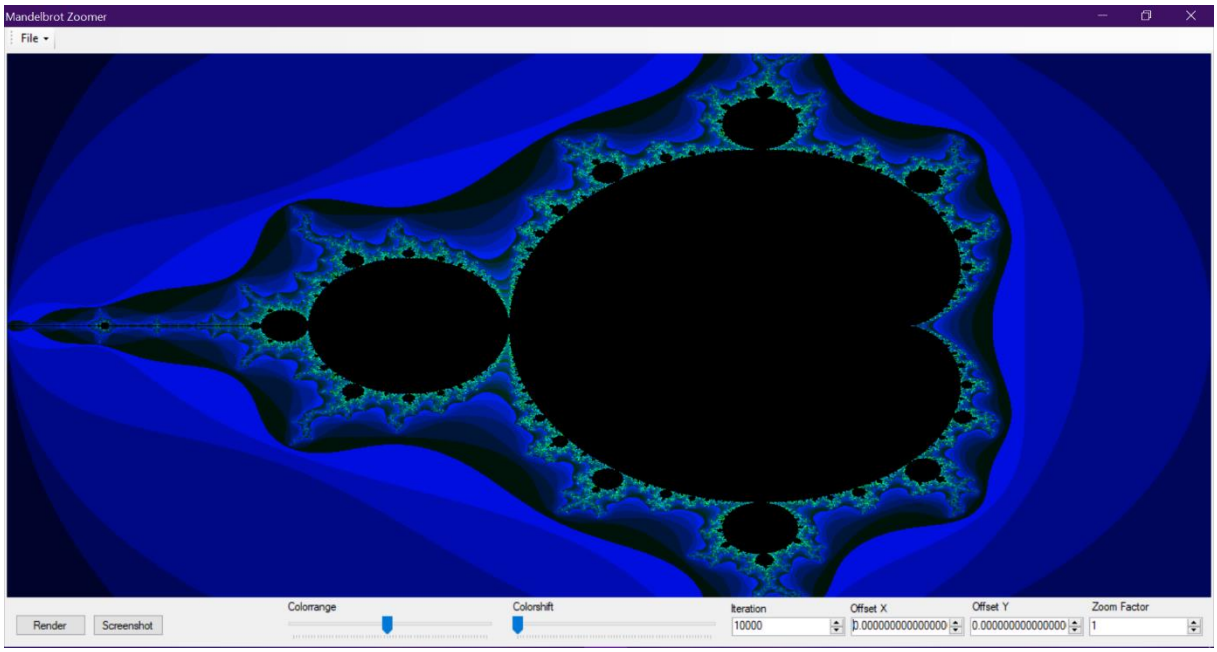
IPA Bericht

Mandelbrot Zoomer



Autor: Nicola Temporal

Version 1.0V Datum: 22.1.2021

Projekt:	Test IPA
Projektstart:	Mittwoch, 09.06.2020
Projektende:	Freitag, 22.01.2021
Auftraggeber:	Daniel Mosimann

Dokumentinformationen

Version	Datum	Name	Beschreibung
Vorlage	23.06.2013	A. Müller	Dokumentvorlage QV2013, Version V1.0
V1.0	22.01.2021	N. Temporal	Abschluss Dokumentation Test IPA

Inhaltsverzeichnis

DOKUMENTINFORMATIONEN 2

INHALTSVERZEICHNIS 3

1 EINLEITUNG 6

1.1 Inhalt und Zweck des Dokuments 6

1.2 Abkürzungen und Definitionen 6

TEIL 1: ABLAUF ORGANISATION UND UMFELD 7

2 AUFGABESTELLUNG 7

2.1 Titel der Arbeit 7

2.2 Ausgangslage 7

2.3 Detaillierte Aufgabestellung 7

2.4 Mittel und Methoden 7

2.5 Vorkenntnisse 7

2.6 Vorarbeiten 7

2.7 Neue Lerninhalte 7

2.8 Arbeiten in den letzten 6 Monaten 7

3 PROJEKTMETHODIK 8

3.1 Projektmethode 8

3.2 Szenario 8

3.3 Phasen 8

3.4 Module 8

3.5 Aufgaben 8

3.6 Projektorganisation 9

3.6.1 Projektrollen 9

4 ZEITPLANUNG 10

4.1 Zeitplan 10

5 ORGANISATION DER IPA 11

5.1 Arbeitsplatz **11**

5.2 Datensicherung **11**

5.3 Wiederherstellung **11**

6 FIRMENSTANDARDS 12

7 ARBEITSJOURNAL 13

7.1 Erster Tag: 16.1.21 **13**

7.2 Zweiter Tag: 17.1.21 **13**

7.3 Dritter Tag: 18.1.21 **14**

7.4 Vierter Tag: 19.1.21 **14**

7.5 Fünfter Tag: 20.1.21 **15**

7.6 Sechster Tag: 21.1.21 **15**

TEIL 2: PROJEKTDOKUMENTATION 16

8 KURZFASSUNG 16

8.1 Ausgangssituation **16**

8.2 Umsetzung **16**

8.3 Ergebnis **16**

9 INITIALISIERUNG 17

9.1 Analyse **17**
9.1.1 Projektziele 17
9.1.2 Anforderungen 18

9.2 Varianten **19**
9.2.1 Variante ILGPU 19
9.2.2 Variante OpenGL 19
9.2.3 Variante AleaGPU 19
9.2.4 Entscheidungsmatrix 19
9.2.5 Gewählte Variante 19

10 KONZEPT 20

10.1 Systementwurf **20**

10.2 Testkonzept **20**
10.2.1 Testziele 20
10.2.2 Teststrategie 20
10.2.3 Testrahmen 21
10.2.4 Testinfrastruktur 21

10.2.5 Testfälle 22

11 REALISIERUNG 25

11.1 System realisieren 25
11.1.1 Prototyping mit ILGPU 25
11.1.2 Erstellung der Projects 26
11.1.3 Einbauen der graphischen Ansicht eines Mandelbrots 26
11.1.4 Schieberegler mit änderbarer Farbpalette 27
11.1.5 Zoom per Links/Rechts Klick 28
11.1.6 Automatischer Zoom (Mittlerer Mausklick, Schieberegler Geschw.) 29
11.1.7 Laden/Speichern einer Position im Mandelbrot 29
11.1.8 Button Momentaufnahme Speichern 31
11.1.9 Button Momentaufnahme Drucken 32

11.2 Testprotokoll 33
11.2.1 Testfall T-001 33
11.2.2 Testfall T-002 34
11.2.3 Testfall T-003 35
11.2.4 Testfall T-004 36
11.2.5 Testfall T-005 37

12 REFLEXION 38

13 SCHLUSSWORT 38

14 LITERATUR- UND QUELLVERZEICHNIS 39

1 Einleitung

1.1 Inhalt und Zweck des Dokuments

Dieses Dokument enthält die Planung und Umsetzung des Projekts Mandelbrot Zoomer RichClient und soll die Arbeitsschritte und Vorgehen aufzeigen.

1.2 Abkürzungen und Definitionen

Abkürzung	Bedeutung
IPA	Individuelle praktische Arbeit
QV	Qualifikationsverfahren
HW	Hardware
SW	Software
Repo	Repository
Dev	Developer

Teil 1: Ablauf Organisation und Umfeld

2 Aufgabestellung

2.1 Titel der Arbeit

Mandelbrot Zoomer RichClient

2.2 Ausgangslage

Ich benötige ein Tool, mit dem ich die Mandelbrot Menge untersuchen kann. Es soll in die Menge zoomen und diese Momente in einem PNG oder Ausdruck festhalten können.

2.3 Detaillierte Aufgabestellung

Die Anwendung soll ein ansprechendes GUI aufweisen. In diesem hat der User eine eingefärbte Ansicht der Mandel Brot Menge. Der Zoom respektive das Entzoomen soll per links/rechtsklick auf die Anzeige möglich sein. Per Klick auf das Mausrad kann man auf einen beliebigen Punkt automatisch in einer bestimmbaren Geschwindigkeit beginnen herein zu Zoomen.
Unter dieser Ansicht befinden sich die Steuerelemente:

- Ein Eingabe Feld zur Anpassung der farblichen Verschiebung der Palette zur Einfärbung
- Ein Schieberegler für die automatische Zoomgeschwindigkeit
- Ein Button zur Speicherung eines gewissen Standorts im Mandelbrot
- Ein Button zum Laden eines gewissen Standorts im Mandelbrot
- Ein Button zur Speicherung als Bild
- Ein Button um die Ansicht auszudrucken

2.4 Mittel und Methoden

• .NET C#
• Projektmethodik Hermes

2.5 Vorkenntnisse

• C# RichClient Bugfixing, Entwicklung
• WinForms
•ein wenig WPF

2.6 Vorarbeiten

Entwicklungsumgebung Updaten und neue Komponenten installieren. Git Repository erstellen.

