



Projektbericht

Teilprojekt Bühnenbildautomation Universitäts-Rechenzentrum Freiburg

Autorin: Hannah Ullrich
Ausbildung: Applikationsentwickler, kaufmännische Systeme
Ausbildungsbetrieb: Universitäts-Rechenzentrum Freiburg
Hermann-Herderstr. 10
79104 Freiburg

Inhaltsverzeichnis

1. PROJEKTBESCHREIBUNG	4
1.1. Einleitung	4
1.2. Projektumfeld	4
1.3. Gesamtprojekt	5
1.4. Teilprojekt	6
1.5. Ist-Zustand	7
1.6. Soll-Zustand	7
2. RESSOURCEN- UND ABLAUFPLANUNG	8
2.1. Ressourcen	8
2.1.1. Computer	8
2.1.2. Vistabox	8
2.2. Ansprechpartner	10
2.3. Ablaufplanung	10
2.4. Kostenplanung	10
3. PROJEKTABLAUF	11
3.1. Analyse	11
3.1.1. Durchführung	11
3.1.2. Reflexion	11
3.2. Design	12
3.2.1. Durchführung	12
3.2.2. Reflexion	13
3.3. Realisierung	14
3.3.1. Klassenerstellung	14
3.3.2. Zusammenbau der Komponenten	16
3.3.3. Reflexion	18
3.4. Testphase	19
3.4.1. Durchführung	19
3.4.2. Reflexion	19
3.5. Dokumentation	19
3.5.1. Durchführung	19
3.5.2. Reflexion	19

4. PROJEKT-ERGEBNIS	20
4.1. Persönliches Fazit	20
4.2. Soll/Ist-Vergleich	21
4.3. Kosten/Nutzen	21
5. ANMERKUNG	22
6. ANLAGEN	22
7. ABBILDUNGSVERZEICHNIS	23
8. INDEX	23

1. Projektbeschreibung

1.1. Einleitung

Bei dem hier dokumentierten Projekt handelt es sich um eine Fallstudie im Rahmen der Umschulung bei Siemens Business Services in Freiburg.

Zusätzlich zur Entwicklung eines vom Kunden gewünschten Programms, beinhaltete die Aufgabe das Erstellen eines Projektberichtes und einer Programm-Dokumentation.

Für das gesamte Projekt standen 70 Stunden zur Verfügung.

1.2. Projektumfeld

Das Universitäts-Rechenzentrum versorgt die Universität Freiburg mit Informationstechnologie, dabei werden neue Technologien in den Bereichen

- Plattformbezogene Software (AIX, HP-UX, IRIX, Linux, Solaris, Windows 95/98/NT)
- System-Software
- Java
- Multimedia
- Network Computing

entwickelt.

Außerdem ist das Rechenzentrum für die Ausführung verschiedener Dienste zuständig:

- E-Mail
- Remote Access
- Webserver
- WLAN (Wireless LAN)
- Zentrale Server, Rechenanlagen
- Backup / Archiv

- Drucker / Plotter / Spezialgeräte
- Multimedia
- Software / Anwendungen
- Betriebssystem-Plattformen
- Diagnose u. Reparatur
- Leihgeräte
- Fachberatung (System u. Anwendung)

Das Rechenzentrum ist in verschiedene Abteilungen unterteilt, eine davon ist die Abteilung Technologie und Entwicklung, in der ich mein Praktikum machte. Hier werden unter anderem 'state-of-the-art' Projekte im Bereich Network-Computing durchgeführt.

Das alles bedeutet Arbeiten im Bereich aktueller und zukunftsweisender Informationstechnologie bei einer Einrichtung von Konzerngrößenordnung.

1.3. Gesamtprojekt

Regelmäßig zum Wintersemester bietet das Rechenzentrum eine Arbeitsgemeinschaft "Neue Medien" für die Grafik- und Designhochschule Freiburg an.

Dieses Jahr findet zusätzlich dazu eine Zusammenarbeit mit dem Theater Berlin in Form von Videokonferenzen statt.

Die Idee ist, das Bühnengeschehen über eine oder mehr Kameras bzw. Web-Cams aufzufangen, die im Theater an verschiedenen Stellen positioniert sind. Die Kameras sind mit einem Personal Computer verbunden, mit dem der aufgenommene Livestream auf verschiedenste Art und Weise transformiert und dann möglichst in Echtzeit als Bühnenbild an die Wand projiziert werden kann. Als Projektionsmedium sind Beamer vorgesehen, da diese auch einfach an den PC angeschlossen werden können.

Den Akteuren auf der Bühne soll es möglichst gemacht werden, aktiv das Bühnenbild zu steuern oder zu verändern.

Dies wäre z.B. mit Hilfe von versteckten Fußschalten auf der Bühne möglich, die über eine Schnittstelle mit dem Rechner verbunden sind, über die der "Film" z.B. angehalten, oder ein Bild-Filter gewechselt werden kann.

