Informatikprojekt

Roland Schäfer,
Institut für theoretische Physik,
Universität Bern.

Betreut durch Prof. O. Nierstrasz,
Software Composition Group,
Institut für Informatik,
Universität Bern

19. April 2002

Inhaltsverzeichnis

1 Pflichtenheft **4**
1.1 Einleitung . . . 4
1.2 Spezifikation der Anforderungen . . . 4
1.2.1 Scans . . . 5
1.3 Endbenutzer Szenarien . . . 6
1.4 Analyse . . . 8
1.4.1 Machbarkeit . . . 8
1.4.2 Prototyp . . . 9

2 Design **10**
2.1 Klassen . . . 10
2.2 Klassendiagramm . . . 11
2.3 Sequence diagrams . . . 13
2.3.1 Sequence diagram für 'start' . . . 13
2.3.2 Sequence diagram für 'stop' . . . 14

3 Mathematik **15**
3.1 Situationsskizze . . . 15
3.2 Motivation . . . 15
3.3 Konzept . . . 16
3.3.1 Translation: . . . 17
3.3.2 Rotation: . . . 17

4 Tests **18**

5 Programmdokumentation **19**
5.1 Hauptfenster . . . 19
5.2 Scandefinition . . . 19
5.2.1 Scanfile . . . 20
5.2.2 Fahrbefehle . . . 20

6 Auszuführende Arbeiten **21**
6.1 Korrekturen/Ergänzungen . . . 21
6.2 Mögliche Erweiterungen . . . 21

7 Fazit des Informatikprojektes **23**

A Quellcode **25**

B Mathematica **87**

C Screenshots **91**

D Scanfile **94**

1 Pflichtenheft

1.1 Einleitung

Für die Kometenmission Rosetta der ESA (European Space Agency) entwickelt die Abteilung Weltraumforschung und Planetologie des Physikalischen Instituts der Universität Bern 3 Instrumente. Die beiden Massenspektrometer RTOF und DFMS sowie den Drucksensoren COPS. Zum Kalibrieren und Spezifizieren der Instrumente wird die Kalibrieranlage CASYMIR aufgebaut, in der die Kometenumgebung mit einem Gasstrahl simuliert wird. Die Instrumente werden mit Hilfe eines 5-Achsentisches im Strahl positioniert.
Die gesamte Anlage wird über eine Java-Applikation gesteuert. Darin eingebunden ist die Steuerung des 5-Achsentisches.
Das vorliegende Dokument beschreibt die Applikation TableControl, die für die Tischsteuerung verantwortlich ist.

1.2 Spezifikation der Anforderungen

Die Steuerung des 5-Achsentisches TableControl soll als Unterprogramm in die Gesamtsteuerung der Testanlage Casymir integriert werden. Die Java-Applikation wird auf PCs mit den Betriebssystemen Winxx oder Linux installiert.
Als Schnittstelle zum Benutzer wird ein graphisches Userinterface (GUI) benutzt.
TableControl muss sich nicht um die Kommunikation mit den Servoverstärkern der einzelnen Schrittmotoren kümmern. Sie übergibt die Steuerbefehle an die Gesamtsteuerung, welche die Verbindung zu den Servoverstärkern aufbaut und die Befehle schliesslich an diese weiterleitet.

Direkte Steuerung: Der Benutzer kann die Soll-Position in Zahlenwerten der einzelnen Koordinaten angeben (Absolut- oder Relativkoordinaten). Die Geschwindigkeit ist in einem bestimmten Bereich frei wählbar. Der Benutzer ist dafür verantwortlich, dass keine Positionen angefahren werden, die zu einer Kollision führen. Die Absolutkoordinaten aller Achsen werden angezeigt und laufend aktualisiert.

Erstellen von Scanabläufen: Bei einem Scan (Details siehe weiter unten) handelt es sich um das Anfahren von einer oder mehreren Positionen. In den einzelne Positionen werden dann Messungen durchgeführt.
Aufgrund der Benutzereingaben wird eine Sequenz von Steuerbefehlen erzeugt und abgespeichert.

Es muss einfach möglich sein, die Steuerbefehle der Scans zu übernehmen und mit den Steuerbefehlen der Instrumente in einem gemeinsamen Scanfile abzuspeichern (Position anfahren → Instrument ansteuern und Messen → nächste Position anfahren.....)

Zeitrahmen: Die ersten Scans werden voraussichtlich im Sommer 01 durchgeführt. Es sollte möglichst bald ein Prototyp erstellt werden, der die Steuerung des 5-Achsentisches durch Eingabe von Absolutkoordinaten und Fahrgeschwindigkeit ermöglicht.

1.2.1 Scans

Eine gewünschte Position wird angefahren und erst dann beginnt das Instrument mit der Messung. Während dem Positionieren wird nie gemessen.

Mögliche Scanarten

Ebenenscan: Auf einer Ebene, die Senkrecht zum Strahl steht, kann ein Punktgitter festgelegt werden, das in einer zu bestimmenden Reihenfolge abgefahren wird.

Kugelscan: In der Umgebung des Strahls wird ein Kugelsegment definiert, worauf ein gleichmässiges Punktgitter gelegt wird. Auch hier sollen die Punkte in einer zu bestimmenden Reihenfolge angefahren werden.

Winkelscan: In einem beliebig ausgewählten Punkt kann ein Winkelscan durchgeführt werden. Dabei wird das Instrument um bestimmte Winkelschritte in der Horizontalen und der Vertikalen um den Punkt gekippt.

1.3 Endbenutzer Szenarien

Einen guten Überblick über die prinzipiellen Endbenutzer Szenarien gibt das use case diagram (Abb. 1).

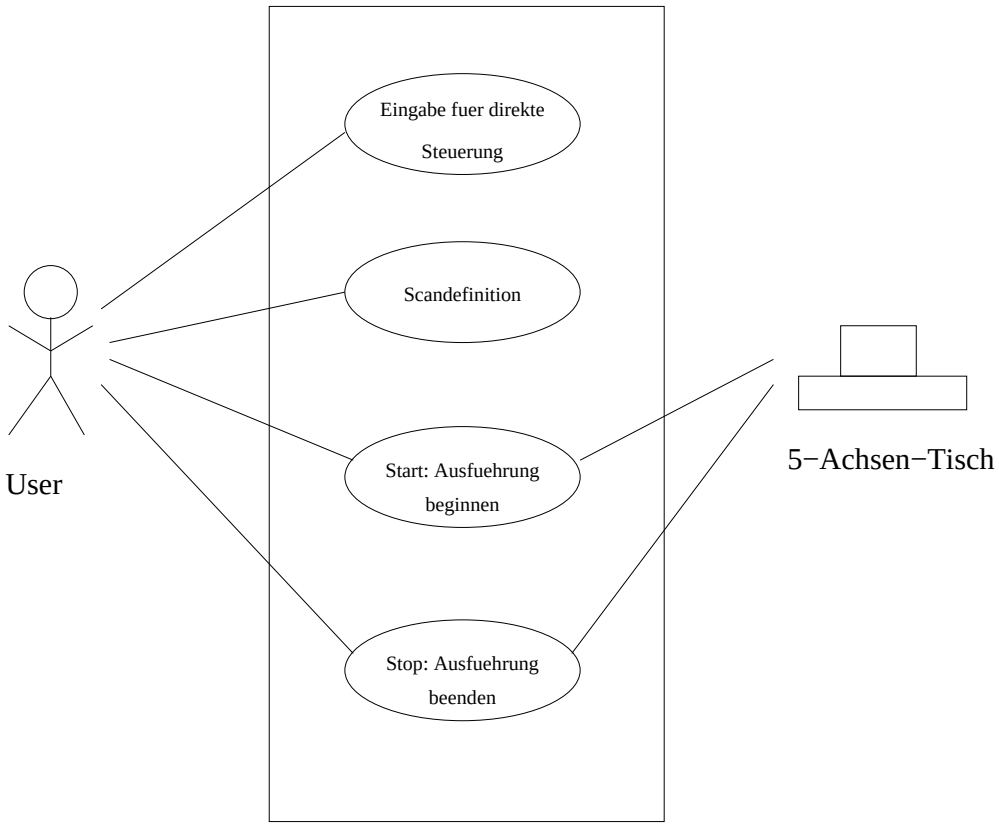

Abbildung 1: Usecase diagram fuer die Applikation Tabel Control

Wir betrachten nun detailiert diverse Aktionen des users.