2.7 Neue Lerninhalte

Ich versuche rechnen mit einer Grafikkarte zu beschleunigen. Da ich noch keine Ahnung von Grafikkarten Beschleunigung habe muss ich hier eine simple Methode finden um das um zu setzen.

2.8 Arbeiten in den letzten 6 Monaten

Anpassung/Erweiterung/Erstellung von bestehenden Anwendungen mit php7, MySQL, Vue in der Schule. Programmieren und Anpassen von .NET Anwendungen auf dem .NET Framework und wenig mit .NET Core.

3 Projektmethodik

3.1 Projektmethode

Für dieses Projekt ist Hermes festgelegt.

HERMES ist die Projektmanagementmethode für Projekte im Bereich der Informatik, der Entwicklung von Dienstleistungen und Produkten sowie der Anpassung der Geschäftsorganisation. HERMES unterstützt die Steuerung, Führung und Ausführung von Projekten verschiedener Charakteristiken und Komplexität. HERMES hat eine klare, einfach verständliche Methodenstruktur, ist modular aufgebaut und erweiterbar.

(Entnommen aus dem Hermes Referenzhandbuch 5.1 2015)

3.2 Szenario

Ich habe das Szenario IT-Individual-Anwendung gewählt, da in der Aufgabenstellung eine neue Applikation programmiert werden muss.

3.3 Phasen

- Initialisierung
- Konzept
- Realisierung
- Einführung (nicht Teil des Auftrags)

3.4 Module

- Projektführung
- Projektgrundlagen
- Testen
- IPA-Dokumentation

3.5 Aufgaben

Aufgaben (Tätigkeiten) je Phase und Modul.

	Initialisierung	Konzept	Realisieren
Projektführung	Aufgabenstellung definieren	Variantenentscheid, Führung, Qualitätssicherung, Planung	
Projektgrundlagen	Erarbeitung Umsetzungs-Varianten		
Testen		Prototype Testing für Variantenentscheid	Testing der Anforderungen
IPA-Dokumentation	Dokument erstellen und Initialisierung Dokumentieren	Dokumentation Konzept	Dokumentation Realisierung

3.6 Projektorganisation

Da in diesem Projekt nur Nicola Temporal an der Umsetzung beteiligt ist benötig es kein spezifisches Diagramm zur Veranschaulichung der Kommunikations- und Organisationshierarchie.

3.6.1 Projektrollen

Auftraggeber:	Daniel Mosimann
Projektausschuss	Nicht vorhanden
Qualität- & Sicherheitsmanager	Nicht vorhanden
Projektleiter:	Nicola Temporal
Fachspezialist	Nicht vorhanden
Fachspezialist	Nicht vorhanden

4 Zeitplanung

4.1 Zeitplan

Zeitplan **Dauer:** 11.08.20-22.01.21 Nicola Temporal **Aktivitäten**	Aufwand SOLL	Aufwand IST	SOLL-IST	16.01.21 vm	16.01.21 nm	17.01.21 vm	17.01.21 nm	18.01.21 vm	18.01.21 nm	19.01.21 vm	19.01.21 nm	20.01.21 vm	20.01.21 nm	21.01.21 vm	21.01.21 nm
1. Initialisierung															
1.1 Szenario, Module, Aufgaben bestimmen	2.0	1.0	1.0	2.0 1.0											
1.2 Zeitplan erstellen	1.0	1.0	1.0	1.0 1.0											
1.3 Projektorganisation definieren	1.0	1.0	1.0	1.0 1.0											
1.4 Situationsanalyse (IST-Situation)	1.0	2.0	0.0		1.0 2.0										
1.5 Aufgabe/Anforderungen verfeinern	2.0	2.0	0.0		1.0 1.0 1.0 1.0										
1.6 Lösungsvarianten/Entscheidungsmatrix	4.0	3.0	1.0		1.0 1.0	2.0 1.0 2.0 0.0									
2. Konzept															
2.1 Gui Design-Template	1.0	1.0	1.0			1.0 1.0									
2.2 Vorstellung ILGPU / Prototyping	8.0	10.0	-1.0			2.0 2.0 2.0 2.0		2.0 2.0 2.0 2.0	2.0						
2.3 Hardware-Requirements	2.0	1.0	0.0					2.0 1.0							
2.4 Test Konzept	2.0	2.0	0.0					2.0 2.0							
3. Realisierung															
3.1 Mandelbrotmenge CPU-Ready generieren (ohne zoor	4.0	3.0	1.0							2.0 2.0 2.0 1.0					
3.2 Einbauen der graphischen Ansicht eines Mandelbrots	2.0	1.0	2.0							2.0 1.0					
3.3 Schieberegler mit änderbarer Farbpalette	2.0	2.0	2.0								2.0 2.0				
3.4 Zoom per Links/Rechts klick	2.0	4.0	0.0									2.0 2.0	2.0		
3.5 Automatischer Zoom (Mittlerer Mausklick, Schiebere	2.0	0.0	2.0									2.0 0.0			
3.6 Laden/Speichern einer Position im Mandelbrot	1.0	1.0	2.0										1.0 1.0		
3.7 Button Momentaufnahme Speichern	1.0	1.0	2.0										1.0 1.0		
3.8 Button Momentaufnahme Drucken	4.0	7.0	-1.0										2.0 2.0	2.0 2.0 2.0 1.0	

Total Aufwand SOLL	**42.0**			8.0	8.0	8.0	8.0	8.0	2.0
Total Aufwand IST		**43.0**		8.0	7.0	9.0	6.0	8.0	5.0
Differenz	**-1.0**								

5 Organisation der IPA

5.1 Arbeitsplatz

Geräte:

- ASUS ZenBook
- Workstation (Nvidia RTX 2080ti, Intel i7 8700K)
- HomeServer (NAS (RAID5) 13.5TB)

Anwendungen:

- VisualStudio 2019 Community Edition
- Sourcetree
- Word
- Excel
- FoxitReader
- Firefox
- AOMEI Backupper Free

5.2 Datensicherung

Die Daten werden mit der Sourcecode Verwaltung Git versioniert. Das Repository befindet sich auf GitHub, auf welches regelmässig gepushed wird. Das lokale Repo wird auf vier Geräten live synchronisiert mit Resilio Sync, darunter auch mein HomeServer. Die Archivierungsoption bei Resilio Sync ist angeschaltet, das heisst Geräte können sich beim Sync gegenseitig Files nicht überschreiben/löschen, sondern Archivieren diese. Auf meinem HomeServer werden stündlich Inkrementelle Backups der synchronisierten Daten auf ein NAS (RAID 5) mit AOMEI Backupper Free gemacht. Des Weiteren ist dieses NAS teil eines Clusters von ortsunabhängigen Geräten, welche im Brandfall oder Einbruch (ist schon passiert im Dez 2020) die Daten weiter gegen Verlust absichern.

5.3 Wiederherstellung

Bei vorhandener Historie kann Git genutzt werden, um spezifische Änderungen rückgängig zu machen oder die gelöschten Files wiederherzustellen. Falls es zu einem Datenverlust kommt, kann man über das GitHub Repo das lokale Repo wiederherstellen. Im Falle des GitHub Ausfalls oder anderweitige Komplikationen, welche das pullen vom Github Repo verhindern, kann man auf die stündlichen Backups auf meinem NAS Cluster zurückgreifen.

6 Firmenstandards

Da ich dieses Projekt nicht innerhalb meines Betriebes mache, habe ich meine eigenen Standards gesetzt, nach welchen ich das Projekt umsetze.

Strickte Schichtentrennung

Die Applikation soll nach MVC Standard aufgebaut werden.

Objektorientierte Programmierung

Die Applikation soll nicht Prozedurale Programmierung verwenden und soll mit Objekten mit ihren zuständigen Aufgaben und Attributen / Properties klar umgesetzt werden.

Verwendung von Librarys

Der selbstgeschriebene Code soll nicht das Rad neu erfinden, sondern bestehende Librarys aus dem NuGet Packet Manager einsetzen, um den selbstgeschriebenen Code auf ein Minimum zu beschränken.