Nach Beendigung des gesamten Projektes soll es dem Anwender möglich sein, entweder über ein '*Grafical User Interface*', im Folgenden '*GUI*' genannt, oder über Schnittstellen die Web-Cams anzusprechen und einen Livestream auf einem beliebigen Medium darzustellen.

Zudem soll es möglich sein, die Bilder über verschiedene Transformationen (z.B.: Fourier Transformation) in Echtzeit verändert darzustellen und/oder verschiedene Filter auf das Video zu legen, um diese als Bühnenbild zu verwenden. Notwendige Hilfsmittel: '*Vistabox*' (siehe 2.1.2) mit integriertem Webserver, zwei oder mehr Web-Cams, Netzwerkanbindung.

Es ist eine Kunstform, die zwischen Bühnenbild und Videokunst rangiert, die mit Technik zu tun hat, und doch leicht wirkt, dass man den technischen Aufwand einfach vergessen kann.

1.4. Teilprojekt

Ein permanenter Datenstrom in Form von Bildern soll über eine Java-Applikation erhalten werden. Dazu wird über eine fest vergebene IP-Adresse eine statische Verbindung zur '*Vistabox*' aufgebaut. Über diese Verbindung werden die einzelnen Bilder (JPEG-Format) geholt und dann in einem eigenen Fenster angezeigt.

Unterschiedliche Filter sollen in diese Applikation implementiert werden, um den Livestream in verschiedenen Varianten darstellen zu können.

Die einzelnen Web-Cams werden über eine *GUI* angesprochen, ebenso verschiedenen Filter ausgewählt.

Insgesamt soll die Anzeige des Streams den Bildschirm eines handelsüblichen Monitors ausfüllen.

1.5. Ist-Zustand

Derzeit können über eine Java-Applikation nur Einzelbilder (384x288 Pixel) dargestellt werden.

Es existiert eine Klasse, in der die Verbindung zum Webserver der *'Vistabox'* hergestellt wird, ein Bild geholt und dieses über einen Frame dargestellt wird.

1.6. Soll-Zustand

Das gesamte Programm soll in Java programmiert werden, da diese Programmiersprache plattformunabhängig ist.

Zunächst muss die bestehende Klasse so verändert werden, dass es möglich ist einen *'Livestream'* anzeigen zu können. Alle Kameras müssen angesprochen werden können.

Außerdem ist es gewünscht, die Bilder zu transformieren, d.h. auf diese verschiedene Filter zu legen.

Der darzustellende *'Film'* wird in einem eigenen Fenster angezeigt.

Für den Nutzer soll eine *GUI* mit *'Java-Swing'* erstellt werden, damit er die verschiedenen Kameras und Filter seiner Wahl aussuchen kann. Das Programm wird auch über die *GUI* beendet.

2. Ressourcen- und Ablaufplanung

2.1. Ressourcen

Zur Realisierung des Projektes standen folgende notwendigen Komponenten zur Verfügung:

2.1.1. Computer

Hardware:

- Pentium III, 800 MHz
- 384 SDRAM
- 100 Mbit Netzwerkanbindung

Software:

- Betriebssystem: Microsoft Windows XP
- Java Plattform: j2sdk1.4.2¹
- Entwicklungsumgebung: NetBeans IDE 3.5.1²
- Dokumentation: Microsoft Office XP

2.1.2. Vistabox

Das Grundgerät ist die “*Convision V610A*“ welches für die Videoüberwachung verwendet werden kann. Es hat einen integrierten Webserver, an den bis zu sechs Kameras über BNC-Stecker³ angeschlossen werden können. Die Videodatenübertragung erfolgt über TCP/IP im LAN, über ISDN oder Analogmodem. Auf der Rückseite befindet sich ein RJ45 Stecker, in den ein 10-Base-T Kabel eingesteckt werden kann. Während meiner Arbeit war die ‘*Vistabox*’ über ein Netzwirkabel mit dem LAN verbunden.

Die komplette Bedienung und Konfiguration der ‘*Vistabox*’ (z.B. IP-Adressvergabe) erfolgt durch einen Java-fähigen Webbrowser über HTML.

¹ <http://java.sun.com/>

² <http://www.netbeans.org>

³ Bajonett-Verschluss nach Bayonet Neill Concelmann

Im so genannten Live-Modus können über ein Applet bis zu 25 Bilder in der Sekunde übertragen werden.

