhängigkeiten Klassen ist im Anhang INSERT\_HYPERLINK.

# 2. Hauptteil (Material und Methoden)

Eine Java-Applikation ist sehr komplex und kann durch falsche Entscheidungen schnell zu Problemen führen. Geschicktes Vorgehen ist daher sehr wichtig und die Grundstrukturen müssen stimmen, bevor Details implementiert werden können. Besonders mit wachsender Grösse des Programms ist die Architektur umso wichtiger.

Begonnen wird mit der konkreten Definition des Programms, was schon in der Einleitung gemacht wurde. Danach findet die Reduktion des Problems statt, wobei die Module und deren Beziehungen und der Grobalgorithmus skizziert werden. Anschliessend können die einzelnen Module und Algorithmen konkret implementiert werden. Möglicherweise werden dafür Bibliotheken verwendet, in diesem Fall „StdDraw (siehe Kapitel 2.2.1 StdDraw). Erst in diesem Schritt wird der Quellcode verfasst.

Damit das Programm stabil und sicher ist, wurde es mit den erwarteten sowie auch unerwarteten Eingabewerten des Benutzers getestet. Dies ist wichtig, da Benutzer nicht immer die erwarteten Werte eingeben und Fehler des Programms zu vermeiden sind. Zudem muss auch die Effizienz getestet werden, sodass ein langsames Programm bemerkt und optimiert werden kann.

Für die Verwendung des Programms muss zunächst eine Verbindung zwischen dem Benutzer und dem Programm hergestellt werden. Dies wird durch ein GUI (graphical user interface) gemacht, was die Interaktion erlaubt. In diesem Programm müssen lediglich ein paar Fenster zum Einlesen und zur Resultat-Ausgabe geöffnet werden. Die Bibliothek „Flanagan“ (siehe 2.2.2 Flanagan) erfüllt diese Forderungen.

## 2.1 Programmiersprache

Wie vorgegeben, wurde das Programm ausschliesslich mit Java programmiert. Java ist eine objekt-orientierte Programmiersprache von Oracle und bietet so viele Vorteile, wie unter anderem Polymorphie (vgl. Ullenboom 2012, p .50).

## 2.2 Bibliotheken

### 2.2.1 StdDraw

Für das Programm wurde eine Zeichnungsbibliothek namens „StdDraw“ vom *Computer Science* Departement der Universität Princeton verwendet (vgl. [Computer Science Department at Princeton University](https://www.cs.princeton.edu/), 2022). Die Bibliothek bietet eine Zeichenfläche und nützliche Methoden an, die dazu dienen, die Zeichenfläche zu manipulieren. Die wichtigsten Methoden für dieses Projekt sind *line, setPenColor, setPenRadius, clear*. Eine weitere nützliche Funktion ist *save*, mit der die Zeichenfläche als Bild-Datei gespeichert werden kann.

### 2.2.2 Flanagan

Für das Einlesen der gewünschten Kurve, Stufe und Dateiname des zu speichernden Bildes wurde die Flanagan-Bibliothek von M. Thomas ([Michael Thomas Flanagan's Java Scientific Library: Input through a dialog box (ucl.ac.uk)](https://www.ee.ucl.ac.uk/~mflanaga/java/Db.html), 2010) implementiert. Mit ihr können einfache Fenster geöffnet werden. Die wichtigsten Methoden für dieses Projekt sind *optionBox*, *readInt*, *yesNo* und *readLine*.

## 2.3 Reduktion des Problems

Komplexe Systeme können in kleinere einfachere Aufgaben geteilt und so einfacher gelöst werden. Dieses Aufteilen heisst **Modularisierung**. Jede einzelne Aufgabe kann für sich gelöst werden.