7 Arbeitsjournal

7.1 Erster Tag: 16.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Szenario, Module, Aufgaben bestimmen	Initialisierung	2	1
Zeitplan erstellen	Initialisierung	1	1
Projektorganisation definieren	Initialisierung	1	1
Situationsanalyse (IST-Situation)	Initialisierung	1	2
Aufgabe/Anforderungen verfeinern	Initialisierung	2	2
Lösungsvarianten/Entscheidungsmatrix	Initialisierung	1	1
Total		***8***	***8***
Wissensbeschaffung			
Einlesen in Hermes, Finden von Möglichkeiten die Grafikkarten-Beschleunigung umzusetzen			
Hilfestellungen			
Verschiedene Dokumente zu Hermes die Herr Mosimann uns auf dem OneNote zur Verfügung gestellt hat. Darunter vor allem das Hermes Referenzhandbuch.			
Reflexion			
Es ist sehr schwer einzuschätzen, wie man Hermes umsetzen soll, da es fast zu viel Spielraum lässt. Ich konnte trotzdem im Zeitplan bleiben.			

7.2 Zweiter Tag: 17.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Lösungsvarianten/Entscheidungsmatrix	Initialisierung	3	2
Gui Design-Template	Konzept	1	1
Vorstellung ILGPU / Prototyping	Konzept	4	4
Total		***8***	***7***
Wissensbeschaffung			
Ich musste mich informieren, wie die verschiedenen Librarys / Technologien funktionieren. Dazu habe ich viel Zeit investieren müssen, da die Informationen sehr verstreut waren bei AleaGPU und OpenGL.			
Hilfestellungen			
AleaGPU Dokumentation, ILGPU Dokumentation, OpenGL Documentation			
Reflexion			
Ich konnte durch die gute Dokumentation von ILGPU schnell zu einem klaren Favoriten in meinen Lösungsvarianten kommen.			

7.3 Dritter Tag: 18.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Vorstellung ILGPU / Prototyping	Konzept	4	6
Hardware-Requirements	Konzept	2	1
Test Konzept	Konzept	2	2
Total		***8***	***9***
Wissensbeschaffung			
Ich habe mir ein Sample zur Mandelbrot Berechnung heruntergeladen, um mich mit dem Kernel und den Basics von ILGPU auseinander zu setzen.			
Hilfestellungen			
ILGPU Dokumentation, ILGPU Sample Apps			
Reflexion			
Heute musste ich länger arbeiten, da ich länger als erwartet am Prototype hatte. Es war ein sehr glücklicher Zufall, dass in den Sample Apps im ILGPU Repo sogar eine Mandelbrot Berechnung dabei war, das hat mir enorm geholfen und Arbeit abgenommen.			

7.4 Vierter Tag: 19.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Mandelbrotmenge CPU-Ready generieren (ohne zoom/verschieben)	Realisierung	4	3
Einbauen der graphischen Ansicht eines Mandelbrots	Realisierung	2	1
Schieberegler mit änderbarer Farbpalette	Realisierung	2	1
Total		***8***	***8***
Wissensbeschaffung			
Zur Verschiebung und Einschränkung der Farbe musste ich den Aufbau eines 32bit Farbcodes kennenlernen.			
Hilfestellungen			
Wikipedia Artikel zu 32bit Farbe			
Reflexion			
Da ich beim Rendern gemerkt habe das die Berechnung des Mandelbrots doch recht lange geht, habe ich die Realisierung der Grafik Komponente verkürzen können, da für die Anforderungen die WinForms Komponente PictureBox ausreicht. Ich habe mir Gedanken gemacht, ob das Automatische Zoomen so noch seinen gewünschten Effekt hat oder ob ich das Feature streichen soll.			

7.5 Fünfter Tag: 20.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Zoom per Links/Rechts klick	Realisierung	2	4
Automatischer Zoom (Mittlerer Mausklick, Schieberegler Geschw.)	Realisierung	2	0
Laden/Speichern einer Position im Mandelbrot	Realisierung	1	1
Button Momentaufnahme Speichern	Realisierung	1	1
Button Momentaufnahme Drucken	Realisierung	2	2
Total		***8***	***8***

Wissensbeschaffung

Ich musste in der Microsoft Dokumentation nach einer Lösung suchen, wie ich herausfinden kann wo auf dem Mandelbrotbild im PictureBox Control sich der Cursor befindet.

Hilfestellungen

Microsoft Dokumentation

Reflexion

Das Berechnen des Zooms und das Verschieben der Koordinaten zum Cursor hat sich als eine Mathematisch recht schwere Aufgabe erwiesen und hat mich mehr Zeit als erwartet gekostet. Das Feature für den automatischen Zoom musste ich entfernen, da es nicht die Anforderungen erfüllen kann und so nicht mehr praktikabel ist. Das Drucken eines Bildes (fullpage) hat sich als schwer herausgestellt, da die Grösse des Bildes manuell berechnet werden muss und nicht mit der Print Komponente, die ich verwende, mitgeliefert wird.

7.6 Sechster Tag: 21.1.21

Tätigkeiten	Projektphase	Zeit SOLL	Zeit IST
Button Momentaufnahme Drucken	Realisieren	2	5
Total		***2***	***5***

Wissensbeschaffung

Ich habe unzählige Artikel von Microsoft durchstöbert nach den Schnittstellen, die ich zur Berechnung der Höhe und Breite des Bildes benötige. Ich wurde zwar fündig, aber das Resultat war nicht akzeptabel. In einem Stackoverflow Artikel wurde ich fündig, da jemand das Problem schon vor mir hatte und so konnte ich den Code übernehmen.

Hilfestellungen

Microsoft Dokumentation, Stack Overflow

Reflexion

Ich habe aufgrund der Suche nach einer Lösung, wie ich das Bild (fullpage) drucken kann viel Zeit überzogen. Dank eines Stackoverflow Artikels war ich in der Lage das Problem zu Lösen.

Teil 2: Projektdokumentation

8 Kurzfassung

8.1 Ausgangssituation

Auf YouTube gibt es viele Videos, die in das Mandelbrotset kontinuierlich hinein zoomen, jedoch gibt es keine Tools die einem selbst das ermöglichen. Das Tool soll den gleichen Effekt erzeugen und zudem als Explorationstool verwendet werden können, um einen guten Punkt zum Zoomen und die richtigen Farbeffekte zu finden.

8.2 Umsetzung

Das Projekt wurde mit Hermes umgesetzt ohne Einführungsphase, da diese kein Bestandteil der Test-IPA ist. Um einschätzen zu können, ob das Projekt im Zeit-Rahmen umsetzbar ist wurde ein Zeitplan erstellt und die einzelnen Features, sowie die Planung des Projekts geschätzt und im Zeitplan eingetragen.

Es wurde möglichst im Zeitplan gearbeitet und Überstunden eingelegt im Falle kleiner Abweichungen. Für den Kompletten Zeitplan wurde ein Soll Ist geführt, um dies im Auge zu behalten. Um das GUI den Anforderungen entsprechend zu designen wurde ein Template in WinForms ausgearbeitet, damit dies im Developmentprozess nicht beginnt abzuweichen.

Es wurde nur auf C# .Net Basis auf Win10 programmiert, was die Wartung durch geringe Technologie-Diversität erleichtert.

8.3 Ergebnis

Der Mandelbrot Zoomer konnte nicht nach allen Anforderungen umgesetzt werden, aufgrund von begrenzter Hardware Geschwindigkeit. Das Feature Automatisch Zoomen musste entfernt werden, da der gewünschte Video Effekt nicht erzielt werden kann, wenn eine angemessene Zahl an Mandelbrot-Iterationen durchgeführt werden soll. Das Berechnen eines 1920x1080 Bild ist in unter einer Sekunde möglich bei 10000 Mandelbrot-Iterationen. Das gesehene Bild kann abgespeichert und nach Wunsch gedruckt werden. Per Links/Rechts-Klick auf das gerenderte Bild kann auf einen bestimmten Punkt Hinein/Herausgezoomt werden. Wenn man den bestimmten Punkt an dem man sich befindet festhalten möchte, kann dieser in einem .mb File gespeichert und wieder geladen werden.

Es wäre zu empfehlen die Applikation, um weitere Performance Optimierung Methoden zu erweitern, um das Automatische-Zoomen immersiv zu ermöglichen. Dies ist zeitlich aber nur schwer einschätzbar und sollte mit Vorsicht in der Planung angegangen werden.

9 Initialisierung

9.1 Analyse

Aktuell gibt es keine gute Möglichkeit das Mandelbrot frei zu erforschen, ohne sich eine eigene Applikation zu schreiben. Inspiriert von vielen YouTube Videos, die in das Madelbrotset hinein Zoomen, habe ich mich dazu entschieden eine RichClient Applikation zu schreiben, welche genau dies ermöglichen soll.

9.1.1 Projektziele

9.1.1.1 Persönliche Ziele

- Trotz meinem persönlich gesetzten knappen Zeitrahmen, das Projekt gewissenhaft umsetzen.
- Fokussiert auf das Projekt arbeiten und jegliche äussere Einflüsse ausblenden.

