Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Fachbereich Design Informatik Medien
Studiengang Bachelor Allgemeine Informatik

Programmieren 3 AI – Praktikum

Projektaufgabe

15 + 5 Punkte

Sebastian Flothow

Wintersemester 2011/12

Abgabe (Email):	2012-02-26 23:59
Abnahme:	2012-02-28, Raum und Uhrzeit werden auf der Veranstaltungswebsite bekanntgegeben

Die Projektaufgabe baut auf Blatt 9 auf. Zu den für Blatt 9 vorgegebenen Quelltexten (`Blatt9.tar.gz`) kommt noch das Interface `ParametrizedComplexFractal` hinzu, die Datei `parametrizedcomplexfractal.hpp` ist von der Veranstaltungsseite herunterzuladen. Ihre Lösung zu Blatt 9, zumindest die Klasse für die Berechnung der Mandelbrot-Menge, soll weiterverwendet werden.

Ziel der Projektaufgabe ist ein Qt-Programm zur interaktiven Erforschung von Fraktalen in der Gaußschen Zahlenebene, d.h. mit der Möglichkeit in das Fraktal hineinzuzoomen und sich darin zu bewegen. Das Programm soll (mindestens) zwei Fraktale darstellen können, die Mandelbrot-Menge und die Julia-Mengen für $f(z) = z^2 + c$; dabei soll auch zwischen verschiedenen Einfärbungen gewählt werden können.[1]

Abschnitt 1 enthält die nötigen Informationen über Julia-Mengen, Abschnitt 2 beschreibt die Anforderungen an das zu schreibende Programm.

Statten Sie Ihr Projektverzeichnis mit einer Qt-Projektdatei aus, die Compiler-Warnungen aktiviert. Beim Kompilieren des Programms dürfen keine Fehler oder Warnungen auftreten. Zudem muss ein Doxyfile vorhanden sein, dass aus in den Quelltexten enthaltenen entsprechenden Kommentaren eine HTML-Dokumentation aller Klassen und ihrer öffentlich sichtbaren Elemente erzeugt.

[1] Strenggenommen sind dies nicht nur zwei Fraktale, sondern unendlich viele, da unendlich viele solcher Julia-Mengen existieren; diese werden jedoch alle nach dem selben Verfahren berechnet, das lediglich durch einen Parameter variiert wird.

Weiterhin muss das Projektverzeichnis eine Datei `README.txt` mit folgenden Angaben enthalten:

- Name des Programms
- Name und Emailadresse des Autors
- Kurzbeschreibung des Programms
- Externe Abhängigkeiten (das sollten hier nur Qt und evtl. Boost-Bibliotheken sein)
- Build-Prozess (also alle nötigen Aufrufe, um aus den Quelltexten ein ausführbares Programm zu erzeugen)

Senden Sie Ihr Projekt bis zum oben angegebenen Zeitpunkt als tar.gz-Archiv an `sebastian@flothow.de`. Das Archiv muss so gepackt sein, dass es genau ein Verzeichnis enthält, dessen Name (bis auf das Suffix) mit dem des Archivs übereinstimmt. Ein solches Archiv kann durch einen Aufruf der Form `tar -czf` *`verzeichnis`*`.tar.gz` *`verzeichnis`* erzeugt werden. Rufen Sie vor dem Packen `make distclean` auf (d.h. die eingesandten Archive sollen nur Quelltexte enthalten, keine Kompilate).

Zur Abnahme ist die persönliche Anwesenheit erforderlich. Führen Sie bei der Abnahme in bis zu fünf Minuten kurz die Funktionalität und Handhabung Ihres Programms vor, und erläutern Sie die Funktionsweise, also die Berechnung der Fraktale, die Einfärbungen, aber auch die Verarbeitung von Benutzereingaben.

1 Julia-Mengen

Zu jeder Funktion $f : \mathbb{C} \mapsto \mathbb{C}$ existiert eine *Julia-Menge*[2] J_f und eine *Fatou-Menge*[3] F_f. Man betrachtet dazu durch die Funktion erzeugte Folgen: Zu jedem beliebigen Anfangswert entsteht durch wiederholte Anwendung von f eine Folge, d.h. man wählt ein $z_0 \in \mathbb{C}$ und setzt mit $z_{n+1} = f(z_n)$ die Folge fort. Eine alternative Schreibweise ist $z_n = f^n(z_0)$, wobei f^n für die n-fache Anwendung der Funktion f steht.