Dieses Programm habe ich folgendermassen modularisiert:

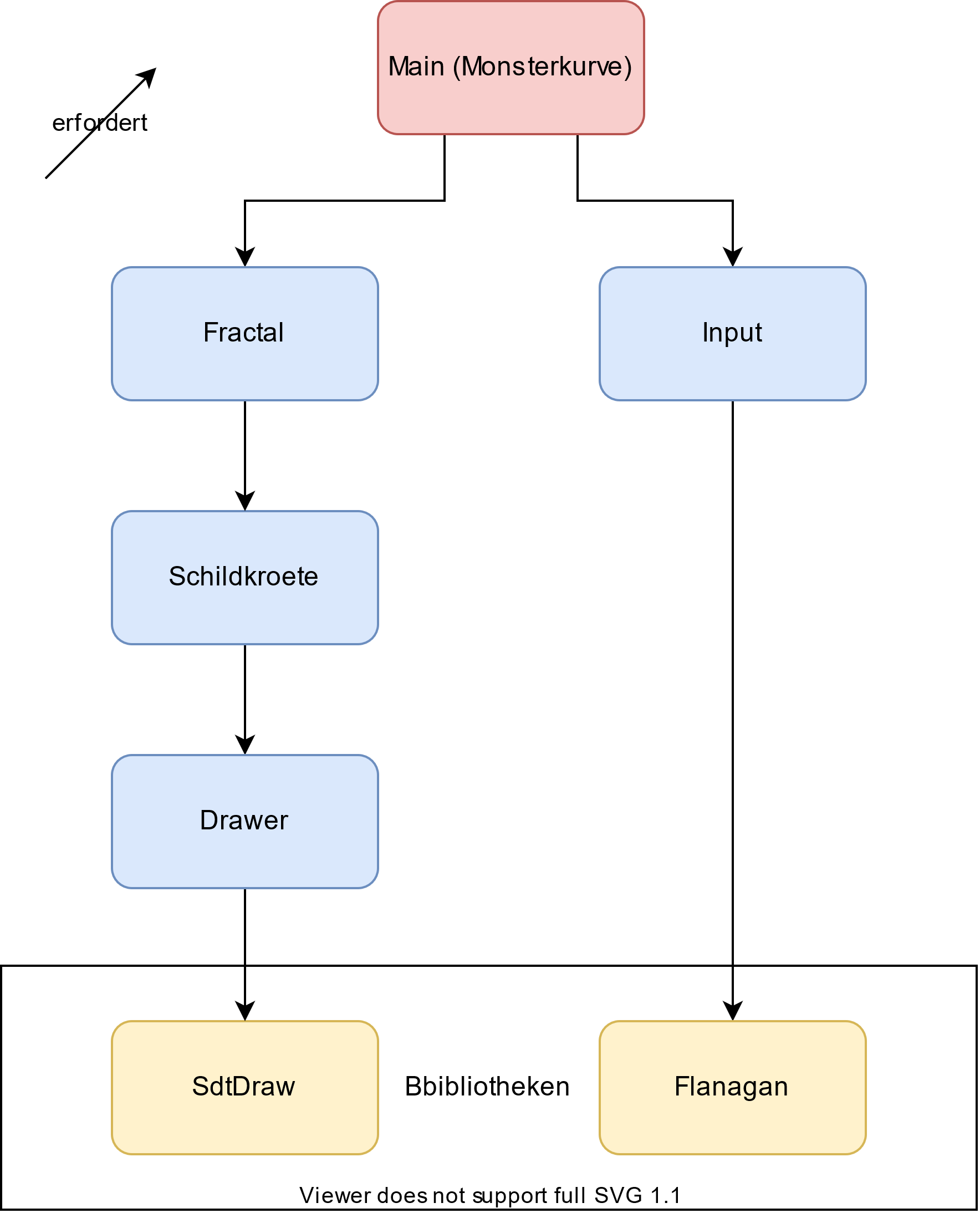


Abb. 5 Modularisierung des Programms

*Main* erfordert ein Hilfsmodul für das GUI, welches auf die Bibliothek *Flanagan* angewiesen ist. Zudem erfordert *Main* die Fraktale (Schneeflocke, Pfeilspitze, Drachenkurve), die das Modul *Schildkroete* verwenden. *Schildkroete* erfordert das Modul *Drawer*, welches unter anderem die Methode *drawLine(Point from, Point to)* bietet. Ausserdem verwendet *Drawer* die Bibliothek *StdDraw*.

Nachdem die Module entworfen sind, können sie und ihr Zusammenspiel implementiert werden. Doch dabei muss zunächst ein **Grobalgorithmus** entworfen werden. Der Grobalgorithmus des Main-Programmes ist in Abb. 1 ersichtlich.