9.1.1.2 Fachliche Ziele

- Saubere Umsetzung der Schichten-Trennung
    - Die Trennung von Presentation, BusinessLayer und DataAccessLayer sollen klar ersichtlich und umgesetzt werden, damit Erweiterungen oder Fixes einfach in Zukunft gemacht werden können.
- Die Applikation so performant wie möglich umsetzen
    - Die Applikation soll die Ergebnisse des Rendering so schnell wie möglich anzeigen können, um eine möglichst flüssige Userexperience zu erreichen.
- Software Ergonomie
    - Das Verwenden der Applikation soll selbsterklärend sein, da keine zusätzlichen Schulungsunterlagen erstellt werden. Die Hardware Requirements sollen möglichst von vielen Geräten unterstützt werden, damit User nicht zusätzliche Hardware kaufen müssen.

9.1.2 Anforderungen

9.1.2.1 Funktionale Anforderungen

- Der User muss seine Aktuelle Position im Mandelbrotset immer im GUI ablesen können.
- Der User muss eine Veranschaulichung des Mandelbrotset innerhalb der Forms haben.
- Das Offset und der Zoom Factor muss eingeschränkt werden damit man nicht das Mandelbrotset verlässt oder zu weit herauszoomt.
- Der User muss eine Möglichkeit zum Drucken der Anzeige auf einer vollen A4 Seite haben.
- Der User muss mehrere Positionen im Mandelbrotset (Offset X / Y Zoom Factor) als .mb File Speichern und Laden können.
- Das Drucken muss eine Auswahl an Druckern zur Verfügung stellen.
- Das Speichern und Laden eines .mb Files soll erleichtert werden durch einen FileDialog, welcher auf die Fileextension filtert.
- Das Form soll verkleinerbar und vergrösserbar sein und die Grösse der Anzeige Mandelbrotsets dynamisch mit vergrössern/verkleinern.
- Der User muss das gezeigte Mandelbrot farblich verschieben und einschränken können.
- Colorshifting/Colorrange soll das Bild sofort ändern, um dem User gleich zu zeigen was seine Änderungen bewirken.
- ILGPU muss so eingesetzt werden das nicht nur auf einer Grafikkarte gerechnet werden kann, sondern auch auf der CPU falls keine CUDA Kompatibilität vorhanden ist.
- Der User muss die Anzeige des Mandelbrots als Bilddatei speichern können.

9.1.2.2 Nicht Funktionale Anforderungen

- Schichten Trennung nach MVC
- Schichtentrennung von Presentation Layer, Business Layer, DataAccess Layer
- Objektorientierte Programmierung

9.2 Varianten

Für die Grafikkarten Beschleunigung musste ich mich mit verschiedenen Varianten auseinandersetzen.

9.2.1 Variante ILGPU

ILGPU ist eine Open-Source Grafikkarten Beschleunigungs-Library welche auf C# basiert und C#-Code direkt in ein Kernel kompiliert, welches auf der CPU oder auf der GPU ausgeführt werden kann. Das erleichtert das Programmieren enorm, sowie das Troubleshooting und Testing. Es fehlt definitiv nicht an Beispiel-Kernels, die man sich herunterladen und ausprobieren kann und ermöglicht so das einfache erlernen der Verwendung von ILGPU.

9.2.2 Variante OpenGL

Ist eine Grafikkarten Engine welche auf C++ und einem eigenen Dialekt basiert. Hier ist der Ramup enorm hoch, da ich keine Erfahrung mit OpenGL sowie C++ habe. Dazu kommt das man mit der exklusiven Ausführung auf der GPU systemtechnisch limitiert ist. Ein grosser Vorteil ist die Performance des Renderings, was gleich verbunden werden könnte mit dem Rendering des Bildes in der Applikation.

9.2.3 Variante AleaGPU

Ist analog zu ILGPU eine Open-Source Grafikkarten-Beschleunigungs-Library, welche auch auf C# geschriebenen Kernels basiert von Nvidia selbst. Die Dokumentation und Beispiele sind hier nicht sehr einfach aufzufinden oder nicht sehr aussagekräftig. Das letzte Update war am 11.2017 und wird wohl nicht weiter unterstützt. Es ist nicht spezifiziert, ob der Code analog auf der CPU ausgeführt werden kann.

9.2.4 Entscheidungsmatrix

Skalierung: 1-10

	ILGPU	OpenGL	AleaGPU
Ease of Use	9	1	6
Performance	7	10	7
Flexibility	8	7	5
Support	10	10	4
Total	**34**	**28**	**22**

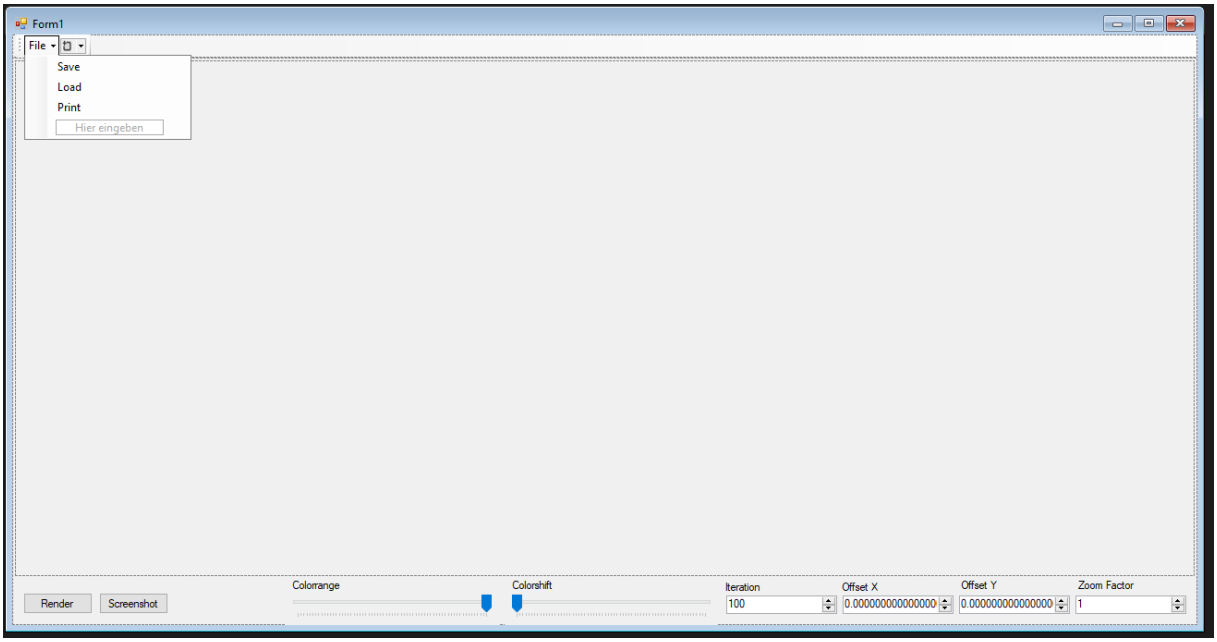
9.2.5 Gewählte Variante

Ich habe ILGPU gewählt für das ganze Packet. Man merkt das dies eine Library von Devs für Devs ist ohne grosse Verkomplizierung, einer unglaublich grossen Auswahl von Beispielen und einer super Dokumentation. So kann ILGPU mit einem Rampup punkten, der schon fast Copy Paste und 10min studieren gleichkommt. Dazu kommt die Flexibilität, das Kernel auf der GPU genauso auf der CPU laufen lassen zu können, mit einem vernachlässigbaren Perfomance Verlust. Eine einfache Wahl zu treffen.

10 Konzept

10.1 Systementwurf

Die Anforderungen geben klar vor was das GUI können muss. Also habe ich auf Grund der Anforderungen ein Mockup GUI erstellt, welches alles Funktionen oberflächlich abbildet. Mit diesem kann man das Backend hinter den verschiedenen Controls gezielt beginnen zu programmieren.



10.2 Testkonzept

10.2.1 Testziele

Die Tests sollen das GUI auf seine Funktionalitäten testen. Es müssen alle Tests erfolgreich durchgeführt werden können und sicherstellen, dass die gestellten Anforderungen an die Applikation erfüllt werden.

10.2.2 Teststrategie

Die Tests verwenden hauptsächlich die Greybox-Methodik. Es wird verzichtet genauer in das System hineinzuschauen, da das GUI alle Funktionalitäten im Hintergrund der Applikation betätigt. Die Testfälle beziehen sich auf die Anforderungen und es wird mit einem LargestFeatureFirst Testvorgehen gearbeitet, um die Potenziell zeitlich grössten Probleme zuerst zu erkennen, damit die Planung so akkurat wie möglichgehalten werden kann. Im Fehlerfall werden im Backend zusammenhängende Tests vorgezogen getestet, um Probleme an bestimmten Stellen im Code konsolidieren zu können. Anschliessend werden die Bugs gefixt bevor weiter getestet wird. Diese Fehlerhaften Tests werden nach jedem Release erneut getestet mit dem gleichen Vorgehen im Falle eines erneuten Fehlschlags.