T: "Tablecontroll"

B: Benutzer

Direkte Steuerung des Positioniertisches

B: startet TableControl auf.

B: wählt Instrument aus (RTOF, DFMS, COPS).

T: Setzt entsprechenden Parameter.

B: wählt Betriebsart “Manuelle Steuerung”.

T: wechselt GUI auf “Manuelle Steuerung”.

B: gibt Relativ- bzw. Absolutkoordinaten von einer oder mehreren Achsen ein (Annahme: Endpunkt liegt in verbotenem Bereich).

T: gibt Fehlermeldung aus.

B: korrigiert seine Eingabe.

T: überprüft Eingabe → ok.

T: erzeugt aus den Benutzereingaben ein Textfile mit den Steuerbefehlen.

Definieren eines Kugelscans

B: wählt Instrument aus (RTOF, DFMS, COPS).

T: setzt entsprechende Parameter.

B: wählt Betriebsart “Erstellen eines Scans”.

B: wählt “Kugelscan”.

T: wechselt GUI auf “Scan auf Kugeloberfläche”.

B: gibt die Kugelparameter ein.

T: überprüft, ob dieser Wert für das gewählte Instrument im erlaubten Bereich liegt (Annahme: Falsche Eingabe).

T: verlangt Korrektur der Werte.

B: gibt erlaubte Werte ein.

T: überprüft Eingabe → ok.

T: erzeugt aus den Benutzereingaben ein Textfile mit den Steuerbefehlen.

Definieren eins Ebenenscans

B: wählt Instrument aus (RTOF, DFMS, COPS).

T: setzt entsprechende Parameter.

B: wählt Betriebsart “Erstellen eines Scans”.

B: wählt “Ebenenscan”.

T: wechselt GUI auf “Scan auf Ebene”.

B: gibt die Ebenenparameter ein.

T: überprüft Eingabe → ok.

T: erzeugt aus den Benutzereingabe ein Textfile mit den Steuerbefehlen.

Definieren eines Winkelscans

B: wählt Instrument aus (RTOF, DFMS, COPS).

T: setzt entsprechende Parameter.

B: wählt Betriebsart “Erstellen eines Winkelscans”.

T: wechselt GUI auf “Erstellen eines Winkelscans”.

B: gibt die Parameter ein.

T: überprüft Eingabe → ok.

T: erzeugt aus den Benutzereingabe ein Textfile mit den Steuerbefehlen.

1.4 Analyse

1.4.1 Machbarkeit

TableControl ist in der im Pflichtenheft beschriebenen Form realisierbar. Als Programmiersprache wird Java gewählt. Die Hauptschwierigkeiten liegen in der Erarbeitung der Mathematik für die verschiedenen Koordinatentransformationen und in der Koordination der diversen nebenlaufenden Prozesse bzw. Threads.

1.4.2 Prototyp

In einer ersten Phase soll ein Prototyp erstellt werden, der die direkte Steuerung des Positioniertisches durch Eingabe von Koordinaten (Relativ- bzw. Absolutkoordinaten) erlaubt. Der Prototyp soll die Möglichkeit bieten, den Tisch für Montagearbeiten zu positionieren sowie Erfahrungen im Umgang mit dem 5-Achsentisch zu sammeln. Diese fliessen fortlaufend in den Entwicklungsprozess ein.
Steuerbefehle werden zur Sicherheit vorerst nur in einem Terminalfenster ausgegeben.

2 Design

2.1 Klassen

Die Klassen werden in alphabetischer Reihenfolge vorgestellt. Der Sourcecode kann in Anhang A eingesehen werden.

Axis: In der Klasse "Table" werden fünf Instanzen dieser Klasse definiert. Diese entsprechen weitgehend den realen Achsen des 5-Achsentisches.

ComClass: Diese Klasse dient zur Kommunikation via serieller Schnittstelle. Sie stellt sicher, dass die Schnittstelle für die Datenübertragung jeweils "gelockt" und anschliessend wieder freigegeben wird.

Dialog_AngleScan: Userinterface zur Definition eines Winkelscans.

Dialog_PlaneScan: Userinterface zur Definition eines Ebenenscans.

Dialog_Scan: Abstrakte Klasse als Superklasse für die Userinterfaces der Scandefinitionen.

Dialog_SphereScan: Userinterface zur Definition eines Kugelscans.

Dialog_TableControl: Haupt-Userinterface der Applikation.

FileChooser: Ermöglicht das öffnen und speichern von Scanfiles via Userinterface.

Instrument: Eine Instanz dieser Klasse ist ein Teil des 5-Achsentisches. Die verschiedenen Instrumente werden in der Klasse "Table" erzeugt.

MyFileThread: Thread-Klasse, die das abarbeiten von Scanfiles ermöglicht.

MyKeyListener: Implementation von KeyListener. Zum Abfangen von fehlerhaften Eingaben in Textfeldern.

MyTextField: Angepasstes Textfeld, das über die Möglichkeit verfügt, einen Bereich von zu akzeptierenden Werten anzugeben.

PlaneScan: Schreibt aus den via Userinterface "Dialog_PlaneScan" eingegebenen Werten den kompletten Scanablauf für einen Ebenenscan in ein entsprechendes Scanfile.

Point3D: Mit diese Klasse wird ein Punkt im 3 dimensionalen Raum durch eine Matrix dargestellt.

PosDisplay: Thread-Klasse zur Anzeige der aktuellen Position einer Achse. Für jede Achse gibt es einen einzelnen Thread.

ProtoFrame_AboutBox: Dialog-Klasse die Informationen zur Applikation liefert. Wird via "Help"-Feld des Menüs gestartet.

ScanFilter: Erweiterung der Klasse FileFilter. Dient als Filter für Files, deren Extension in der Klasse Utils angegeben sind. (Hier bisher nur .scan).

Scan: Abstrakte Superklasse der Klassen "PlaneScan" und "SphereScan".

SingleMove: Klasse zur Definition einzelner Fahrbefehle.

SphereScan: Schreibt aus den via Userinterface "Dialog_SphereScan" oder "Dialog_AngleScan" eingegebenen Werten den kompletten Scanablauf für einen Kugel- oder Winkelscan in ein entsprechendes Scanfile.

TableControl: Hauptklasse mit der Methode "main".

Table: Implementation des 5-Achsentisches mit den Definitionen der Achsen und der Instrumente.

Transformationen: Hilfsklasse mit Methoden für Umrechnungen.

Utils: Hilfsklasse zum Herausfinden der Extension von Files.

2.2 Klassendiagramm

Das Klassendiagramm (Abb. 2) zeigt den Aufbau der Applikation. Um einen noch besseren Überblick über die verwendeten Klassen und deren Interaktionen zu erhalten, wird an dieser Stelle auf die mittels javadoc erstellte Html-Dokumentation verwiesen. Die entsprechenden Files sind im mitgelieferten directory 'javadoc' zu finden.