Einfach ausgedrückt können dann je nach gewähltem Startwert z_0 zwei Fälle unterschieden werden: Falls Werte, die dicht bei z_0 liegen, zu ähnlichen Folgen führen, gilt $z_0 \in F_f$; ansonsten, falls also schon geringfügige Änderungen des Startwerts zu einem deutlich anderen Verhalten führen, gilt $z_0 \in J_f$.

Wir betrachten im Folgenden Funktionen der Form $f(z) = z^2 + c$, dabei ist $c \in \mathbb{C}$ eine frei wählbare Konstante. Die Julia-Menge eines Polynoms hat die Eigenschaft, dass sie in der Darstellung in der Gaußschen Zahlenebene eine Teilmenge der Fatou-Menge umschließen kann; die Julia-Menge mit ihrem Inneren heißt *gefüllte Julia-Menge* K_f. Es gilt außerdem

$$K_f = \{z \in \mathbb{C} | \lim_{n\to\infty} |f^n(z)| < \infty\}$$

[2] nach Gaston Julia, 1893–1978
[3] nach Pierre Fatou, 1878–1929

d.h. für $z_0 \in K_f$ sind die zugehörigen Folgen z_n beschränkt, während für außerhalb liegende Startwerte die zugehörigen Folgen betragsmäßig gegen unendlich streben. Eine grafische Darstellung von Julia-Mengen kann daher ganz ähnlich der Mandelbrot-Menge berechnet werden, indem für jeden Pixel per Escape Time Algorithm geprüft wird ob der dem Pixel zugehörige komplexe Wert Element der gefüllten Julia-Menge ist. Auch hier können Punkte außerhalb der Menge danach unterschieden werden, wieviele Iterationen bis zum Überschreiten einer vorgegebenen Schranke benötigt werden, und abhängig davon eingefärbt werden. Ein entsprechender Algorithmus ist in Listing 1 gegeben; günstige Werte für die Konstanten sind $t = 2$ und $n_{max} = 128$, insbesondere n_{max} kann aber auch variiert werden. Abb. 1 zeigt einige damit berechnete Darstellungen von Julia-Mengen.

Listing 1: Escape Time Algorithm

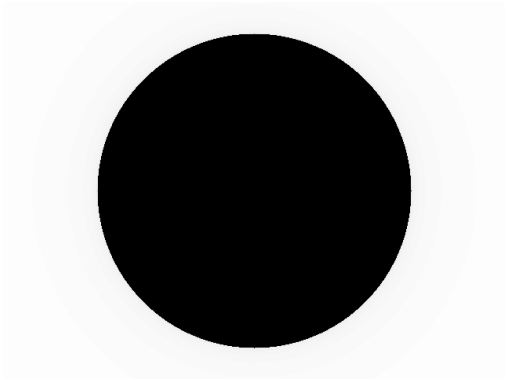
```
1  Seien z ∈ ℂ der einzufärbende Punkt, c ∈ ℂ frei wählbar jedoch für alle
       Pixel einer Darstellung gleich, t ∈ ℝ eine Schwelle für den Betrag von
        z ab der Divergenz angenommen wird, n_max ∈ ℕ eine Obergrenze für die
       Iterationshäufigkeit.
2
3  n ← 0
4
5  while |z| < t and n < n_max
6      z ← z² + c
7      n ← n + 1
8
9  Farbe des Pixels abhängig von n wählen
```

2 Anforderungen an das Programm

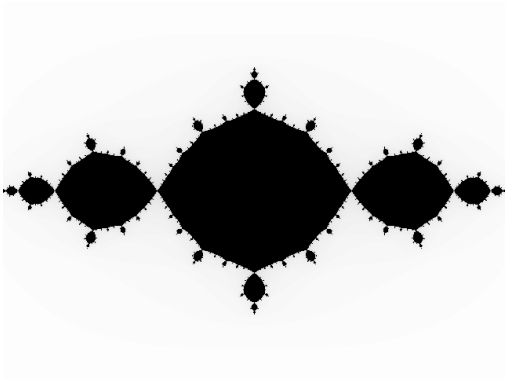
2.1 Basisanforderungen

Die Benutzeroberfläche soll im Wesentlichen aus einem Fenster bestehen. Der Großteil des Fensters soll der Anzeige des Fraktals dienen; bei Änderung der Fenstergröße soll sich dieser Teil des Fensters in der Größe anpassen. Das Fenster soll eine Menüleiste haben, darin soll sich mindestens ein Datei-Menü befinden mit der Möglichkeit eine (mit sinnvollem Inhalt gefüllte) About-Box aufzurufen und das Programm zu beenden.