10.2.3 Testrahmen

Voraussetzung ist das der Dev jeden Test selbst smoke-testet auf potenzielle unbehandelte Exceptions. Die endgültigen Tests sollen möglichst nur noch das richtige Verhalten der Applikation Testen.

Testfailure-Levels:

- 3 - Not Working (Unhandled Exceptions)
- 2 - Not Working Correctly (Unexpected Behavior)
- 1 - Success (Expected Behavior)

Bei einem Testfailure-Level 3 werden Tests sofort abgebrochen. Bei einem Level 2 sollen Test solange weitergeführt werden, bis man alle Kombinationen / Varianten des Testfalls durchgearbeitet wurden, um möglichst alle Probleme zu konsolidieren.

Die Tests werden nach der erstmaligen Fertigstellung der Applikation durchgeführt durch Nicola Temporal.

10.2.4 Testinfrastruktur

Das GUI ist eine portable Applikation und verwendet keine externen Daten. Die Applikation benötigt bestimmte Hardwarevoraussetzungen, um bestimmte Features zu testen. Dazu wird ein Computer benötig, welcher eine CUDA basierte Grafikkarte installiert hat. In meinem Fall habe ich dazu meine Workstation verwendet, welche eine RTX2080 TI eingebaut hat. Die Applikation muss aber auch ohne Grafikkarte verwendbar sein, um dies zu testen habe ich mein Notebook ASUS ZenBook verwendet.

10.2.5 Testfälle

ID / Bezeichnung	*T-001 Mandelbrot Rendering*
Beschreibung	In diesem Test muss geprüft werden, ob das Mandelbrot korrekt in der PictureBox angezeigt wird, wenn der Render Button geklickt wird oder das Form resized wird.
Testvoraussetzung	Eine Workstation mit CUDA support sowie eine Workstation ohne CUDA Support. Bilder des Mandelbrotsets zum Vergleich mit dem Ergebnis der Applikation.
Testschritte	1. Starten der Applikation 2. Klick auf den Render Button 3. Vergrösserung des Forms 4. Verkleinerung des Forms auf die minimale Grösse
Erwartetes Ergebnis	Beim zweiten Schritt soll das Mandelbrotset korrekt grafisch abgebildet werden und entsprechend der Grösse des Forms angepasst werden. Beim dritten Schritt soll das Angezeigte Mandelbrot korrekt mit dem Form vergrössert werden und soll sich immer noch innerhalb des gegeben Rahmens befinden. Beim vierten Schritt soll das Form wie im dritten Schritt das Bild immer noch den gegebenen Rahmen ausfüllen. Das Form soll blockieren kleiner gemacht zu werde,n als die Controls Platz einnehmen. Die Controls dürfen sich nicht beginnen zu überlappen.

ID / Bezeichnung	*T-002 Zooming*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation korrekt zoomt/entzoomt auf einen angeklickten Punkt im Mandelbrotset in der PictureBox.
Testvoraussetzung	T-001 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Mehrmals links klicken auf einen Punkt im Mandelbrotset in der PictureBox an dem man sich gut orientieren kann. 4. Mehrmals rechts klicken auf einen Punkt Mandelbrotset in der PictureBox an dem man sich gut orientieren kann. 5. Mehrmals einen Rechtsklick auf die PictureBox bis man auf den Zoom Factor 1 gelangt und dies dann nochmals wiederholen.
Erwartetes Ergebnis	Beim dritten Schritt muss überprüft werden, ob bei jedem Klick das Bild sich auf den angeklickten Punkt zentriert und in das Bild hineinzoomt. Das Zoom Factor NumericUpDown Control soll den Zoom Factor korrekt aktualisieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren Beim vierten Schritt muss überprüft werden, ob bei jedem Klick das Bild sich auf den angeklickten Punkt zentriert und aus dem Bild hinauszoomt. Das Zoom Factor NumericUpDown Control soll den Zoom Factor korrekt aktualisieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren Der Zoom Factor soll nicht das entsprechende Limit 1 unterschreiten und soll aber immer noch auf den Punkt der rechtsgeklickt wurde zentrieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren.

ID / Bezeichnung	*T-003 Colorshifting / Colorrange*
Beschreibung	Dieser Test überprüft, ob die farbliche Verschiebung korrekt funktioniert
Testvoraussetzung	T-001 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Die Colorshift Trackbar soll auf diverse Werte gesetzt werden sowie die Colorrange Trackbar.
Erwartetes Ergebnis	Beim dritten Schritt bei Betätigung der Colorshift Trackbar soll eine farbliche Verschiebung des Bildes durch das ganze Farbspektrum möglich sein sowie die Einschränkung auf eine bestimmte Farbliche Diversität.

ID / Bezeichnung	*T-004 Save / Load*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation den Zoomfaktor und das Offset Y sowie Offset X korrekt abspeichern kann und wieder laden
Testvoraussetzung	T-001, T-002 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Testen des Speicherns / Ladens (Dieser Schritt soll mind. 3-mal durchgeführt werden) a. Durch mehrere Links oder Rechts Klicks soll eine individuelle Koordinate / Zoom Factor generiert und per Screenshot festgehalten werden. b. Unter dem File Kontext Menü auf Save klicken und eine .mb Datei abspeichern. c. Applikation schliessen oder die individuelle Koordinate / Zoom Factor anpassen d. Unter dem File Kontext Menü auf Save klicken und die abgespeicherte .mb Datei laden.
Erwartetes Ergebnis	Beim dritten Schritt muss überprüft werden, ob die Applikation erfolgreich jeweils wieder den festgehaltenen Punkt laden kann. Dazu kann man die PictureBox Visuell überprüfen und die Werte der Offset / Zoom NumericUpDowns vergleichen.

ID / Bezeichnung	*T-005 Printing*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation das aktuell angezeigte Mandelbrotset in der PictureBox auf einer vollen Seite ausdrucken kann.
Testvoraussetzung	T-001, T-002 muss erfolgreich verlaufen sein Ein Drucker (Es kann auch Print to PDF verwendet werden)
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Drucken (Dieser Schritt soll mind. 3-mal durchgeführt werden) a. Durch mehrere Links oder Rechts Klicks soll eine individuelle Koordinate / Zoom Factor generiert werden. b. Unter dem File Kontext Menü auf Print klicken und einen Ausdruck machen
Erwartetes Ergebnis	Beim Schritt 3 soll jeweils das in der PictureBox angezeigte Mandelbrot auf einer Vollen A4 Seite ausgedruckt werden.

11 Realisierung

11.1 System realisieren

11.1.1 Prototyping mit ILGPU

11.1.1.1 Ziel

Hier war der Fokus eine Applikation zu erstellen, welche mit Hilfe von ILGPU das Mandelbrotset berechnet.

11.1.1.2 Vorgehen

Ich habe mir eine Consolen-Applikation erstellt und habe mir im Nuget-Packetmanager die ILGPU Library auf der aktuellen Version heruntergeladen. Zuerst habe ich selbst probiert ein Kernel für die Grafikkarte zu schreiben, konnte aber mit Hilfe der Dokumentation keines erstellen. Beim genauen durchlesen der Dokumentation wurde auf ein GitHub Repo verwiesen, welche Beispiele enthielt. Der Zufall wollte es, das es schon ein geschriebenes Kernel gab, welches das Mandelbrot ohne Zoom oder Offset berechnet. So konnte ich einen Prototyp erstellen, welcher in der Console das Mandelbrot ausgibt.

```
1 Verweis
static void MandelbrotKernel(
    Index1 index,
    int width, int height, int max_iterations,
    ArrayView<int> output)
{
    float h_a = -2.0f;
    float h_b = 1.0f;
    float v_a = -1.0f;
    float v_b = 1.0f;

    if (index >= output.Length)
        return;

    int img_x = index % width;
    int img_y = index / width;

    float x0 = h_a + img_x * (h_b - h_a) / width;
    float y0 = v_a + img_y * (v_b - v_a) / height;
    float x = 0.0f;
    float y = 0.0f;
    int iteration = 0;
    while ((x * x + y * y < 2 * 2) && (iteration < max_iterations))
    {
        float xtemp = x * x - y * y + x0;
        y = 2 * x * y + y0;
        x = xtemp;
        iteration += 1;
    }
    output[index] = iteration;
}
```

```
namespace consoleapp2
{
    0 Verweise
    class Program
    {
        0 Verweise
        static void Main()
        {

            Mandelbrot.CompileKernel(false);
            int width = 75;
            int height = 75;
            int[] result = new int[width* height];
            Mandelbrot.CalcCPU(result,width,height,0f,0f,0f,100);
            Console.ReadLine();
            for(int x = 0; x <width; x++)
            {
                string line = "";
                for (int y = 0; y < height; y++)
                    line += result[y+x*width] < 100 ? "--" : "XX";
                Console.WriteLine(line);
            }
            Console.ReadLine();
        }
    }
}
```

11.1.2 Erstellung der Projects

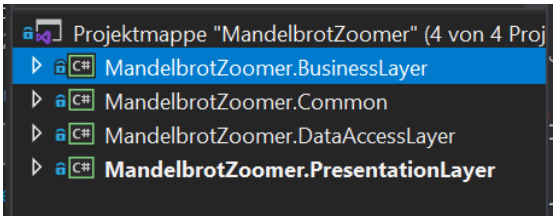
11.1.2.1 Ziel

Eine Visual Studio Solution, welche die Schichtentrennung nach PBD schon forciert, indem die Projekte sich gegenseitig so referenzieren, dass z.B. eine Klasse aus dem DataAccessLayer nicht vom PresentationLayer aus sichtbar ist und umgekehrt.