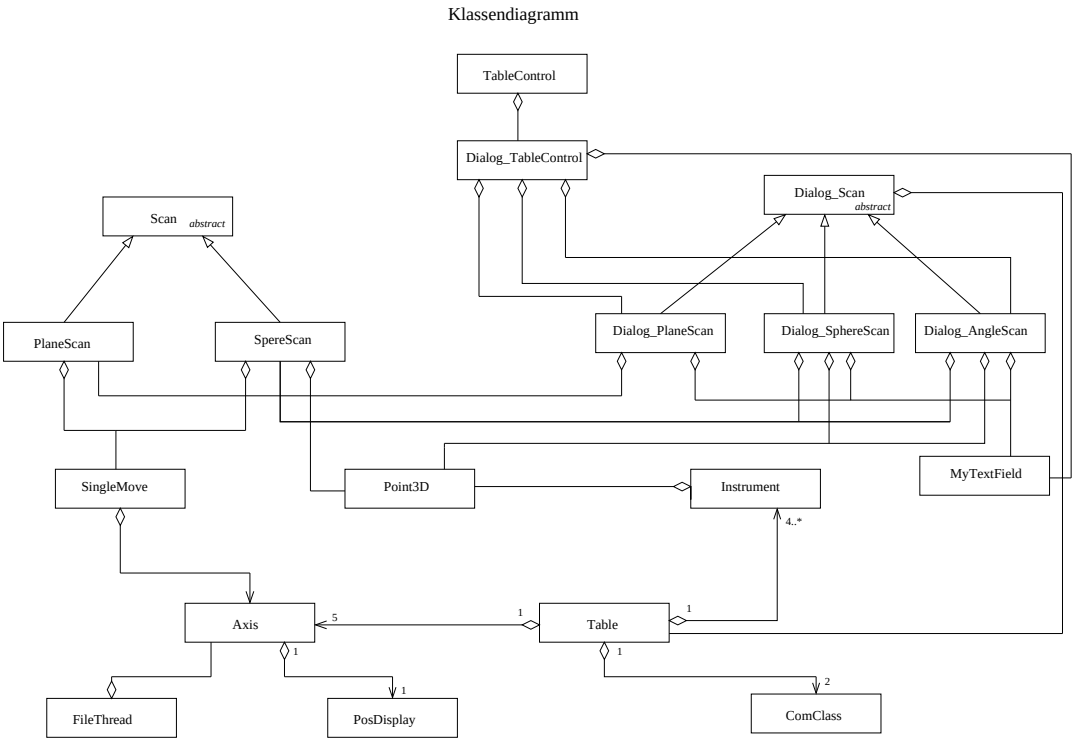

Abbildung 2: Klassendiagramm der Applikation TableControl

2.3 Sequence diagrams

Einen Überblick über die Interaktion der Objekte der Anwendung geben uns sequence diagrams. In der Folge werden Diagramme für die beiden use cases 'start ' und 'stop' gezeigt. Dabei stehen die vertikalen Linien für die erzeugten Objekte und die Pfeile zwischen den Objekten stehen für den Aufruf einer Methode des Zielobjektes vom Startobjekt aus.

2.3.1 Sequence diagram für 'start'

Wir nehmen an, dass alle Angaben für die direkte Steuerung bereits korrekt eingegeben wurden.

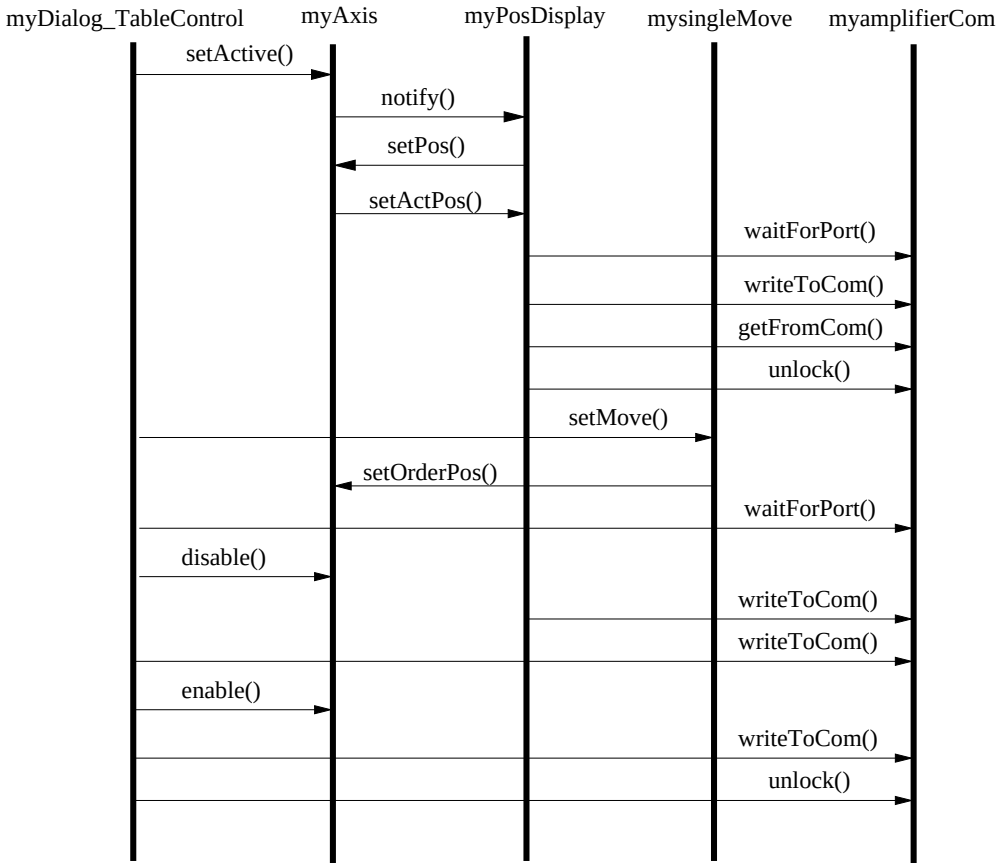

Abbildung 3: Sequence diagram für den use case 'start'.

2.3.2 Sequence diagram für 'stop'

Wir nehmen an, dass der Tisch einen Fahrbefehl ausführt, jedoch aus irgend einem Grund dabei gestoppt werden muss.

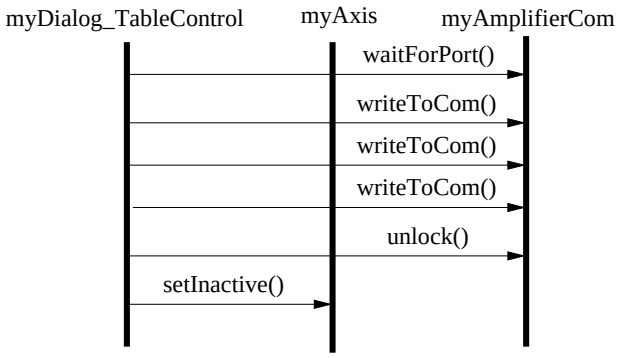

Abbildung 4: Sequence diagram für den use case 'stop'.

3 Mathematik

In diesem Abschnitt werden die notwendigen mathematischen Aspekte besprochen.

3.1 Situationsskizze

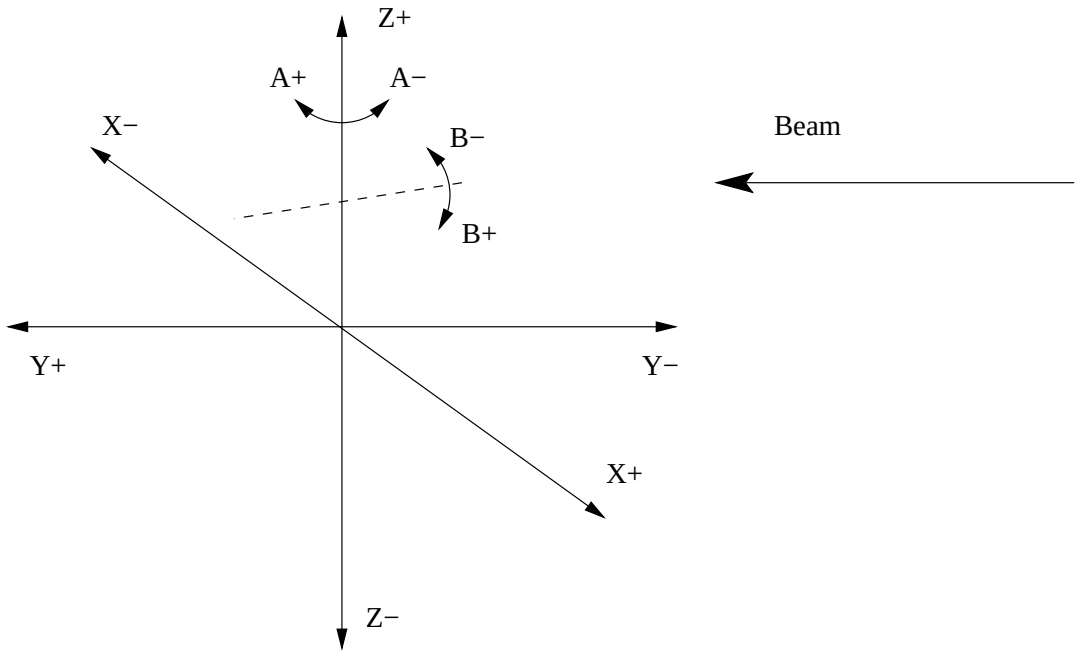

Abbildung 5: Situationsskizze

Wie auf der Skizze zu sehen ist, bilden die 3 kartesischen Achsen(X,Y,Z) ein Linkssystem. Die Y-Achse steht dabei parallel zur Strahlenrichtung (Beam). Die Rotationsachse A ist identisch zur Z-Achse. Die Rotationsachse B steht parallel zur XY-Ebene und ist von der aktuellen Position der A-Achse abhängig, d.h. wird bei Drehung von A mitgedreht.