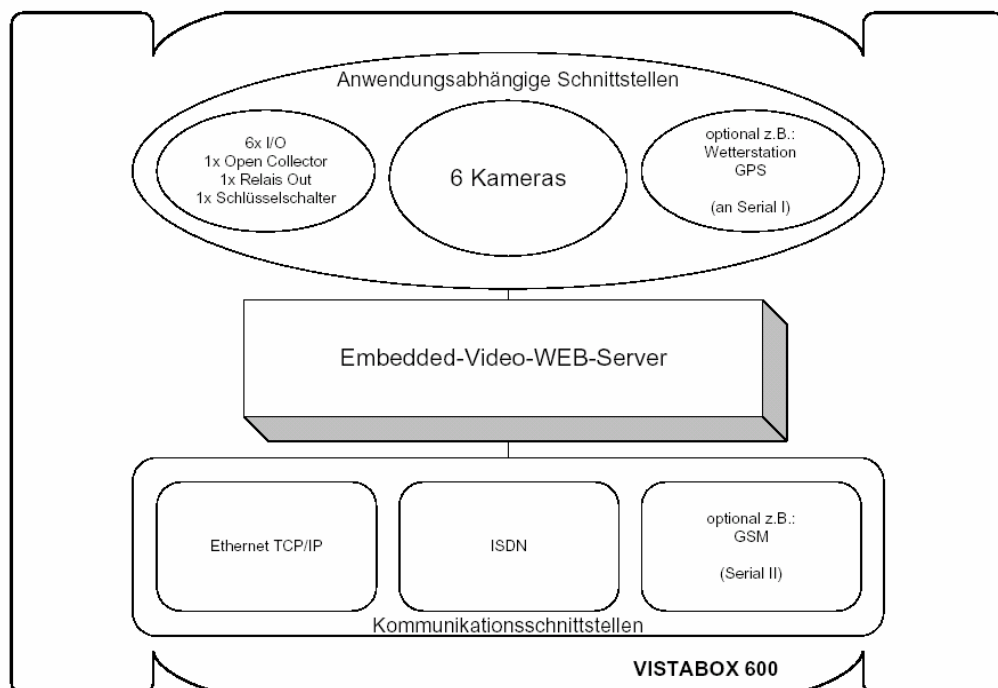
Es stehen sechs Alarmeingänge und für die direkte Steuerung von Alarmgeräten zwei Ausgänge zur Verfügung. Über eine serielle Schnittstelle können weitere Geräte, wie schwenk- und zoombare Kameras gesteuert werden.

Es gibt zwei digitale Ausgänge:

1. Relaisausgang: Dieser Ausgang liefert 12V / 500mA.
2. Open-Kollektor: Dieser Ausgang kann mit bis zu 12V / 100mA belastet werden.

Es können also Geräte, die diesen Anforderungen entsprechen, angeschlossen werden, z.B.: Signallampe, Sirene, Steuerungen.

Abbildung 1 Schnittstellen 'Vistabox'⁴



⁴ <http://www.convision.de>

2.2. Ansprechpartner

Für den Projektablauf und die technischer Umsetzung war der Leiter der Abteilung "Technologie und Entwicklung", Herr Dr. Winterer zuständig.

Bei Fragen zur Programmierung war auch er der Ansprechpartner.

2.3. Ablaufplanung

Die Ablaufplanung wurde in Zusammenarbeit mit dem Ausbilder erstellt.

Analyse	• Ist-Zustand	3 Std.
	• MotionJPEG	3 Std.
Design	• Entwurf einer GUI	6 Std.
Realisierung	• Klassenaufbau LiveStream	21 Std.
	• Zusammenbau der Komponenten	9 Std.
Testphase	• Fehlersuche und -behebung	8 Std.
Dokumentation	• Projektbericht	10 Std.
	• Programmdokumentation	10 Std.
		70 Std.

2.4. Kostenplanung

Die Kosten des Teil-Projekts bestehen aus den Kosten für den Praktikanten, also 70 Stunden à 6 €, entspricht 420.- €.

Da eine Berechnung der Kosten des Gesamt-Projekts der Zeit nicht möglich ist, wurde auch auf eine Berechnung des Anteils vom Teilprojekt verzichtet.

3. Projekttablauf

3.1. Analyse

3.1.1. Durchführung

In dieser Phase werden die notwendigen technischen Hilfsmittel übergeben und erklärt.

Das bedeutet sich erst mal mit den technischen Geräten vertraut zu machen: was ist die *'Vistabox'* und was kann sie.

Auch ist es notwendig, sich in die vorhandene Software einzulesen und zu verstehen.

Es zeigt sich, dass eine Verbindung zu dem Server aufgebaut wird, ein JPEG-Bild geholt und dieses in einem Frame dargestellt wird. Im Anschluss daran wird die Verbindung wieder getrennt.

Die *'Vistabox'* kann über ein Applet angesprochen werden und liefert dann einen M-JPEG Datenstrom. M-JPEG ist die Abkürzung für *'Motion-Joint Picture Expert Group'*. Dies ist ein Dateiformat und hardwareabhängiges Kompressionsverfahren, in dem jedes Einzelbild einer Videosequenz vollständig - und zwar für sich selbst JPEG-komprimiert - abgespeichert wird.

Dies bedeutet, dass der Datenstrom aus aneinandergereihten Bildern im JPEG-Format besteht.

In groben Zügen wird skizziert, wie die Benutzeroberfläche angelegt werden könnte.