11.1.2.2 Vorgehen

Als erstes habe ich ein WinForms .NET Framework Projekt erstellt und habe es MandelbrotZoomer.PresentationLayer genannt. Sowie drei .NET Framework Klassenbibliotheken mit den Namen MandelbrotZoomer.DataAccessLayer MandelbrotZoomer.BusinessLayer und MandelbrotZoomer.Common. Auf diesen habe ich die Referenzen gesetzt:

- MandelbrotZoomer.PresentationLayer -> MandelbrotZoomer.BusinessLayer
- MandelbrotZoomer.BusinessLayer -> MandelbrotZoomer.DataAccessLayer
- MandelbrotZoomer.PresentationLayer -> MandelbrotZoomer.Common
- MandelbrotZoomer.BusinessLayer -> MandelbrotZoomer.Common
- MandelbrotZoomer.DataAccessLayer -> MandelbrotZoomer.Common



11.1.3 Einbauen der graphischen Ansicht eines Mandelbrots

11.1.3.1 Ziel

Der User soll eine Anzeige des Mandelbrotsets haben.

11.1.3.2 Vorgehen

Das einfachste ein Int[] in eine Grafik umzuwandeln ist in eine Bit-Map. Das Int[] ist ein zweidimensionales Array in einem eindimensionalen. Also musste ich Logik bauen, welche durch dieses Array iteriert und dabei x und y aus dem Index berechnet.

```
2 Verweise
public static Bitmap BitmapConverter(int[] data, int width, int height, int colorshift, int colorrange, int iterations)
{
    Bitmap bitmap = new Bitmap(width, height);
    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            int requiredIterations = data[x + y * width];
        }
    }
    return bitmap;
}
```

Danach musste ich aus der Anzahl Iterationen berechnen, welche Farbe dem Pixel zugewiesen wird.

```
private static readonly double colorrange24 = 256d * 256d * 256d;
2 Verweise
public static Bitmap BitmapConverter(int[] data, int width, int height, int colorshift, int colorrange, int iterations)
{
    Bitmap bitmap = new Bitmap(width, height);
    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            double color = data[x + y * width] * colorrange24 / iterations;
            bitmap.SetPixel(x, y, Color.FromArgb(255, data[x + y * width] == iterations ?  Color.Black: Color.FromArgb((int)color)));
        }
    }
    return bitmap;
}
```

11.1.4 Schieberegler mit änderbarer Farbpalette

11.1.4.1 Ziel

Die Berechnung der Farbe mit den Parametern aus den Trackbars Colorshift und Colorrange ergänzen. Das die Anzeige aktualisiert wird mit den Änderungen.

11.1.4.2 Vorgehen

Ich musste mich Informieren, wie ein 32bit Farbcode aufgebaut ist, um den farblichen Bereich einzuschränken und diesen zu verschieben. Dies habe ich mit dieser Berechnung gelöst:

```
private static readonly double colorrange24 = 256d * 256d * 256d;
2 Verweise
public static Bitmap BitmapConverter(int[] data, int width, int height, int colorshift, int colorrange, int iterations)
{
    Bitmap bitmap = new Bitmap(width, height);
    double absoluteColorRange = colorrange24 / 64d * colorrange;
    double absoluteColorShift = colorrange24 / 64d * colorshift;
    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            double color = data[x + y * width] * absoluteColorRange / iterations + absoluteColorShift;
            bitmap.SetPixel(x, y, Color.FromArgb(255, data[x + y * width] == iterations ?  Color.Black: Color.FromArgb((int)color)));
        }
    }
    return bitmap;
}
```

Danach musste ich lediglich die Change Events der Trackbars mit der Aktualisierung Methode der Anzeige verbinden.

```
//
// tbColorRange
//
this.tbColorRange.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
this.tbColorRange.Location = new System.Drawing.Point(444, 895);
this.tbColorRange.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
this.tbColorRange.Maximum = 64;
this.tbColorRange.Name = "tbColorRange";
this.tbColorRange.Size = new System.Drawing.Size(346, 69);
this.tbColorRange.TabIndex = 14;
this.tbColorRange.Value = 64;
this.tbColorRange.ValueChanged += new System.EventHandler(this.Render_ValueChanged);
//
```

```
//
// tbColorShift
//
this.tbColorShift.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
this.tbColorShift.LargeChange = 1;
this.tbColorShift.Location = new System.Drawing.Point(799, 895);
this.tbColorShift.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
this.tbColorShift.Maximum = 64;
this.tbColorShift.Name = "tbColorShift";
this.tbColorShift.Size = new System.Drawing.Size(346, 69);
this.tbColorShift.TabIndex = 12;
this.tbColorShift.ValueChanged += new System.EventHandler(this.Render_ValueChanged);
//
```

11.1.5 Zoom per Links/Rechts Klick

11.1.5.1 Ziel

Implementation eines Zooms und eines Offsets in der Berechnung des Mandelbrots im bestehenden Kernel. Die Registration der MouseEvents.

11.1.5.2 Vorgehen

Als erstes begann ich die MausEvents zu registrieren, welche das Offset relativ zum Cursor in der Anzeige des Mandelbrots ändern und das Zoom entsprechend zum Rechts oder Links Klick anpasst. Dies war mathematisch eine schwere Aufgabe. Dazu kam, das die relative Cursor Position in der Anzeige ebenfalls berechnet werden musste.

```
var p = pbMandelbrot.PointToClient(this.PointToScreen(e.Location));
var point = new Point(p.X, p.Y);
var width = pbMandelbrot.Image.Width;
var height = pbMandelbrot.Image.Height;


decimal rangeH = 1.5m / numZ.Value;
decimal h_a = (-0.5m + numX.Value - rangeH);
decimal h_b = (-0.5m + numX.Value + rangeH);

decimal rangeV = 1m / numZ.Value;
decimal v_a = (numY.Value + rangeV);
decimal v_b = (numY.Value - rangeV);

int index = point.X + point.Y * width;
int img_x = index % width;
int img_y = index / width;

decimal x = h_a + img_x * (h_b - h_a) / width + 0.5m;
decimal y = v_a + img_y * (v_b - v_a) / height;

deactivateRenderValueChanged = true;
numX.Value = x < numX.Maximum ? (x > numX.Minimum ? x : numX.Minimum) : numX.Maximum;
numY.Value = y < numY.Maximum ? (y > numY.Minimum ? y : numY.Minimum) : numY.Maximum;
```

Nun musste ich in der Berechnung des Mandelbrotsets diese Parameter ergänzen und einfliessen lassen.

```
2 Verweise
public static int[] CalcGPU(int width, int height, double offsetx, double offsety, double zoom, int max_iterations)
{
    int[] buffer = new int[width * height];

    int num_values = buffer.Length;
    var dev_out = accelerator.Allocate<int>(num_values);

    double rangeH = 1.5d / zoom;
    double h_a = (-0.5d + offsetx - rangeH);
    double h_b = (-0.5d + offsetx + rangeH);

    double rangeV = 1d / zoom;
    double v_a = (offsety + rangeV);
    double v_b = (offsety - rangeV);

    // Launch kernel
    mandelbrot_kernel(num_values, width, height, h_a, h_b, v_a, v_b, max_iterations, dev_out.View);
    accelerator.Synchronize();
    dev_out.CopyTo(buffer, 0, 0, num_values);

    dev_out.Dispose();
    return buffer;
}
```

11.1.6 Automatischer Zoom (Mittlerer Mausklick, Schieberegler Geschw.)

11.1.6.1 Ziel

Das automatische Zoomen auf den angeklickten Punkt im Mandelbrot mit einer einstellbaren Geschwindigkeit um einen Video Effekt zu erzielen.