3.2 Motivation

Aus den von den Benutzern definierten Scans muss eine Programmabfolge generiert werden, die in der gewünschten Weise Punkte im Strahl mit der entsprechenden Instrumentenausrichtung anfahren kann. Die einzelnen

Punkt/Richtungs-Kombinationen sollen mit Hilfe der hier beschriebenen Mathematik auf geeignete Weise zu einer Programmabfolge zusammengestellt werden.

3.3 Konzept

Einheiten: Die kleinsten Einheiten des 5-Achsetisches sind μm und μ^o. Die Eingaben in den Dialog-Fenstern erfolgt jedoch in mm und ^{o}. Auch bei den Berechnungen werden letztere Einheiten verwendet. Die Umwandlung in Tisch-Einheiten geschieht erst bei der Erstellung der Fahrbefehle.

Zur Identifikation der Instrumentenachsen im Raum genügen die Koordinaten eines Referenzpunktes sowie die Winkel (A, B) der Instrumentenachsen. Die Instrumentenachse ist diejenige Achse, die normalweise parallel zum Strahl ausgerichtet ist.
Als Referenzpunkt wir der Mittelpunkt der Interface-Fläche zur Kalibrieranlage gewählt. Die beiden Winkel ergeben sich aus den notwendigen Korrekturen in A- und B-Richtung, um die Instrumentenachse parallel zum Strahl auszurichten.
Die Bewegungen der Tischachsen werden nach folgendem Schema berechnet: Zuerst wird der Sollpunkt für den Referenzpunkt bestimmt. Das ist also jener Punkt, wo der Referenzpunkt während der Messung plaziert sein soll.

Ebenenscan: Bei diesem Scan, wo die Instrumentenachse parallel zum Strahl ist, wird die Differenz zwischen der aktueller Position des Referenzpunktes und dem Sollpunkt berechnet. Dadurch ergeben sich die relativen Bewegungen in den kartesischen Koordinaten X, Y, und Z.

Kugel-/Winkelscan: Zuerst müssen beim Kugelscan die Koordinaten des Sollpunktes, der auf einer Kugeloberfläche liegt, berechnet werden.
Anschliessend wird nun die Instrumentenachse um eine Drehachse gedreht. Da dies eine Verschiebung des Referenzpunktes zur Folge hat, muss die aktuelle Position des Referenzpunktes berechnet werden. Aus der Differenz von diesem und dem Sollpunkt werden nun die relativen Bewegungen in den kartesischen Koordinaten bestimmt.

Oben erwähnte Berechnungen werden mit Hilfe von Matrizen und Vektoren ausgeführt. Es werden grundsätzlich zwei Bewegungsarten unterschieden:

- Translation: X, Y, Z Achsen

- Rotation: A, B Achsen

3.3.1 Translation:

Das Bild einer Translation kann auf zwei Arten berechnet werden: Multiplikation mit einer Diagonalmatrix, Addition eines Vektors.
Da die Implementation einfacher ist, wird die Translation mit Hilfe eines Translationsvektors ausgeführt, der als Einträge die relativen Achsenbewegungen hat.

3.3.2 Rotation:

Hier müssen die beiden Achsen unterschieden werden, da sie sich nicht analog Transformieren. Es muss beachtet werden, dass es sich beim 5-Achsentisch um ein Linkssystem handelt.

A-Achse: Das Bild einer Rotation um die A-Achse kann mit einer Rotationsmatrix ausgeführt werden, da die Rotationsachse gerade mit der z-Achse übereinstimmt.

B-Achse: Die Rotationsmatrix einer solchen Transformation ist schwieriger zu bestimmen, da die Rotationsachse von der aktuellen Position der A-Achse abhängt. Um die Rotationsmatrix zu erhalten, muss ein Koordinatenwechsel ausgeführt werden, der die Position der A-Achsen berücksichtigt.

Die nun beschriebenen mathematischen Objekte können im Anhang B betrachtet werden. Die dort aufgeführten Berechnungen wurden mit Mathematica ausgeführt und dienten zur Entwicklung und Kontrolle der in der Applikation verwendeten mathematischen Operationen.

Kollisionsabsicherung Drehungen um Winkel $> 5^o$ werden in Teildrehungen zerlegt. Nach jeder solchen Drehung soll dann die Korrektur in den kartesischen Koordinaten erfolgen, um den Referenzpunkt wieder an seine ursprüngliche Stelle zu bringen. Durch dieses Verfahren wird vermieden, dass bei einer grossen Drehung der Versatz den Bewegungsraum des Balgs verlässt und etwas zerstört wird.

4 Tests

Neue Klassen wurden so früh wie möglich getestet und ins Programmgefüge eingebaut.
Bisher war es aus organisatorischen Gründen noch nicht möglich, die Applikation an der Anlage zu testen. Bei den ersten Tests werden sicher noch einige Probleme auftreten. Durch die objektorientierte Struktur der Applikation, sollte es leicht möglich sein, Modifikationen anzubringen.

5 Programmdokumentation

Anhand von Screenshots soll in diesem Abschnitt eine kurze Einführung in die Installation und Bedienung der Applikation gegeben werden.
Die Screenshots sind im Anhang C zu finden.

Installation Die Datei TableControl.tar wird in das gewünschte Verzeichnis kopiert und mit der Befehlszeile tar -xvf TableControll.tar installiert. Dabei werden alle nötigen Unterverzeichnisse automatisch erstellt.

Programmstart Nachdem von Installationsverzeichnis ins Verzeichnis java gewechselt wurde, kann mit der Befehlszeile java TableControl die Applikation gestartet werden.

5.1 Hauptfenster

Das Hauptfenster bietet die Möglichkeit, einzelne Fahrbefehle oder ganze Scanabläufe zu starten. Dabei können auf der rechten Seite die verschiedenen Parameter für die direkte Steuerung eingegeben und mit dem Button "übernehmen" dem System übergeben werden. Andererseits kann auf der linken Seite zum Dialog zur Eingabe eines Scanablaufes gewechselt oder ein schon bestehender Scanablauf geöffnet werden.
Links unten kann nun der gewählte Betrieb gestartet und wieder gestoppt werden.
Ein Monitoring-Fenster, dass sich rechts unten befindet, zeigt dem User wichtige Informationen an.

5.2 Scandefinition

Vom Hauptfenster gelangt man auf Tastendruck zur Definition einer der 3 verschiedenen Scanabläufe. Die drei Dialogfenster haben alle die gleiche Funktionalität.
Ein Scanablauf kann durch Eingabe der entsprechenden Parameter neu definiert werden.
Zum editieren eines bereits vorhandenen Scanablaufs wird auf der rechten Seite das betreffende File geöffnet.
Nach der Neudefinition oder der Anpassung wird der Scanablauf mittels dem entsprechenden Button auf der rechten Seite gespeichert.

5.2.1 Scanfile

Es wurde ein File generiert, dass zwar alle Befehle zum Anfahren der verschiedenen Scanpunkte beinhaltet, jedoch noch keine Instrumentenbefehle enthält. Ein Ausschnitt aus einem generierten Scanfile kann im Anhang D betrachtet werden. Wie dort bereits angedeutet, müssen die Instrumentenbefehle jeweils mit einem Semikolon beginnen.

5.2.2 Fahrbefehle

Die Fahrbefehle werde mittels serieller Schnittstelle direkt an die Servoverstärker der einzelnen Achsen geschickt.

Beispiel Fahrbefehl:

\ **0** : Wahl des entsprechenden Servoverstärkers.
Order 1 8192 1000 1000 500 500 0 0 : eigentlicher Fahrbefehl.
Move 1: ausführen des Fahrbefehles.

Wer den ganzen Befehlssatz des Servoverstärkers und die Aufschlüsselung der Fahrbefehle einsehen möchte, sei hier auf [1] bzw. [2] verwiesen.

6 Auszuführende Arbeiten

6.1 Korrekturen/Ergänzungen

Bis zum heutigen Zeitpunkt war es nicht möglich die Applikation direkt am 5-Achsentisch zu testen. Einige Parameter müssen zuerst bei Testläufen ermittelt werden. Im folgenden Abschnitt wird auf die noch auszuführenden Arbeiten eingegangen.

Eintragen von Parametern: Bei den Dialog Klassen wird die Klasse MyTextField verwendet. Diese Instanzen dieser Klasse besitzen einen Array range bei dem spezifiziert werden kann, welche Zahlen [min,max] bei der Eingabe akzeptiert werden. Die ranges sind bisher defaultmässig auf einen bestimmten Bereich eingestellt. Diese Bereiche sollten angepasst werden.
Durch sorgfältiges Wählen der Bereiche kann ein wichtiger Beitrag zur Kollisionsverhinderung geleistet werden.