3.1.2. Reflexion

Die Analyse-Phase kann schneller abgeschlossen werden als erwartet, da die Anforderungen klar vorgegeben und verständlich sind.

3.2. Design

3.2.1. Durchführung

Die Designphase bezieht sich in erster Linie auf die Erstellung der *'Graphical User Interface'*, die *'GUI'* innerhalb der Klasse *'RegieGui'*.

Die Vorgabe ist, unter Verwendung von *'Java-Swing'* die *'GUI'* zu erstellen.

Dies bedeutet eine Auseinandersetzung mit der Architektur von *'Swing'*.

Die Swing Bibliothek umfasst mehr als 500 Klassen.

Die einzelnen Swing-Komponenten bilden eine Hierarchie, die von der Basisklasse *'JComponent'* ausgeht.

In den ersten Stunden ist die Frage, welche Komponenten verwendet werden und wie sie anzuordnen sind.

'JFrame' erwies sich als geeignet, weil dies eine Klasse ist, deren Darstellung des Fensterrahmens vom Betriebssystem bestimmt wird und der innere Bereich dagegen komplett in Java gezeichnet wird.

Dieser *'JFrame'* wird mit Hilfe des *'GridLayouts'* in fünf Bereiche unterteilt, als eine grafische Matrix mit einheitlichen Zellen. Dieses Layout kann jederzeit geändert werden und scheint flexibel genug für später gewünschte Erweiterungen. In dieses Grundgerüst können dann weitere Komponenten von *'Swing'* eingefügt werden.

Für die Imagefilter-Auswahl werden Checkboxes der Klasse *'JCheckBox'* implementiert. Um einzelne Kameras auswählen zu können werden Buttons von der Klasse *'JButton'* verwendet.

Zum Beenden der Anwendung wird ein weiterer Button eingefügt.

Im fortgeschrittenen Programmierstadium zeigt sich, dass die Verwendung von Checkboxes weniger sinnvoll ist, da das Gesamtprojekt weitaus mehr Filtermöglichkeiten vorsieht. Eine Anordnung von zwanzig oder gar mehr Checkboxes ist dann unübersichtlich für den Nutzer.

Die Entscheidung fällt auf die Implementierung von Listboxen aus der Klasse *'JList'*. Eine für die Filter- und eine für die Bildformat-Auswahl.

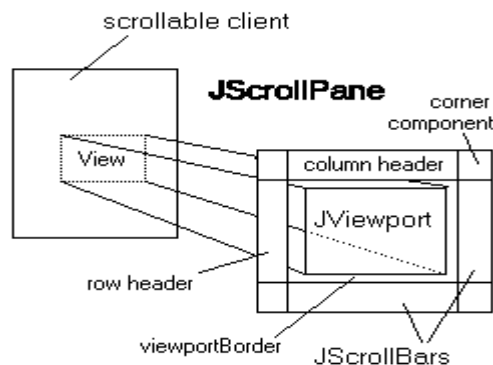
Dabei treten unerwartete Probleme auf, die viel Zeit kosten.

In der Listbox soll eine Bildlaufleiste sein, damit im Feld navigiert werden kann.

Es ist erst nicht klar, dass es dafür auch wieder eine eigene Klasse gibt, nämlich die Klasse *'JScrollPane'*.

Dadurch wird die *'Scrollbarkeit'* von den so genannten *'lightweight componenten'*⁵ in Swing unterstützt. Im *'JScrollPane'* wird ein Darstellungsfeld *'JViewport'* zugeordnet, in diesem Fall das Listfeld.

Abbildung 2 Java-Klasse *JScrollPane*⁶



3.2.2. Reflexion

Es stellt sich heraus, dass die Design-Phase während des ganzen Projektes eigentlich nicht abgeschlossen wird. Die Grenze zwischen Design und Programmierung ist nicht klar definiert.

Die Bibliothek von *Swing*⁷ erweist sich als sehr komplex und es benötigt mehr Einarbeitungszeit sich darin zurechtzufinden.

Je weiter die Programmierung voranschreitet, desto mehr Notwendigkeiten ergeben sich, die *'GUI'* zu ändern, nicht zuletzt um die Erweiterbarkeit jederzeit zu gewährleisten.

⁵ Damit wird eine größere Vielfalt und plattformunabhängiges Erscheinungsbild erreicht

⁶ java.sun.com/j2se/1.4/docs/api/javawx/swing/JScrollPane.html

⁷ <http://java.sun.com/j2se/1.4.2/docs/api/>

3.3. Realisierung

Die Realisierungsphase umfasst die Bereiche der Klassenerstellung und dem Zusammenbau der Komponenten.

Diese beiden Bereiche sind nicht klar trennbar, da zum Teil beim Zusammenbau der Komponenten gravierende Änderungen notwendig sind, um das Programm lauffähig zu bekommen. Um erste Tests machen zu können, mussten frühzeitig die einzelnen Klassen miteinander verbunden werden.