11.1.6.2 Vorgehen

Das Ziel konnte hier aus Performancegründen nicht erreicht werden und macht das Feature Deffered. Also habe ich mich entschieden dieses nicht zu Implementieren, da es seinen Zweck nicht erfüllen kann.

11.1.7 Laden/Speichern einer Position im Mandelbrot

11.1.7.1 Ziel

Mit den Buttons Save/Load im File Kontext Menü muss das Offset und der Zoom Factor als .mb File abgespeichert und geladen werden können. Es soll einen FileDialog für das Laden und Speichern aufrufen, indem man das File definieren kann und einen gesetzten Filter auf .mb Dateien hat. Nach einem Laden muss die Anzeige des Mandelbrots aktualisiert werden.

11.1.7.2 Vorgehen

Als erstes musste ich definieren, was für ein serialisiertes Format ich für das File verwende. Ich habe mich für ein JSON entschieden. Dazu habe ich eine POCO Klasse erstellt mit den Properties OffsetX, OffsetY und dem Zoom Factor, welches ich in einen String Serialisieren/Deserialieren kann. Hier habe ich das NuGet von Newtonsoft Newtonsoft.JSON verwendet.

```
3 Verweise
public class MandelbrotLocation
{
    2 Verweise
    public decimal OffsetX { get; set; }

    2 Verweise
    public decimal OffsetY { get; set; }

    2 Verweise
    public decimal ZoomFactor { get; set; }
}
```

Dazu habe ich den SaveFileManager geschrieben im DataAccessLayer, welcher das Speichern und Laden von Objekten generisch zur Verfügung stellt.

```
1 Verweis
public class SaveFileManager
{
    1 Verweis
    public T Load<T>(string fullFileName)
    {
        var content = File.ReadAllText(fullFileName);
        return JsonConvert.DeserializeObject<T>(content);
    }

    1 Verweis
    public void Save<T>(string fullFileName, T o)
    {
        var content = JsonConvert.SerializeObject(o);
        File.WriteAllText(fullFileName,content);
    }
```

Im GUI habe ich den Klick-Event des Save und Load Buttons implementiert und gehandled. Darin habe ich die gegebenen Komponenten OpenFileDialog und SaveFileDialog verwendet, um den Pfad des Files zu bekommen. Mit den ausgelesenen Values aus den NumericUpDowns habe ich beim Speichern das POCO abgefüllt und mit dem Pfad den SaveFileManager verwendet, um das Objekt als File zu speichern. Mit dem gegebenen Pfad beim Laden, konnte ich mit dem SaveFileManager das File laden und in das POCO deserialisieren. Mit dem geladenen POCO habe ich die Values der NumericUpDowns befüllt.

```
1 Verweis
private void tsmSave_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog()
    {
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments),
        DefaultExt = "mb",
        AddExtension = true,
        Filter = "Mandelbrot Location (*.mb)|*.mb"
    };
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
        controller.SaveMandelbrotPostition(saveFileDialog.FileName, numX.Value, numY.Value, numZ.Value);

}

1 Verweis
private void tsmLoad_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog()
    {
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments),
        DefaultExt = "mb",
        AddExtension = true,
        Filter = "Mandelbrot Location (*.mb)|*.mb"
    };
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        var location = controller.LoadMandelbrotPostition(openFileDialog.FileName);
        numX.Value = location.OffsetX;
        numY.Value = location.OffsetY;
        numZ.Value = location.ZoomFactor;
    }
}
```

Diese Änderungen mussten auch die Anzeige des Mandelbrots aktualisieren. Da das Setzen der Values der NumericUpDowns automatisch die Aktualisierungsmethode ausführt, musste ich hier nichts anpassen.

11.1.8 Button Momentaufnahme Speichern

11.1.8.1 Ziel

Mit dem Button Screenshot im File Kontext Menü muss die Anzeige des Mandelbrots abgespeichert werden als Bilddatei mit einem entsprechenden FileDialog.

11.1.8.2 Vorgehen

Da ich in der Anzeige schon eine Bitmap hinterlegt hatte, konnte ich diese nehmen und mit einer gegebenen Komponente in ein JPG umwandeln. Um den Pfad zum Speichern zu erhalten, habe ich ebenfalls den SaveFileDialog verwendet. Um das File zu speichern, habe ich eine Überladung im SaveFileManager für das Speichern von Bitmaps gemacht, welche das Bild ebenfalls umwandelt.

```
private void btScreenshot_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog()
    {
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures),
        DefaultExt = "jpg",
        AddExtension = true,
        Filter = "Jpeg (*.jpg)|*.jpg"
    };
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
        controller.SaveScreenshot(saveFileDialog.FileName, currentPicture);

}
```

```
1 Verweis
public void Save(string fullFileName, Bitmap bmp)
{
    bmp.Save(fullFileName, ImageFormat.Jpeg);
}
```

11.1.9 Button Momentaufnahme Drucken

11.1.9.1Ziel

Mit dem Button Print muss die Anzeige des Mandelbrots fullpage gedruckt werden. Es soll ein Dialog verwendet werden, der eine Auswahl des Druckers zur Verfügung stellt, welcher die Seite drucken soll.

11.1.9.2Vorgehen

Um das Drucken zu verwalten, habe ich im DataAccessLayer einen PrintManager implementiert, welcher eine Bitmap entgegen nimmt. Dieser resized das Bild automatisch, sodass es ein A4 ausfüllt und gibt den Druck-Dialog aus, indem man den Drucker wählen kann.

```
public void PrintImage(Bitmap bmp)
{
    PrintDocument pd = new PrintDocument();

    pd.PrintPage += (object sender, PrintPageEventArgs e) =>
    {
        float newWidth = bmp.Width * 100 / bmp.HorizontalResolution;
        float newHeight = bmp.Height * 100 / bmp.VerticalResolution;

        float widthFactor = newWidth / e.MarginBounds.Width;
        float heightFactor = newHeight / e.MarginBounds.Height;

        if (widthFactor > 1 | heightFactor > 1)
        {
            if (widthFactor > heightFactor)
            {
                newWidth = newWidth / widthFactor;
                newHeight = newHeight / widthFactor;
            }
            else
            {
                newWidth = newWidth / heightFactor;
                newHeight = newHeight / heightFactor;
            }
        }
        e.Graphics.DrawImage(bmp, 0, 0, (int)newWidth, (int)newHeight);
    };

    if (new PrintDialog() { Document = pd }.ShowDialog() == DialogResult.OK)
            pd.Print();
}
```

Schlussendlich musste ich den Klick Event des Buttons handeln, welcher mit dem Bild in der Anzeige den PrintManager aufruft.

```
1 Verweis
private void tsmPrint_Click(object sender, EventArgs e)
{
    controller.PrintMandelbrot(
            (double)numX.Value,
            (double)numY.Value,
            (double)numZ.Value,
            (int)numIterationen.Value,
            tbColorShift.Value,
            tbColorRange.Value
        );
}
```

11.2 Testprotokoll

11.2.1 Testfall T-001

ID / Bezeichnung	*T-001 Mandelbrot Rendering*
Beschreibung	In diesem Test muss geprüft werden, ob das Mandelbrot korrekt in der PictureBox angezeigt wird, wenn der Render Button geklickt wird oder das Form resized wird.
Testvoraussetzung	Eine Workstation mit CUDA support sowie eine Workstation ohne CUDA Support. Bilder des Mandelbrotsets zum Vergleich mit dem Ergebnis der Applikation.
Testschritte	1. Starten der Applikation 2. Klick auf den Render Button 3. Vergrösserung des Forms 4. Verkleinerung des Forms auf die minimale Grösse
Erwartetes Ergebnis	Beim zweiten Schritt soll das Mandelbrotset korrekt grafisch abgebildet werden und entsprechend der Grösse des Forms angepasst werden. Beim dritten Schritt soll das Angezeigte Mandelbrot korrekt mit dem Form vergrössert werden und soll sich immer noch innerhalb des gegeben Rahmens befinden. Beim vierten Schritt soll das Form wie im dritten Schritt das Bild immer noch den gegebenen Rahmen ausfüllen. Das Form soll blockieren kleiner gemacht zu werde,n als die Controls Platz einnehmen. Die Controls dürfen sich nicht beginnen zu überlappen.