Klasse Table: Für die Instrumente müssen jeweils die Referenzen (z.B. RTOF_ref) korrekt eingetragen werden. Der vertikale Offset der B-Achse vom Nullpunkt des Tisches wird im Feld offset_z gesetzt. Dasselbe Feld muss auch in der Klasse point3D neu gesetzt werden. Die X- und Z-Koordinaten des Zentrums des Strahls müssen im Feld beam eingetragen werden.

Klasse ComClass: Bei dieser Klasse müssen die Methoden writeToCom(String s) und getFromCom() implementiert werden. Die beiden Instanzen dieser Klasse müssen in Table korrekt initialisiert werden.

Klasse Axis: In dieser Klasse ist die Methode inPos() zu implementieren. Sie gibt "true"zurück, falls die betreffende Achse ihre Sollposition erreicht hat. Dazu muss ein digitaler Ausgang des Servoverstärkers "abgehört" werden. Nähere Informationen zu den digitalen Ausgängen findet man auf den Seiten 10/11 von [2].

6.2 Mögliche Erweiterungen

Scanmanagement: Bisher fehlt beim Management von Scans eine wichtige Funktionalität: Ein Scan kann nur gestartet und gestoppt, jedoch nicht unterbrochen und anschliessend wieder aufgenommen werden.
Dazu müsste in der Klasse MyFileThread eine entsprechende Erweiterung implementiert werden. Im Wesentlichen braucht es einen Zeiger, der die aktuelle Position im Scanfile kennt.

Instrumentenbefehle: Die Instrumentenbefehle müssen noch ohne die Hilfe eines Dialoges direkt in das jeweilige Scanfile eingegeben werden. Es könnte sich als hilfreich erweisen, diese Eingabe mittels Dialogfenster zu gestalten.

7 Fazit des Informatikprojektes

Das Informatikprojekt begann im November 1999. Es war mein erstes grosses Projekt, das ich selbstständig zu bewältigen hatte. Einige Rahmenbedingungen haben die Arbeit wesentlich erschwert. Da ich Informatik als Nebenfach belege, konnte ich neben dem Hauptfach Physik nur wenige Stunden pro Woche am Projekt arbeiten. Ich selbst setzte mir die Rahmenbedingung, die Arbeit auf einem zur Verfügung gestellten Linuxrechner zu erledigen. Da ich bisher noch nie mit Linux gearbeitet hatte, musste ich einige Stunden in die Einarbeitung auf diesem Betriebssystem investieren. Erschwerend kam dazu, dass der zur Verfügung gestellte Rechner eher Leistungsschwach war und deshalb einige nützliche Entwicklungsumgebungen wie 'SNIFF' oder 'JBuilder' unbrauchbar langsam liefen. Schliesslich entwickelte ich auf dem leistungsfähigen Editor 'Xemacs' und musste dabei auf viele nützliche Tools wie z.B. einen Debugger verzichten. Auch die Programmiersprache 'Java' war für mich neu. Tutorials auf der Java Homepage (developer.java.sun.com) erwiesen sich als sehr hilfreich. Für Java-Anfänger kann ich diese cite sehr empfehlen.
Die Spezifikation der Anforderungen zu Beginn der Arbeit nahm mehr Zeit in Anspruch als erwartet, da sich die Kunden noch gar kein klares Bild von der zu entwickelnden Applikation gemacht hatten. Es blieb mir somit nichts anderes übrig, als mir mit den wenigen Kundeninformationen selbst ein Bild über die Anforderungen zu machen und dann das Pflichtenheft von den Kunden überprüfen zu lassen. Es wurde mir auch bewusst, dass die Spezifikation der Anforderungen nie definitiv abgeschlossen werden konnte, da der Kunde einige notwendigen Features der Applikation erst mit dem Benutzen erkennen würde.
Daher war es für mich nun auch praktisch erwiesen, dass ich möglichst bald einen ersten Prototypen erstellen musste, damit früh Erfahrungen im Umgang mit dem Programm gesammelt werden konnten. Die Zusammenarbeit mit den Kunden erwies sich auch in diesem Fall wieder als schwierig, da sie sich kaum Zeit zum testen der Prototypen nahmen und somit wichtiges Feedback ausblieb. Bei einem zukünftigen Informatikprojekt werde ich vermehrt die Kunden dazu drängen mit dem Prototypen zu arbeiten und mir oft ein Feedback zu geben.
Mit fortschreitender Zeit wurde mir die Dynamik des Entwicklungsprozesses von Software immer mehr bewusst. Die Untauglichkeit des veralteten Wasserfallmodelles, bei dem erst zum nächsten Entwicklungsprozess übergegangen wird, wenn der vorhergehende Schritt bereits abgeschlossen ist, kam deutlich zum Vorschein. Nur die stetige Interaktion zwischen den einzelnen Entwicklungsschritten erwies sich als erfolgreich. Was ich eigentlich schon von der

Theorie her wusste, bekam nun viel mehr Gewicht.
Das Projekt war so aufgeteilt, dass ich den Hauptteil der Applikation zu entwickeln hatte, und ein Mitarbeiter des Kunden die Kommunikation des Programmes mit den Servoverstärkern des 5-Achsentisches zu implementieren hatte. Diese Aufteilung erwies sich als sehr schlecht, da die benötigte Kommunikationsschnittstelle nicht rechtzeitig fertiggestellt wurde. So hatte ich keine Möglichkeit, meine Prototypen zur Steuerung des Tisches zu testen. In Zukunft werde ich auf eine sinnvollere Aufteilung eines Projektes achten müssen.
Den Hauptteil der Dokumentation zur Applikation erstellte ich erst gegen Ende der Projektarbeit. Das führte dazu, dass ich viele wichtige Details zum Design und zum Code wieder mühsam aufarbeiten musste. Obwohl ich durch Vorlesungen auf die Wichtigkeit des frühzeitigen Dokumentierens aufmerksam gemacht wurde, waren mir Resultate beim der Applikation wichtiger. Eine falsche Vorgehensweise, die sich in vielen zusätzlichen Arbeitsstunden auswirkte. Auch die lange Dauer der Projektarbeit führte zu einem grossen Mehraufwand, da ich mich nach diversen Unterbrüchen immer wieder neu einarbeiten musste.
Ich hoffe, dass meine Nachfolger in Sachen Projektarbeit von meinen Erfahrungen profitieren können und nicht auch die gleichen Fehler begehen.