3.3.1. Klassenerstellung

Zusätzlich zu der Klasse *'RegieGui'*, in der die *'GUI'* dargestellt wird, brauchte es eine weitere um die Verbindung zu der *'Vistabox'* herzustellen und den Livestream zu erhalten. Diese Klasse nannte ich zuletzt *'MyPush'*.

Erste Versuche in einer Schleife ein JPEG nach dem anderen zu holen, scheiterten, da immer wieder eine neue Verbindung aufgebaut werden musste und der Frame, in dem das Bild gezeichnet wird, jedes Mal erneut dargestellt wird. Es ergab ein unruhiges Bild und somit ein absolut unbefriedigendes Ergebnis.

Weiter ist es erforderlich sich mit anderen zusätzlichen Java-Klassen auseinander zusetzen, die notwendig sind um Images zeichnen zu können.

- *'BufferedImage'*
Sehr komplexe Bilder die mehrmals verwendet werden, können schneller geladen werden.
- *'MediaTracker'*
Vermeidung von Bildflackern; Erkennt den Status von Media-Objekten, d.h. das Programm wartet solange, bis das Bild vollständig geladen ist.

Zunächst ist noch weiterhin unklar, warum mittels eines Applets ein Stream empfangen wird, jedoch nicht mittels einer Applikation. Eine intensivere

Auseinandersetzung mit den Möglichkeiten der *'Vistabox'* ist nötig, um dies zu klären.

Es stellt sich heraus, dass bei einer Stream - Anforderung, sowohl im Applet als auch bei der reinen Darstellung im Browser niemals der Bildaufruf mit *.jpg* endet. Als Endung wird immer *.push* verwendet.

Der Versuch über den Aufruf "IP-Adresse/fullsize.push" einen Stream zu erhalten scheitert.

Mit etwas Hilfe vom Praktikumbetreuer gelingt es nach vielen Experimenten eine Schleifenstruktur zu erstellen, die es ermöglicht einen Stream anzuzeigen. Diese erste Lösung muss noch verfeinert werden, da entweder der Stream läuft, aber der Frame mit dem Bild nicht gezeichnet wird, oder es kommt nur ein einzelnes Bild.

Nachdem der Datenstrom kontinuierlich angezeigt werden kann, setzen wir uns mit den Möglichkeiten der verschiedenen Bildfilter auseinander.

Bei Recherchen im Internet sind viele verschiedene und auch sehr reizvolle Bildfilter zu finden. Hierzu gibt es keine speziellen Vorgaben für die Filter. Die geeignetsten können ausgesucht werden. Die einzige Anforderung war, mit dem *'CropImageFilter'* ein sogenanntes *'cropped Image'* zu erstellen. Mit diesem Filter schneidet man aus einem Bild einen quadratischen Teil aus, der als Grundlage für die *'Fast Fourier Transformation'* notwendig ist.

Nicht zufriedenstellend ist dabei, dass ein Teil des Bildes nicht mehr sichtbar ist. Daher ist es notwendig noch andere Filter auszutesten, um ein Bild auf quadratische Maße zu bekommen.

Das Problem ist, dass die *'Vistabox'* nur Bilder mit 388 x 284 Pixel Format liefern kann. Dieses Bild muss vergrößert werden, damit es auf der ganzen Breite und Höhe des Monitors angezeigt werden kann. Das geht nur auf Kosten der Bildqualität. Mit Hilfe von Algorithmen werden entweder Zeilen und Spalten verdoppelt, oder von den Pixel Bereichsmittelwerte berechnet und somit zusätzliche Pixel konstruiert. Die Bilder sind damit unschärfer und die Maße stimmen nicht mehr, z.B. quadratische Formen werden rechteckig dargestellt, Köpfe erscheinen lang gezogen.

Die Zeit drängt und nach Absprache mit dem Projektleiter wird das Problem zurückgestellt. Die Priorität liegt eher bei dem Einbau weiterer Filter und diese funktionsfähig zu bekommen. Dies wird aufgrund des aktuellen Wissensstandes problemlos ausgeführt.

Durch den Fortschritt des Projektes entwickelt sich die Idee, ein Bild „einfrieren“ zu können. Der *'Thread'* soll auf Knopfdruck pausieren, dann ein Standbild anzeigen und jederzeit wieder gestartet werden können.

Hierzu müssen zusätzlich zwei Schaltflächen eingebaut werden, über die eine Variable auf wahr oder falsch gesetzt wird und diese dem *'Thread'* übergeben. Der Zustand der Variablen wird in einer while-Schleife abgefragt und der *'Thread'* in einen 'schlafenden' Zustand versetzt.

3.3.2. Zusammenbau der Komponenten

Bei der ersten Verbindung der *'GUI'* mit der *MyPush*-Klasse, wird der Stream angezeigt. Es ist aber nicht möglich diesen anzuhalten. Die *'GUI'* kann nicht mehr in den Vordergrund gebracht werden, um von da aus das Programm zu beenden. Die Benutzeroberfläche bekommt keine "Laufzeit" mehr, das Programm belegt die gesamten Ressourcen der Applikation um den Stream zu zeichnen.