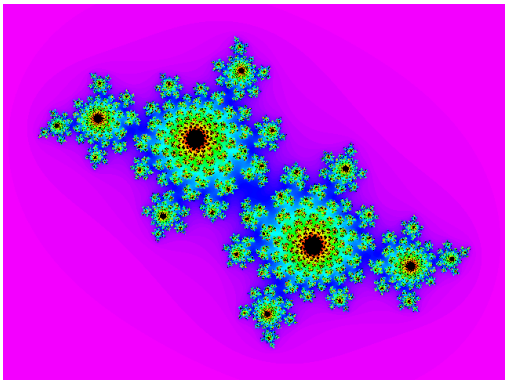
Es soll möglich sein zwischen verschiedenen Fraktalen und Einfärbungen umzuschalten. Implementieren Sie dazu eine Klasse zur Berechnung der Julia-Menge als Unterklasse von `ParametrizedComplexFractal`, wobei die in diesem Interface festgelegten Methoden `getParam` und `setParam` dem Lesen und Schreiben von c dienen sollen. Implementieren Sie weiterhin mindestens eine weitere Einfärbungsklasse (Unterklasse von `ColorMapper`), die den Wertebereich $0 \ldots 1$ auf verschiedene Farbtöne abbildet.



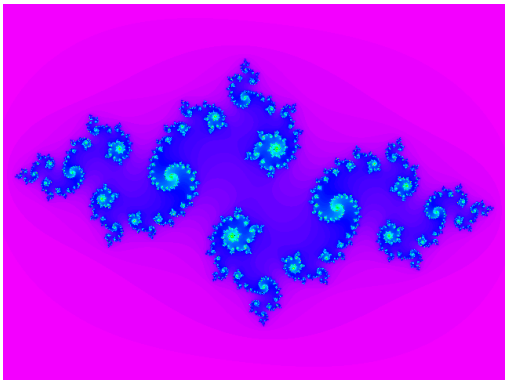
(a) c = 0, es entsteht der Einheitskreis (Radius 1, Mittelpunkt im Koordinatenursprung)



(b) c = -1



(c) c = -0,4+0,6i



(d) c = -0,8+0,2i

Abbildung 1: Verschiedene Julia-Mengen für $f(z) = z^2 + c$. Die gefüllte Julia-Menge ist jeweils schwarz dargestellt, die Punkte außerhalb weiß bzw. verschiedenfarbig eingefärbt.

Beim Start soll das Programm die Mandelbrotmenge zeigen, dabei soll der Koordinatenursprung (0+0i) im Mittelpunkt des Anzeigebereichs liegen und jeder Pixel einem Schritt von 0,005 entlang der realen bzw. imaginären Achse entsprechen.

Es soll dann möglich sein, das Fraktal zu skalieren und den dargestellten Bereich entlang der Koordinatenachsen zu bewegen. Sofern das angezeigte Fraktal einen variablen Parameter hat (also eine Unterklasse von `ParametrizedComplexFractal` ist) sollen auch Real- und Imaginärteil dieses Parameters variiert werden können; wichtig ist dabei dass dies kleinschrittig erfolgen kann (Schrittweite $< 0{,}1$), aber auch ein ausreichend großer Wertebereich genutzt werden kann.

Die Anzeigeparameter sollen einerseits intuitiv manipuliert werden können (z.B. Skalieren und Bewegen durch Klicken bzw. Ziehen im Anzeigebereich), um das Fraktal frei erkunden zu können, andererseits soll es möglich sein sie numerisch abzulesen und einzugeben, um bestimmte Ansichten gezielt reproduzieren zu können. Weiterhin soll eine

Funktion vorhanden sein, mit der die Anzeige auf den Anfangszustand (0,005/Pixel, Ursprung zentriert) zurückgesetzt werden kann.

Da das Berechnen einer Ansicht einige Zeit in Anspruch nehmen kann soll im Hauptfenster angezeigt werden, ob das Programm gerade mit einer Berechnung beschäftigt ist oder neue Eingaben entgegennehmen kann.