A Quellcode

```java
/**
 * Axis.java
 *
 *
 * Created: Sun Jun  4 20:24:04 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * In der Klasse 'Table' werden fuenf Instanzen dieser Klasse definiert.
 * Diese entsprechen weitgehend den realen Achsen des 5-Achsentisches.
 *
 */


public class Axis {
    //there are  5 possible axis to generated: X, Y, Z, A, B
    //this is theire superclass
    //but every axis only can exist once.
    private SingleMove myMove;
    public Table myTable;
    private long max_value;
    private long min_value;
    private double j = 0     ;// for tests only
    public String myId;
    public double myPos;
    public long soll_pos;
    public PosDisplay myPosDisplay;
    private boolean isActive = false;
    Thread sleepThread;

    private long orderPos;


    public Axis (String id, Table t) {
    //this constructor generates an axis of type myId and
        //puts it to the specific Table t
    myTable = t;
    myId = id;
    myPosDisplay = new PosDisplay(this);
    myPos = 0; // nur fuer test, sonst  myPos = getPos();
    sleepThread = new Thread();
    sleepThread.setPriority(1);
    }


    public void enable() {
    //Enables the axis
        // myTable.amplifierCom.waitForPort();
    myTable.amplifierCom.writeToCom(myId);
    myTable.amplifierCom.writeToCom("EN");
        // myTable.amplifierCom.unlock();
    //myThread.resume();
    }


    public void disable() {
    //set it to Disable
        // myTable.amplifierCom.waitForPort();
    myTable.amplifierCom.writeToCom(myId);
    myTable.amplifierCom.writeToCom("DIS");
        // myTable.amplifierCom.unlock();
    //    myThread.suspend();
    }

    public void startMove() {
    //starts the specified Move
        enable();
        myTable.amplifierCom.writeToCom("MOVE 1");
    }

    public void stopMove() {
    //stops the current move without the possibility to resume later on
    //sets the current axis to disable
        myTable.amplifierCom.writeToCom(myId);
        myTable.amplifierCom.writeToCom("STOP");
```

```
77          disable();
78      }
79
80      public synchronized void setPos(PosDisplay pd) {
81      while (!isActive()) {
82          try {
83          wait();
84          }
85          catch (InterruptedException e) {
86          System.out.println(e.toString());
87          }
88      }// end while
89      //  System.out.println("activ");
90      pd.setActPos();
91      }
92
93
94      public boolean inPos() {
95      try {
96          sleepThread.sleep(500);
97      }
98      catch (InterruptedException e1) {
99      }
100     boolean inPos = true; // to be modified
101     // digital out of amplifier shows, that axis is 'inPos'
102     // to be implemented by chris
103     myPosDisplay.setActPos();
104         System.out.print("orderPos=");
105         System.out.print(orderPos);
106         System.out.print("  getPos=");
107     System.out.println(getPos());
108         // System.out.println(Transformation.double2long(getPos()));
109
110         if (Math.abs(orderPos-getPos()) < 4.0 ) {
111             inPos=true;
112             System.out.println("  inPos is true");
113         } else {
114             inPos=false;
115             System.out.println("  inPos is false");
116         }
117
118
119
120     if (inPos)
121     setInactive();
122     return inPos;
123     }
124
125     public synchronized void setActive() {
126     isActive = true;
127     notify();
128     }
129
130     public boolean isActive() {
131     return isActive;
132     }
133
134     public void setInactive() {
135     isActive = false;
136     }
137
138   public void init() {
139     this.setActive();
140     try {
141         sleepThread.sleep(300);
142     }
143     catch (InterruptedException e) {
144         System.out.println(e.toString());
145     }
146     //  myPosDisplay.setActPos();
147     setInactive();
148     }
149
150     public double getPos() {
151     return myPosDisplay.actPos;
152     }
```

```
153
154   public void setOrderPos(long pos) {
155
156      orderPos = pos;
157
158   }
159
160  }// End class Axis
161
```

```java
/**
 * ComClass.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Diese Klasse dient zur Kommunikation via serieller Schnittstelle. Sie
 * stellt sicher, dass die Schnittstelle f"ur die Daten"ubertragung
 * jeweils `gelockt' und anschliessend wieder freigegeben wird.
 *
 */

import java.io.*;
import java.util.*;
import javax.comm.*;


public class ComClass {
    private long baudrate;
    private String myId;
    private String port = "/dev/ttyS1";

    private CommPortIdentifier portId;
    private static SerialPort commPort;
    private static OutputStream outStream;
    private static InputStream inpStream;

    private static boolean lock;
    private static boolean access;
    private static String echoString;



    public ComClass(String id, long rate) {
    baudrate = rate;
    myId = id;

        System.out.println(myId);
        if (myId.equals("com1")) {                  // com1 Id for table
            System.out.println("open port " +port + "...");

            try {
                portId = CommPortIdentifier.getPortIdentifier(port);
            } catch (NoSuchPortException e){
                System.out.println("Port not found");
                System.exit(1);
            }
            System.out.println("Port id ("+port+") = "+portId);

            try {
                commPort = (SerialPort) portId.open("tableApp",2000);
            } catch (PortInUseException e) {
                System.out.println("Port in use");
                System.exit(1);
            }

            System.out.println("Set to 9600 baud, 8 data bits, 1 stop bit, no parity");

            try {
                commPort.setSerialPortParams(9600, SerialPort.DATABITS_8,SerialPort.STOPBITS

            } catch (UnsupportedCommOperationException e) {
                System.out.println("...failed");
                commPort.close();
                System.exit(1);
            }

            System.out.print("open input stream ...");
            try {
                inpStream = commPort.getInputStream();
            } catch (IOException e) {
                System.out.println("no input stream");
                commPort.close();
                System.exit(1);
```

```
77              }
78              System.out.println(" ok");
79
80              System.out.print("open output stream ...");
81              try {
82                  outStream = commPort.getOutputStream();
83              } catch (IOException e) {
84                  System.out.println("no output stream");
85                  commPort.close();
86                  System.exit(1);
87              }
88              System.out.println(" ok");
89              lock = false;
90              access = false;
91          }
92
93      }
94
95      protected void finalize() {
96
97          System.out.println("Closing serial port");
98          commPort.close();
99
100     }
101
102     public static void waitForPort() {
103         // wait until the serial port is free
104         while(lock) ;
105
106         access = true;
107         System.out.println("port will be used");
108     }
109
110     public static void unlock() {
111         // free serial port
112
113         lock = false;
114         access = false;
115         System.out.println("port is unlocked");
116     }
117
118     private static void readEcho () {
119         // reads echo and waits for -->
120
121        char[] charBuffer = new char[512];
122        char c;
123        int numBytes;
124        String s;
125        int pos=0;
126        int counter = 0;
127        InputStreamReader  isr = new InputStreamReader(inpStream);
128        BufferedReader br = new BufferedReader(isr);
129        c = 'n';
130
131        // echoString = "PFB\n-34567\n-->";
132        // if (0 < 1) return;
133
134
135
136        try {
137              while (c  != '>' ) {
138             // wait for echo
139                 while (!br.ready()) ;
140             // counter = 0;
141             // while (!br.ready() && counter <1000 ) {
142             //     counter++;
143         //     System.out.print(".");
144         // }
145
146
147         c = (char) br.read();
148
149
150                 // System.out.println("echo read:" + c +"***");
151         charBuffer[pos++] = c;
152
```

```
153          //  System.out.println("counter:"+ Integer.toString(pos));
154
155              }
156
157              echoString = new String(charBuffer);
158          System.out.println("string read:" + echoString +"***");
159
160
161         } catch (IOException e) {
162                      System.out.println("IO Exception");
163              //        commPort.close();
164         }
165
166      }
167
168      public static synchronized void writeToCom(String s) {
169          // writes to the servo-amplifier, to be implemented by chris
170      //for tests writes to stdout, "\n" has to be replaced by 'return'
171
172          byte answer[] = new byte[10];
173          int n_read = 0;
174
175          // System.out.println(myId);
176
177          lock = true;
178          if (access == false)
179              System.out.println("*** port should not be used without flag ***");
180
181          System.out.print(s + "\n");
182          System.out.print("send above command to table y/n ?  ");
183
184          //try {
185          //    n_read=System.in.read(answer);
186          //} catch (IOException e) {
187          //    System.out.println("*** keyboard exception ***");
188          //}
189      //
190      answer[0] = 'y';
191
192          switch((int)answer[0]) {
193              case 'y':