Die Lösung ist *MyPush* in einen *'Thread'* um zu schreiben. Da die Klasse schon von *'JFrame'* abgeleitet wird, und nur eine Ableitung möglich ist, wird das Interface *'Runnable'* implementiert. Dieses Interface verlangt eine Methode *'run()'*, die ohne Argumente aufgerufen wird.

Die Ableitung der Klasse *'JFrame'* ist notwendig, um im Konstruktor direkt den Frame für das Bild aufbauen zu können. Der Livestream benötigt ein *'Graphics Component'* um gezeichnet werden zu können.

Die Methoden von *'Threads'* werden klar definiert, wie sie aufgerufen, gestartet und gestoppt werden können.

Wird erst ein Bild in fullscreen-Modus angezeigt, dieses geschlossen und dann ein Bild in 600x800 Pixel Größe, springt das Bild zwischen den beiden Modi hin und her. Es zeigt sich, dass beim erneuten Aufrufen eines Bildes die alten Parameter weiterhin mitverarbeitet werden und der Stream immer wieder umgesprungen ist.

Deshalb wird in die Schleife, in der der Bildstrom geholt wird, eine Variable eingebaut, die auf wahr gesetzt wird, wenn der Bildframe geschlossen wird. Die *'run()'*-Methode und somit der *'Thread'* wird mit Hilfe dieser Variablen beendet. Ein neuer *'Thread'* mit anderen Filtern kann gestartet werden.

Auch hier entwickelt sich durch den Fortschritt des Projekts die Vorstellung, dass die Filter während des laufenden Stream geändert werden sollen. Zuerst herrschte die Meinung vor, dass dies nicht geht, da schon bei den ersten Versuchen Schwierigkeiten auftreten, es kommt aber doch ziemlich schnell zur Lösung. Realisiert wird es durch das Einbinden eines *'Interfaces'* aus der Swing-Bibliothek dem *'ListSelectionListener'*.

Über die implementierte Methode dieses *'Interfaces'* werden die Listfelder ständig überwacht, ob ein anderer Eintrag ausgewählt wird.

Abbildung 3 GUI vor den Veränderungen

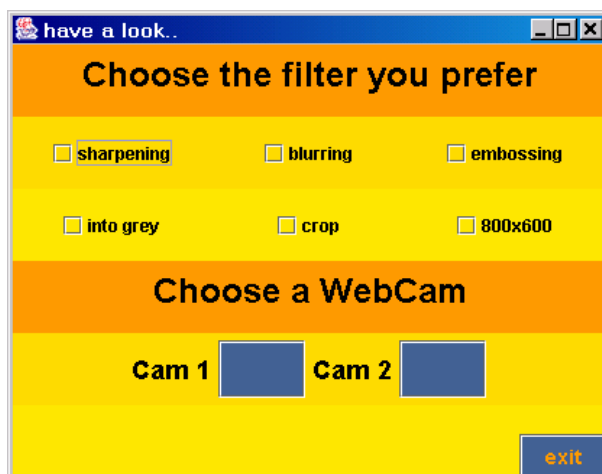
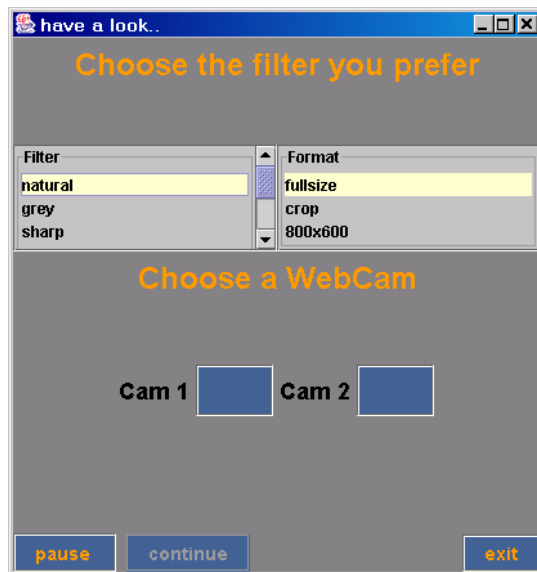


Abbildung 4 GUI bei Projektende



3.3.3. Reflexion

Trotz der klaren Aufgabenstellung fällt es am Anfang der Programmierphase schwer, sich einzuarbeiten.

Im Mittelpunkt steht zu lange die vorgegebene Klasse, die letztendlich nicht brauchbar ist. Nachdem eine Loslösung davon stattfand, eine eigene Struktur aufgebaut wird, sind die Abläufe klarer und die Umsetzung möglich. Es kann gezielter an Probleme heran getreten werden und die Lösungsfindung fällt leichter.