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	1
Fehlerbeschreibung	-

11.2.2 Testfall T-002

ID / Bezeichnung	*T-002 Zooming*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation korrekt zoomt/entzoomt auf einen angeklickten Punkt im Mandelbrotset in der PictureBox.
Testvoraussetzung	T-001 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Mehrmals links klicken auf einen Punkt im Mandelbrotset in der PictureBox an dem man sich gut orientieren kann. 4. Mehrmals rechts klicken auf einen Punkt Mandelbrotset in der PictureBox an dem man sich gut orientieren kann. 5. Mehrmals einen Rechtsklick auf die PictureBox bis man auf den Zoom Factor 1 gelangt und dies dann nochmals wiederholen.
Erwartetes Ergebnis	Beim dritten Schritt muss überprüft werden, ob bei jedem Klick das Bild sich auf den angeklickten Punkt zentriert und in das Bild hineinzoomt. Das Zoom Factor NumericUpDown Control soll den Zoom Factor korrekt aktualisieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren Beim vierten Schritt muss überprüft werden, ob bei jedem Klick das Bild sich auf den angeklickten Punkt zentriert und aus dem Bild hinauszoomt. Das Zoom Factor NumericUpDown Control soll den Zoom Factor korrekt aktualisieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren Der Zoom Factor soll nicht das entsprechende Limit 1 unterschreiten und soll aber immer noch auf den Punkt der rechtsgeklickt wurde zentrieren. Die Offset NumericUpDown Controls sollen das X/Y Offset der Zentrierung korrekt aktualisieren.

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	1
Fehlerbeschreibung	-

11.2.3 Testfall T-003

ID / Bezeichnung	*T-003 Colorshifting / Colorrange*
Beschreibung	Dieser Test überprüft, ob die farbliche Verschiebung korrekt funktioniert
Testvoraussetzung	T-001 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Die Colorshift Trackbar soll auf diverse Werte gesetzt werden sowie die Colorrange Trackbar.
Erwartetes Ergebnis	Beim dritten Schritt bei Betätigung der Colorshift Trackbar soll eine farbliche Verschiebung des Bildes durch das ganze Farbspektrum möglich sein sowie die Einschränkung auf eine bestimmte Farbliche Diversität.

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	1
Fehlerbeschreibung	-

11.2.4 Testfall T-004

ID / Bezeichnung	*T-004 Save / Load*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation den Zoomfaktor und das Offset Y sowie Offset X korrekt abspeichern kann und wieder laden
Testvoraussetzung	T-001, T-002 muss erfolgreich verlaufen sein
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Testen des Speicherns / Ladens (Dieser Schritt soll mind. 3-mal durchgeführt werden) a. Durch mehrere Links oder Rechts Klicks soll eine individuelle Koordinate / Zoom Factor generiert und per Screenshot festgehalten werden. b. Unter dem File Kontext Menü auf Save klicken und eine .mb Datei abspeichern. c. Applikation schliessen oder die individuelle Koordinate / Zoom Factor anpassen d. Unter dem File Kontext Menü auf Save klicken und die abgespeicherte .mb Datei laden.
Erwartetes Ergebnis	Beim dritten Schritt muss überprüft werden, ob die Applikation erfolgreich jeweils wieder den festgehaltenen Punkt laden kann. Dazu kann man die PictureBox Visuell überprüfen und die Werte der Offset / Zoom NumericUpDowns vergleichen.

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	1
Fehlerbeschreibung	-

11.2.5 Testfall T-005

ID / Bezeichnung	*T-005 Printing*
Beschreibung	Dieser Test soll überprüfen, ob die Applikation das aktuell angezeigte Mandelbrotset in der PictureBox auf einer vollen Seite ausdrucken kann.
Testvoraussetzung	T-001, T-002 muss erfolgreich verlaufen sein Ein Drucker (Es kann auch Print to PDF verwendet werden)
Testschritte	1. Starten der Applikation 2. Klicken auf den Render Button 3. Drucken (Dieser Schritt soll mind. 3-mal durchgeführt werden) a. Durch mehrere Links oder Rechts Klicks soll eine individuelle Koordinate / Zoom Factor generiert werden. b. Unter dem File Kontext Menü auf Print klicken und einen Ausdruck machen
Erwartetes Ergebnis	Beim Schritt 3 soll jeweils das in der PictureBox angezeigte Mandelbrot auf einer Vollen A4 Seite ausgedruckt werden.

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	2
Fehlerbeschreibung	Das Bild wird nicht wie erwartet Fullpage gedruckt, sondern kleiner oder es überschreitet die A4 Seite

Tester (Testperson)	Nicola Temporal
Datum Testdurchführung	21.01.2021
Fehlerklasse (Testergebnis)	1
Fehlerbeschreibung	-

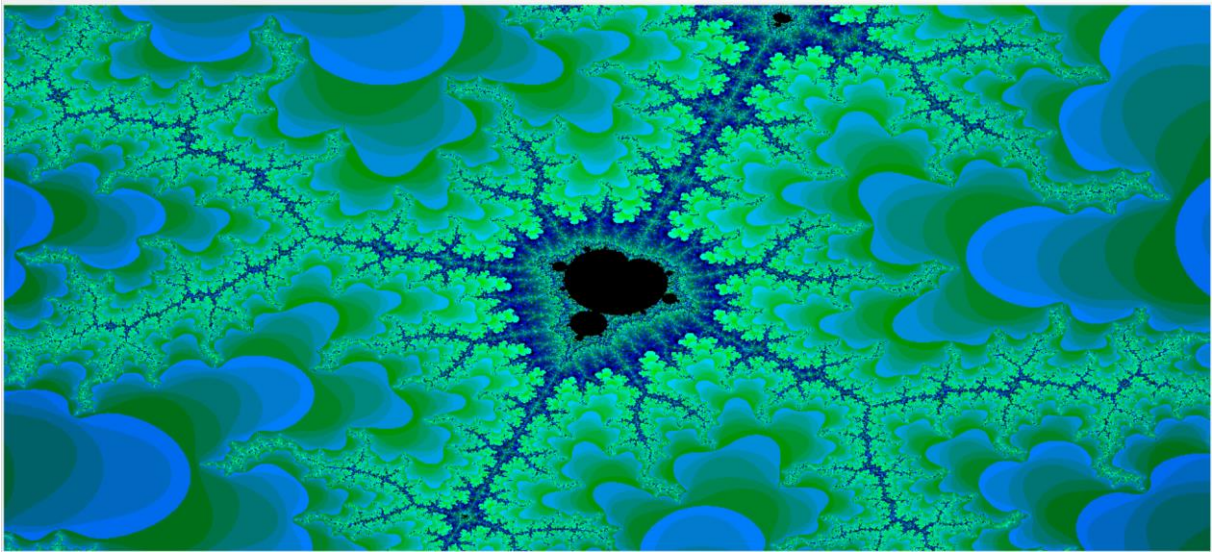
12 Reflexion

Das Coding ist erfolgreich erfolgt, jedoch gab es Features die nicht wie angedacht möglich waren. Es kann nicht immer in der Planungsphase vorausgesehen werden, wie performant eine Applikation umgesetzt werden kann. Diese Einschätzung kann zu höheren Ansprüchen führen als möglich sind. Das Drucken umzusetzen ist anspruchsvoller gewesen als erwartet. Ich musste dort speziell mehr Recherche betreiben, um eine Lösung für mein Problem zu finden.

Die gesamte Dokumentation war schwer einzuschätzen was unter einzelnen Punkten gewollt oder gefordert ist, ohne eine richtige IPA gesehen zu haben oder einem geführten Unterricht.

13 Schlusswort

Ich habe mein Bestes gegeben, um die Applikation umzusetzen. Das Ergebnis ist eine Suchtmaschine, welche mich schon während dem Programmieren oft abgelenkt hat, weil das Zoomen so spannend war. Ich habe schon immer grosses Interesse an Mathematik gehabt und das Mandelbrot ist eine super Verbindung zwischen Mathematik und programmatischer Berechnung und kann Bilder liefern, die noch niemand zuvor gesehen hat. Eine Form von Kunst, die nicht neu erfunden werden kann, aber stetig neu entdeckt.



14 Literatur- und Quellverzeichnis

Farbcode 32bit, https://de.wikipedia.org/wiki/Farbtiefe_(Computergrafik)

Newtonsoft JSON, https://www.nuget.org/packages/Newtonsoft.Json/

Stackoverflow Printing, https://stackoverflow.com/questions/4206784/c-sharp-how-to-print-aspect-ratio-full-page

Microsoft Documentation, https://docs.microsoft.com/de-de/

ILGPU Documentation, http://www.ilgpu.net/Documentation

ILGPU Samples, https://github.com/m4rs-mt/ILGPU.Samples/tree/master/Src

AleaGPU Documentation, http://www.aleagpu.com/release/3_0_4/doc/ (Diese Seite gibt es nicht mehr)

OpenGL Documentation, https://docs.gl/