Achten Sie darauf, dass die Bedienung des Programms möglichst leicht verständlich ist. Sofern einzelne Aspekte der Handhabung nicht unmittelbar ersichtlich sind sollten Sie einen entsprechenden Erläuterungstext verfassen, der durch das Programm angezeigt werden kann (z.B. in der About-Box, oder, wenn er kurz genug ist, auch direkt im Hauptfenster).

Das Programm soll eine plattformübergreifend kompilierbare Anwendung sein. Verwenden sie also keine systemspezifischen Bibliotheken und Aufrufe, sondern nur Qt, die C/C++-Standardbibliothek und ggf. die Boost-Bibliothek. Beachten Sie dass die Qt-Bibliotheken einige systemspezifische Methoden enthalten, die in der Dokumentation entsprechend gekennzeichnet sind; verwenden Sie diese nicht.

Dokumentieren Sie alle Klassen mit Doxygen. Versehen Sie dazu in den Header-Dateien die Klassen selbst und (zumindest) alle öffentlichen Felder und Methoden mit entsprechenden Kommentaren. Achten Sie darauf alle Aspekte der Methoden zu dokumentieren, also Parameter, Rückgabewert und ggf. welche Exceptions in welchen Situationen geworfen werden.

Für die Umsetzung der Basisanforderungen werden bis zu 15 Punkte vergeben. Bei der Vergabe der Punkte werden auch die Konzeption des Programms, die Qualität und Kommentierung der Quelltexte sowie die Programmdemonstration bei der Abnahme berücksichtigt

2.2 Optionale Erweiterungen

Für sinnvolle Erweiterungen werden bis zu 5 Bonuspunkte vergeben, jedoch nur, wenn die Basisanforderungen ausreichend umgesetzt sind. Denkbare Erweiterungen sind z.B.:

- Zusätzliche Fraktale bzw. Berechnungsvarianten
  - Die Berechnungsvorschrift für die Mandelbrot-Menge kann abgewandelt werden zu $z_{n+1} = \bar{z}_n^2 + c$ (d.h. z wird vor dem Quadrieren zunächst konjugiert). Das resultierende Fraktal wird als *Mandelbar* oder *Tricorn* bezeichnet.
  - Die Berechnungsvorschrift für die Mandelbrot-Menge kann auch verallgemeinert werden zu $z_{n+1} = z_n^m + c$. Für $m = 2$ resultiert die gewohnte Mandelbrot-Menge, für ganzzahlige $m > 2$ entstehen die sogenannten *Multibrot-Mengen*, drehsymmetrische Fraktale mit $(m - 1)$ der Mandelbrot-Menge ähnlich sehenden Auswüchsen. Reelle Zahlen für den Exponenten liefern ebenfalls interessante Resultate, und auch komplexwertige m sind zumindest einen Versuch wert – eine Klasse für die Multibrot-Mengen könnte also das Interface `ParametrizedComplexFractal` implementieren.
  - Der Escape Time Algorithm kann relativ leicht so erweitert werden, dass auch Zwischenwerte für den Iterationszähler berechnet werden und damit ei-

ne gleichmäßigere Einfärbung entsteht; siehe: Linas Vepstas, Renormalizing the Mandelbrot Escape.[4] Diese Erweiterung sollte als separate Klasse implementiert werden, um den direkten Vergleich mit dem einfachen Escape Time Algorithm zu ermöglichen. Auch die Berechnung der Julia-Mengen kann analog verfeinert werden.

  - Denkbar wäre auch eine Klasse für die Darstellung „echter“ (anstelle gefüllter) Julia-Mengen, dies ist allerdings nicht trivial.

- Multithreading, um auch während einer laufenden Berechnung Benutzereingaben verarbeiten zu können und/oder die Berechnungen auf mehrere Prozessoren zu verteilen.

- Mehrsprachigkeit unter Verwendung der I18N-Funktionalität von Qt (mindestens zwei Sprachen für die Benutzeroberfläche mit automatischer Auswahl der geeignetsten Sprache; Default-Sprache sollte Deutsch oder Englisch sein).

- Eine Export-Funktion, um die momentane Ansicht (evtl. in höherer Auflösung) als PNG-Datei zu speichern.

Andere Erweiterungen sind nach Absprache ebenfalls möglich.

[4] `http://linas.org/art-gallery/escape/escape.html` – bei der Umsetzung in C++ aber bitte ohne `goto` arbeiten.