Die Zeitanforderung wird in dieser Phase deutlich überschritten, auch aus dem Grund, da die einzelnen Bereiche nicht mehr klar trennbar waren. Design und Test sind immer wieder Bestandteil der Programmierphase.

3.4. Testphase

3.4.1. Durchführung

Die eigentliche Testphase fällt verhältnismäßig kurz aus, da nach jedem neuen programmierten Abschnitt notwendige Tests gemacht werden.

Abschlusstests finden statt, in dem die entwickelte Software unter ähnlichen Bedingungen wie im Theater vorgeführt und anschließend beurteilt wird.

Die Projektleitung ist sehr zufrieden mit dem Ergebnis, welches den ursprünglichen und zusätzlichen Anforderungen absolut entspricht.

3.4.2. Reflexion

Es hat sich als richtig erwiesen regelmäßig Tests während der ganzen Realisierungsphase hindurch zu machen. Letztendlich wäre es problematisch geworden, erst am Ende alle notwendigen Tests durchzuführen, da es sonst nicht machbar gewesen wäre, das Projekt fristgerecht fertig zu stellen. Deshalb wurde die geplante Vorgabe der Testphase umverteilt und "en Block" lediglich zwei Stunden verbraucht.

3.5. Dokumentation

3.5.1. Durchführung

Das Führen eines Projekttagebuchs erwies sich als sehr hilfreich. Hiermit konnte ein Großteil der Dokumentation aufgebaut werden.

3.5.2. Reflexion

Die Dokumentations-Phase ist kürzer als geplant. Zum einen weil es schwer fällt, sich von der Programmierung zu lösen, zum anderen, weil der Anspruch vorherrschte, das Projekt den Anforderungen gemäß präsentieren zu können.

4. Projekt-Ergebnis

4.1. Persönliches Fazit

Je weiter ich in dem Projekt voranschritt, desto leichter fiel es mir, mich in der Umgebung zurechtzufinden. Vor allem auch im Bezug auf die Möglichkeiten von Java.

Allerdings hätte ich an manchen Stellen, vor allem in der Anfangsphase, etwas früher gegensteuern sollen, um die vorgegebene Zeit besser einhalten zu können. Der eigene Anspruch, selbst die Lösung für manches Problem zu finden, war zum Teil zu hoch. Durch die fehlende Erfahrung geeignete Informationen z.B. im 'www'⁸ zu finden, resultierten einige Probleme. Im Laufe der Zeit wurde ich immer häufiger fündig und habe sehr viel online recherchiert. Die 'API' (Applikation Programmers Interface)⁹ von Sun war mir während der Arbeit eine sehr große Hilfe. Bei der Fülle von Informationen war es zum Teil sehr schwierig, geeignete Klassen ausfindig zu machen und sie sinnvoll umzusetzen, bzw. sich auf die wesentlichen Dinge zu konzentrieren.

Schade war, dass ich kaum die Möglichkeit hatte, an einem entsprechend ausgerüsteten PC die Software laufen lassen zu können. Mit dem Rechner, der mir zur Verfügung stand, war es nicht möglich den Stream ruckelfrei und in Echtzeit anzuzeigen. Es gab immer Verzögerungen von ca. 1 min.

Am Ende fiel es sehr schwer sich vom Programmieren loszulösen, um mit der Dokumentation zu beginnen. Neue Ideen kamen auf und allmählich konnte ich immer mehr abschätzen, welche weiteren Möglichkeiten es gibt, um das Projekt zu verbessern und weiter auszubauen.

Die Erfahrungen und auch Ideen, die ich gesammelt habe sind ein guter Grundstein für die Weiterführung des Projekts mit dem ich betraut bin.

⁸ worldwideweb

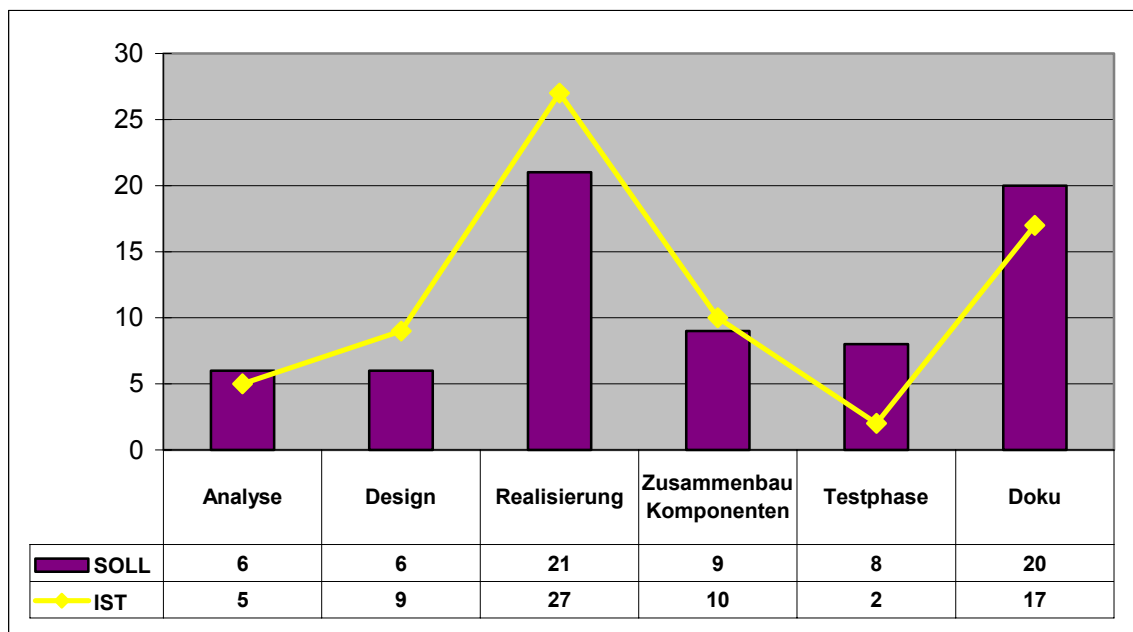
⁹ <http://java.sun.com/j2se/1.4.2/docs/api/>