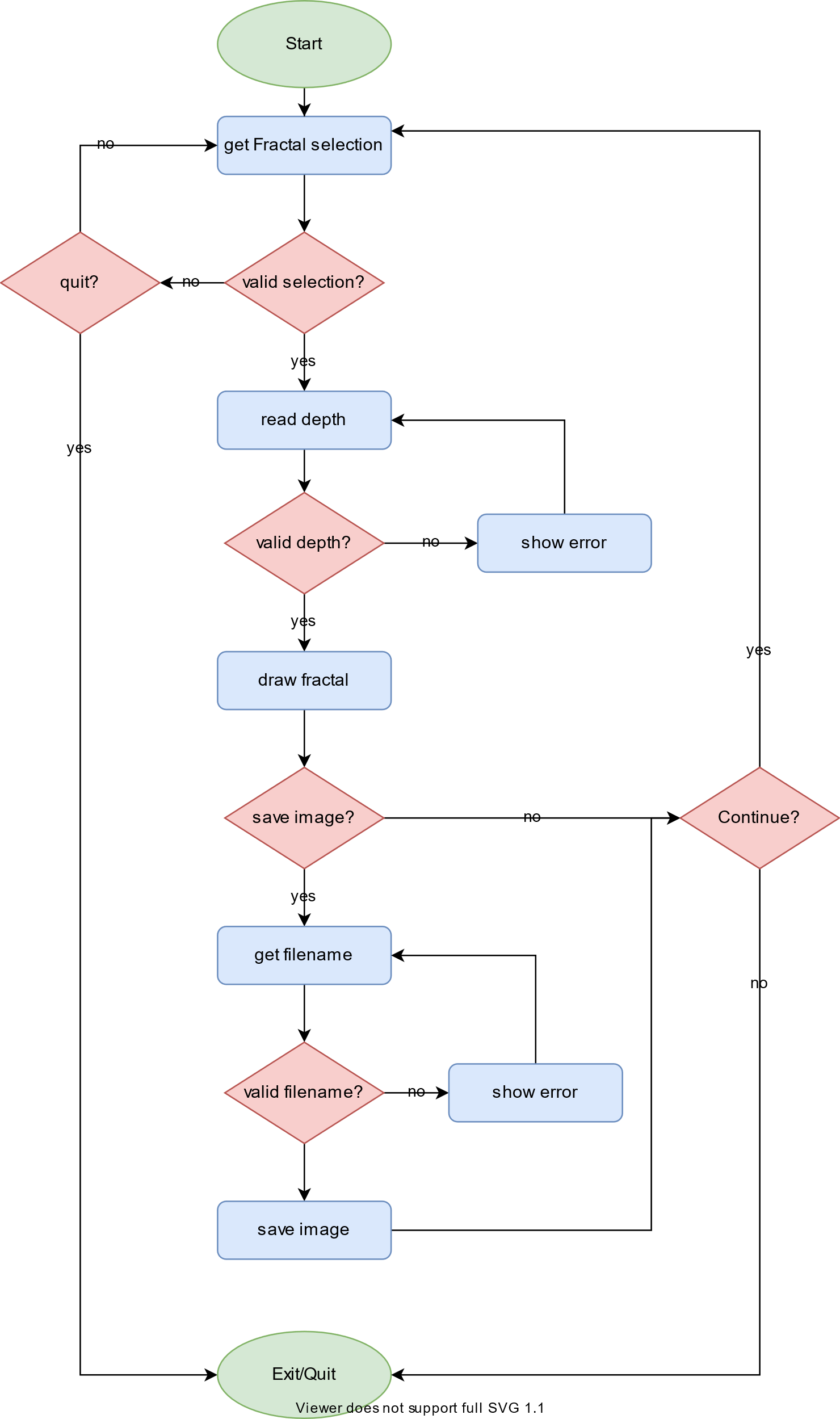


Abb. 6 Grobalgorithmus des Main Programms

Bei allen drei Fraktalen beruht der Grobalgorithmus auf das allgemeine Prinzip der Rekursion. Dies bedeutet, dass eine Methode sich selbst immer wieder aufruft, bis eine End-Bedingung erfüllt ist.