194              case 'Y': s = s + "\r";
195                  try {
196                      outStream.write(s.getBytes());
197                      readEcho();
198                  } catch (IOException e) {
199                      System.out.println("IO Exception");
200                      commPort.close();
201                  }
202                  break;
203
204              default: System.out.println("no command sent");break;
205          }
206
207
208
209      }
210
211      public static synchronized double getFromCom() {
212         byte[] readBuffer = new byte[512];
213         int numBytes;
214         String s = echoString;
215         String l;
216         StringBuffer sb;
217         int pos = 0;
218         double value = 0.0;
219
220         lock = true;
221         if (access == false)
222              System.out.println("*** port should not be used without flag ***");
223
224         // try {
225             //while (inpStream.available() > 0) {
226             //    numBytes = inpStream.read(readBuffer);
227             //}
228             // s = new String(readBuffer);
```

```
229             // s = "-45345\n-->aabb\n-->aacc";
230 
231             System.out.println("data read:" + s);
232 
233             sb = new StringBuffer(s);
234             pos = s.indexOf('\n');
235             System.out.println("pos:"+ Integer.toString(pos));
236 
237             while (pos != -1) {
238                  l =  s.substring(0,pos);
239                  System.out.println("substring:"+l);
240 
241                  try {
242                      value = Double.parseDouble(l);
243 
244                  } catch (NumberFormatException e) {
245                      System.out.println("not an number");
246                  }
247                  sb = new StringBuffer(s);
248                  sb.delete(0,pos+1);
249                  s = sb.toString();
250                  System.out.println("shorter s:"+s);
251                  pos = s.indexOf('\n');
252                  System.out.println("pos:"+ Integer.toString(pos));
253 
254                  // System.out.println("data read: " + new String(readBuffer));
255             }
256 
257       // } catch (IOException e) {
258       //              System.out.println("IO Exception");
259       //              commPort.close();
260       //           }
261 
262 
263     // reads from the servo-amplifier, to be implemented by chris
264     //for tests writes to stdout, "\n" has to be replaced by 'return'
265          return value;
266     }
267 
268 }//End class Translation
269 
```

```java
/**
 * Dialog_AngleScan.java
 *
 * Created: Sun Jun  4 20:37:07 2000
 *
 * @author Roland Schaefer
 *
 * Userinterface zur Definition eines Winkelscans.
 */


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.util.*;


public class Dialog_AngleScan extends Dialog_Scan {

    // declaration of swing components
    MyTextField xField, yField, zField;
    MyTextField alphaMin, alphaMax, numAlphaText;
    MyTextField betaMin, betaMax, numBetaText;
    // end declaration of swing components



    //declaration of other components
    private static String newline = "\n";
    Point3D center;
    long a_range[]= new long[2];
    long b_range[]= new long[2];
    long na, nb, radius;
    // end declaration of other components


    // constructor of Frame
    public Dialog_AngleScan(Table table) {
    super(table, "WinkelScan");
    }



    public void initParameter() {

    // fixpoint choice
    JPanel fixPoint_choice = new JPanel(new FlowLayout(FlowLayout.CENTER,
                                10, 30));
    fixPoint_choice.setPreferredSize(new Dimension(150, 200));
    fixPoint_choice.setBorder(
              BorderFactory.createCompoundBorder(
                  BorderFactory.createTitledBorder("FixPunkt"),
                  BorderFactory.createEmptyBorder(-20,0,0,0)));

    JLabel xLabel = new JLabel("x:");
    JLabel yLabel = new JLabel("y:");
    JLabel zLabel = new JLabel("z:");

    double xRange[] = {-1000, 1000};
        xField = new MyTextField(8, xRange);
    xField.addKeyListener(new MyKeyListener(xField));

    double yRange[] = {-1000, 1000};
        yField = new MyTextField(8, yRange);
    yField.addKeyListener(new MyKeyListener(yField));

    double zRange[] = {-1000, 1000};
        zField = new MyTextField(8, zRange);
    zField.addKeyListener(new MyKeyListener(zField));

    fixPoint_choice.add(xLabel);
    fixPoint_choice.add(xField);
    fixPoint_choice.add(yLabel);
    fixPoint_choice.add(yField);
```

```
77      fixPoint_choice.add(zLabel);
78      fixPoint_choice.add(zField);
79      parameterPanel.add(fixPoint_choice);
80      // end fixPoint_choice
81
82      //alpha_range
83      JPanel alpha_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
84                              10, 30));
85      alpha_range.setPreferredSize(new Dimension(150, 150));
86
87      JLabel alphaMinLabel = new JLabel("min:");
88      JLabel alphaMaxLabel = new JLabel("max:");
89
90      double aMinRange[] = {-1000, 1000};
91          alphaMin = new MyTextField(6, aMinRange);
92      alphaMin.addKeyListener(new MyKeyListener(alphaMin));
93
94      double aMaxRange[] = {-1000, 1000};
95          alphaMax = new MyTextField(6, aMaxRange);
96      alphaMax.addKeyListener(new MyKeyListener(alphaMax));
97
98      alpha_range.add(alphaMinLabel);
99      alpha_range.add(alphaMin);
100     alpha_range.add(alphaMaxLabel);
101     alpha_range.add(alphaMax);
102
103     alpha_range.setBorder(
104               BorderFactory.createCompoundBorder(
105                   BorderFactory.createTitledBorder("Winkel A"),
106                   BorderFactory.createEmptyBorder(-20,0,0,0)));
107
108     parameterPanel.add(alpha_range);
109     //end alpha_range
110
111
112     // number_alpha
113     JPanel number_alpha = new JPanel(new FlowLayout(FlowLayout.LEFT,
114                             10, 10));
115     number_alpha.setPreferredSize(new Dimension(150, 150));
116
117         JLabel numAlpha1 = new JLabel("Anzahl (A):");
118
119     double alphaNumRange[] = {-1000, 1000};
120         numAlphaText = new MyTextField(8, alphaNumRange);
121     numAlphaText.addKeyListener(new MyKeyListener(numAlphaText));
122
123     number_alpha.add(numAlpha1);
124     number_alpha.add(numAlphaText);
125
126     number_alpha.setBorder(
127               BorderFactory.createCompoundBorder(
128                   BorderFactory.createTitledBorder("Messpositionen"),
129                   BorderFactory.createEmptyBorder(0,0,0,0)));
130
131     parameterPanel.add(number_alpha);
132     //end number_alpha
133
134     // beta_range
135     JPanel beta_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
136                             10, 30));
137     beta_range.setPreferredSize(new Dimension(150, 150));
138
139     JLabel betaMinLabel = new JLabel("min:");
140     JLabel betaMaxLabel = new JLabel("max:");
141
142
143
144     double bMinRange[] = {-1000, 1000};
145         betaMin = new MyTextField(6, bMinRange);
146     betaMin.addKeyListener(new MyKeyListener(betaMin));
147
148     double bMaxRange[] = {-1000, 1000};
149         betaMax = new MyTextField(6, bMaxRange);
150     betaMax.addKeyListener(new MyKeyListener(betaMax));
151
152
```

```
153      beta_range.add(betaMinLabel);
154      beta_range.add(betaMin);
155      beta_range.add(betaMaxLabel);
156      beta_range.add(betaMax);
157      beta_range.setBorder(
158                BorderFactory.createCompoundBorder(
159                    BorderFactory.createTitledBorder("Winkel B"),
160                    BorderFactory.createEmptyBorder(-20,0,0,0)));
161
162      parameterPanel.add(beta_range);
163      // end beta_range
164
165      // number_beta
166      JPanel number_beta = new JPanel(new FlowLayout(FlowLayout.LEFT,
167                               10, 10));
168      number_beta.setPreferredSize(new Dimension(150, 150));
169
170      JLabel numBeta1 = new JLabel("Anzahl (B):");
171
172      double betaNumRange[] = {-1000, 1000};
173          numBetaText = new MyTextField(8, betaNumRange);
174      numBetaText.addKeyListener(new MyKeyListener(numBetaText));
175
176      number_beta.add(numBeta1);
177      number_beta.add(numBetaText);
178
179      number_beta.setBorder(
180                BorderFactory.createCompoundBorder(
181                    BorderFactory.createTitledBorder("Messpositionen"),
182                    BorderFactory.createEmptyBorder(0,0,0,0)));
183
184      parameterPanel.add(number_beta);
185      //end number_beta
186
187      }//end initParameter()
188
189
190      public void openFile_actionPerformed(ActionEvent e) {
191      fc.setFileFilter(new ScanFilter());
192      int returnVal = fc.showOpenDialog(Dialog_AngleScan.this);
193
194      if (returnVal == JFileChooser.APPROVE_OPTION) {
195          myFile = fc.getSelectedFile();
196          monitoringArea.append( "Opening: " +myFile.getName()
197                     + "." + newline);
198          try {
199          importParameter(myFile);
200          }
201          catch (IOException e1) {
202          System.out.println("import nicht moeglich");
203          }
204      }
205      else {
206          monitoringArea.append(" Open command cancelled by user." + newline);
207      }
208      }// end openFile_actionPerformed
209
210
211      public void saveFile_actionPerformed(ActionEvent e) {
212      if (instrumentSet & setParameter()) {
213          int returnVal = fc.showSaveDialog(Dialog_AngleScan.this);
214          if (returnVal == JFileChooser.APPROVE_OPTION) {
215          File file = fc.getSelectedFile();