4.2. Soll/Ist-Vergleich

Die folgende Grafik zeigt die Abweichungen der Zeitanforderungen innerhalb der einzelnen Phasen.

Begründungen hierzu sind in den Reflexionen der Bereiche zu finden.

Abbildung 5 Soll/Ist Vergleich



4.3. Kosten/Nutzen

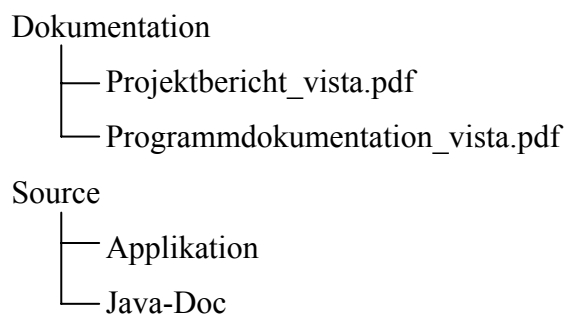
Eine Kosten-Nutzen-Rechnung kann aus zwei Gründen nicht erfolgen. Zum einen lässt sich der Nutzen erst nach Abschluss des Gesamt-Projekts feststellen. Zum anderen geht es bei dem ganzen Projekt nicht um Zeitersparnis, sondern Ausnutzung technologischer Mittel und die Möglichkeit der Interaktion des Menschen damit. Es steht jedoch außer Frage, dass dieses Projekt, wenn es abgeschlossen ist, frei zugänglich für andere interessierte Gruppen ist und somit nicht mehr soviel Zeit investiert werden muss, um selbst ein Projekt in dieser Form aufzuziehen.

5. Anmerkung

Projektbericht sowie die Programmdokumentation sind auf der beiliegenden CD Ordner 'Dokumentation' zu finden.

Das ausführbare Programm und die entsprechende 'Javadoc' sind im Ordner 'Source' enthalten.

Dateiverzeichnis der CD:



6. Anlagen

Am Ende des Projektberichtes befinden sich die Eidesstattliche Versicherung zur Durchführung und Dokumentation der betrieblichen Projektarbeit der Abschlussprüfung in den IT-Berufen und der genehmigte Antrag auf betriebliche Projektarbeit.

7. Abbildungsverzeichnis

ABBILDUNG 1	SCHNITTSTELLEN 'VISTABOX'	9
ABBILDUNG 2	JAVA-KLASSE JSCROLLPANE	13
ABBILDUNG 3	GUI VOR DEN VERÄNDERUNGEN	17
ABBILDUNG 4	GUI BEI PROJEKTENDE	18
ABBILDUNG 5	SOLL/IST VERGLEICH	21

8. Index

A		
<i>API</i>	20	
B		
<i>BufferedImage</i>	14	
C		
<i>Convion V610A</i>	8	
<i>CropImageFilter</i>	15	
F		
<i>Fast Fourier Transformation</i>	15	
G		
<i>Grafical User Interface</i>	6	
<i>Graphics Component</i>	16	
<i>GridLayouts</i>	12	
<i>GUI</i>	6, 7, 12	
I		
<i>Interface</i>	17	
J		
<i>Java-Swing</i>	7, 12	
<i>JButton</i>	12	
		<i>JCheckBox</i>
		12
		<i>JComponent</i>
		12
		<i>JFrame</i>
		12, 16
		<i>JList</i>
		12
		<i>JScrollPane</i>
		13
		<i>JViewport</i>
		13
		L
		<i>ListSelectionListener</i>
		17
		M
		<i>MediaTracker</i>
		14
		<i>Motion-Joint Picture Expert Group</i>
		11
		<i>MyPush</i>
		14
		R
		<i>RegieGui</i>
		14
		<i>Runnable</i>
		16
		T
		<i>Thread</i>
		16