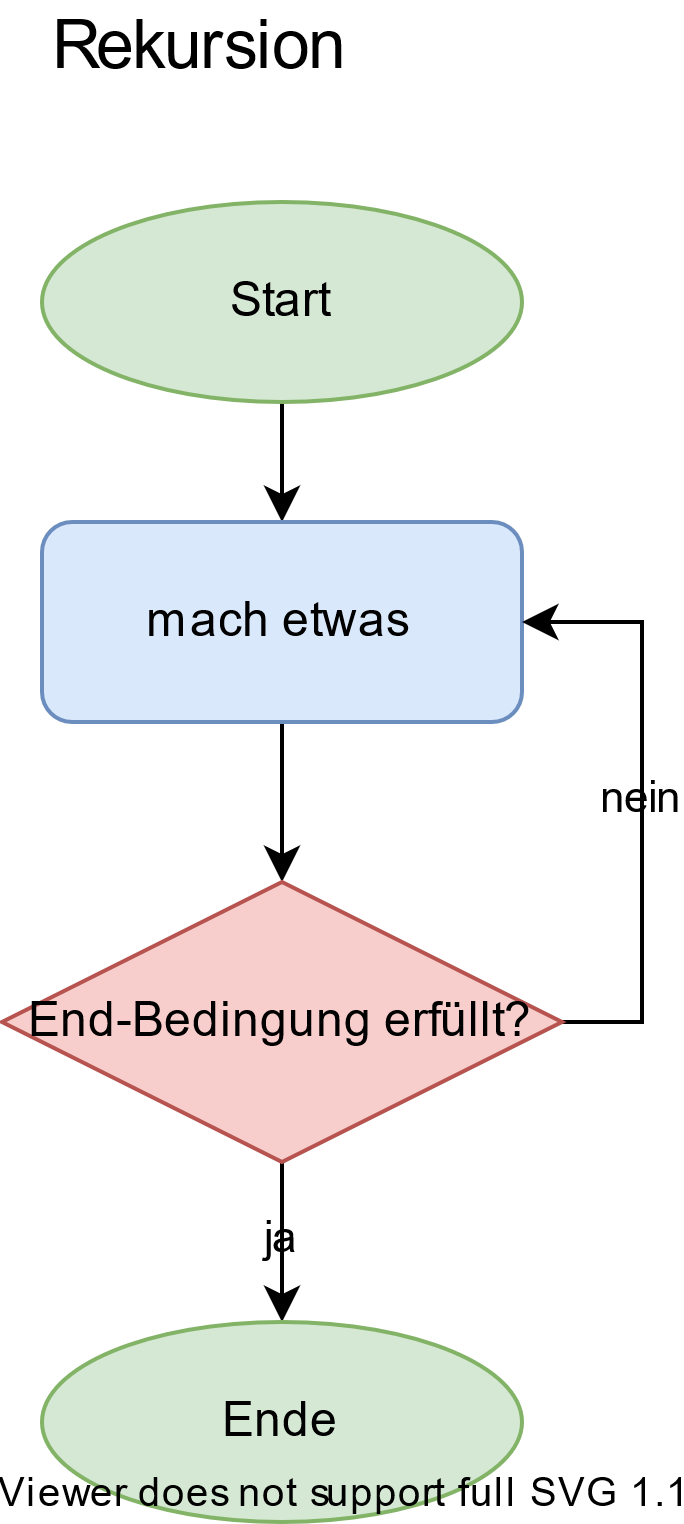


Abb. 7 Ablaufsprinzip der Rekursion

## 

## 2.4 Entwicklung der Module

Ablaufdiagramme, Beschreibung variablen

## 2.5 Entwicklung der Algorithmen

Allgemeines prinzip der Rekursion in allen 3 Algorithmen

Konkrete impl.

### 2.5.1 Schneeflocke

### 2.5.2 Pfeilspitze

### 2.5.3 Drachenkurve

## 2.6 Testing

Unit testing?

Mit ganz krummen eingangs werten

# 3. Resultate

## 3.1 Bilder

## 3.2 Effizienz

## 3.3 Features des Main-Programms? Oder zu einleitung??

# 4. Diskussion

## Flächedrachenkurve

blabla

## 4.1 Optimierungsmöglichkeiten

mathematischer

## 4.2 Rückblick auf das Problem

Was für Schwierigkeiten?

Hätte besser angehen können?

# 5. Nachwort?

# 6 Quellenverzeichnis

# 7. Abbildungsverzeichnis

# 8. Tabellenverzeichnis

# 9. Quellcodeverzeichnis

# 10. Anhang

6.3 Glossar?

6.6 Weitere Diagramme?

6.7 Gesamter Quellcode