216          SphereScan myScan = new SphereScan (file, myTable, 0);
217          myScan.createGrid(center, a_range, b_range, na, nb);
218          monitoringArea.append(" Saving: " + file.getName() + "." + newline);
219          }
220          else {
221          monitoringArea.append(" Save command cancelled by user." + newline);
222          }
223      }
224      else {
225          monitoringArea.append("Instrument waehlen!");
226          return;
227      }
228      }// end saveFile_actionPerformed
```

```
229
230
231     public boolean setParameter() {
232     if (xField.valueSet && yField.valueSet && zField.valueSet && alphaMin.valueSet
233         && alphaMax.valueSet && numAlphaText.valueSet && betaMin.valueSet
234         && betaMax.valueSet && numBetaText.valueSet) {
235         center = new Point3D(Transformation.double2long(xField.myValue),
236                 Transformation.double2long(yField.myValue),
237                 Transformation.double2long(zField.myValue));
238         a_range[1] = Transformation.double2long(alphaMin.myValue);
239     a_range[0] = Transformation.double2long(alphaMax.myValue);
240     b_range[1] = Transformation.double2long(betaMin.myValue);
241     b_range[0] = Transformation.double2long(betaMax.myValue);
242     na = (long) numAlphaText.myValue;
243     nb = (long) numBetaText.myValue;
244     return true;
245     }
246     else {
247         monitoringArea.append("Fixpunkt wurde nicht spezifiziert!\n");
248         return false;
249     }
250     }// end setParameter
251
252     public void importParameter(File file) throws IOException {
253     BufferedReader in = new BufferedReader(new FileReader(file));
254         String input;// = new String;
255         String output;
256     double myDouble;
257     StringTokenizer stringToken;
258     for (int i=0; i<18; i++ ) {
259         input = in.readLine();
260         stringToken = new StringTokenizer(input);
261         String str = new  String(stringToken.nextToken());
262         if (str.equals("$x_Wert:")) {
263         if (stringToken.hasMoreTokens()) {
264             output = stringToken.nextToken();
265             xField.myValue = Double.parseDouble(output)/1000;
266             xField.setText(Double.toString(xField.myValue));
267             xField.valueSet = true;
268         }//end if
269         }// end if
270         if (str.equals("$y_Wert:")) {
271         if (stringToken.hasMoreTokens()) {
272             output = stringToken.nextToken();
273             yField.myValue = Double.parseDouble(output)/1000;
274             yField.setText(Double.toString(yField.myValue));
275             yField.valueSet = true;
276         }// end if
277         }// end if
278
279         if (str.equals("$z_Wert:")) {
280         if (stringToken.hasMoreTokens()) {
281             output = stringToken.nextToken();
282             zField.myValue  = Double.parseDouble(output)/1000;
283             zField.setText(Double.toString(zField.myValue));
284             zField.valueSet= true;
285         }// end if
286         }// end if
287
288         if (str.equals("$a_Min:")) {
289         if (stringToken.hasMoreTokens()) {
290             output = stringToken.nextToken();
291             alphaMin.myValue = Double.parseDouble(output)/1000;
292             alphaMin.setText(Double.toString(alphaMin.myValue));
293             alphaMin.valueSet = true;
294         }
295
296         }
297         if (str.equals("$a_Max:")) {
298         if (stringToken.hasMoreTokens()) {
299             output = stringToken.nextToken();
300             alphaMax.myValue = Double.parseDouble(output)/1000;
301             alphaMax.setText(Double.toString(alphaMax.myValue));
302             alphaMax.valueSet = true;
303         }// end if
304         }// end if
```

```
305
306         if (str.equals("$a_Num:")) {
307         if (stringToken.hasMoreTokens()) {
308             output = stringToken.nextToken();
309             numAlphaText.myValue = Long.parseLong(output);
310             numAlphaText.setText(Double.toString(numAlphaText.myValue));
311             numAlphaText.valueSet = true;
312         }// end if
313         }// end if
314
315         if (str.equals("$b_Min:")) {
316         if (stringToken.hasMoreTokens()) {
317             output = stringToken.nextToken();
318             betaMin.myValue = Double.parseDouble(output)/1000;
319             betaMin.setText(Double.toString(betaMin.myValue));
320             betaMin.valueSet = true;
321         }// end if
322         }// end if
323
324         if (str.equals("$b_Max:")) {
325         if (stringToken.hasMoreTokens()) {
326             output = stringToken.nextToken();
327             betaMax.myValue = Double.parseDouble(output)/1000;
328             betaMax.setText(Double.toString(betaMax.myValue));
329             betaMax.valueSet = true;
330         }// end if
331         }// end if
332
333         if (str.equals("$b_Num:")) {
334         if (stringToken.hasMoreTokens()) {
335             output = stringToken.nextToken();
336             numBetaText.myValue = Long.parseLong(output);
337             numBetaText.setText(Double.toString(numBetaText.myValue));
338             numBetaText.valueSet = true;
339         }// end if
340         }// end if
341
342         if (str.equals("$Instrument:")) {
343         if (stringToken.hasMoreTokens()) {
344             output = stringToken.nextToken();
345             if (output.equals("RTOF")) instrumentList.setSelectedIndex(1);
346             myTable.myInstrument = myTable.RTOF;
347             if (output.equals("DFMS")) instrumentList.setSelectedIndex(2);
348             myTable.myInstrument = myTable.DFMS;
349             if (output.equals("COPS")) instrumentList.setSelectedIndex(3);
350             myTable.myInstrument = myTable.COPS;
351         }// end if
352         }// end if
353     }// end for
354     in.close();
355     }// end importParameter
356
357
358 }// end class Dialog_AngleScan.java
359
```

```java

/**
 * Dialog_PlaneScan1.java
 *
 *
 * Created: Sun Jun  4 20:28:26 2000
 *
 * @author Roland Schaefer
 *
 * Userinterface zur Definition eines Ebenenscans.
 *
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.util.*;


public class Dialog_PlaneScan extends Dialog_Scan {

    // declaration of swing components
    MyTextField yField;
    MyTextField xMin, xMax, numXText;
    MyTextField zMin, zMax, numZText;
    // end declaration of swing components



    //declaration of other components
    private final String newline = "\n";
    long yValue;
    long x_range[]= new long[2];
    long z_range[]= new long[2];
    long nx, nz;
    // end declaration of other components


    // constructor of Frame
    public Dialog_PlaneScan(Table table) {
    super(table, "EbenenScan");
    }// constructor of Frame




    public void initParameter() {
    // fixPoint choice
    JPanel fixPoint_choice = new JPanel(new FlowLayout(FlowLayout.CENTER,
                                10, 60));
    fixPoint_choice.setPreferredSize(new Dimension(150, 200));
    fixPoint_choice.setBorder(
              BorderFactory.createCompoundBorder(
                  BorderFactory.createTitledBorder("FixPunkt"),
                  BorderFactory.createEmptyBorder(-20,0,0,0)));

    JLabel yLabel = new JLabel("y:      ");

    double yRange[] = {-1000, 1000};
        yField = new MyTextField(6, yRange);
    yField.addKeyListener(new MyKeyListener(yField));

    fixPoint_choice.add(yLabel);
    fixPoint_choice.add(yField);
    parameterPanel.add(fixPoint_choice);
    // end fixPoint choice

    //x_range
    JPanel x_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
                              10, 30));
    x_range.setPreferredSize(new Dimension(150, 150));

    JLabel xMinLabel = new JLabel("min:");
    JLabel xMaxLabel = new JLabel("max:");
```

```
77
78      double xMinRange[] = {-1000, 1000};
79          xMin = new MyTextField(6, xMinRange);
80      xMin.addKeyListener(new MyKeyListener(xMin));
81
82      double xMaxRange[] = {-1000, 1000};
83          xMax = new MyTextField(6, xMaxRange);
84      xMax.addKeyListener(new MyKeyListener(xMax));
85
86      x_range.add(xMinLabel);
87      x_range.add(xMin);
88      x_range.add(xMaxLabel);
89      x_range.add(xMax);
90
91      x_range.setBorder(
92                BorderFactory.createCompoundBorder(
93                    BorderFactory.createTitledBorder("Richtung x"),
94                    BorderFactory.createEmptyBorder(-20,0,0,0)));
95
96      parameterPanel.add(x_range);
97      //end x_range
98
99
100     // number_x
101     JPanel number_x = new JPanel(new FlowLayout(FlowLayout.LEFT,
102                               10, 10));
103     number_x.setPreferredSize(new Dimension(150, 150));
104
105         JLabel numX = new JLabel("Anzahl (x):");
106
107     double xNumRange[] = {-1000, 1000};
108         numXText = new MyTextField(8, xNumRange);
109     numXText.addKeyListener(new MyKeyListener(numXText));
110
111     number_x.add(numX);
112     number_x.add(numXText);
113
114     number_x.setBorder(
115               BorderFactory.createCompoundBorder(
116                   BorderFactory.createTitledBorder("Messpositionen"),
117                   BorderFactory.createEmptyBorder(0,0,0,0)));
118
119
120     parameterPanel.add(number_x);
121     //end number_x
122
123     // z_range
124     JPanel z_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
125                               10, 30));
126     z_range.setPreferredSize(new Dimension(150, 150));
127
128     JLabel zMinLabel = new JLabel("min:");
129     JLabel zMaxLabel = new JLabel("max:");
130
131     double bMinRange[] = {-1000, 1000};
132         zMin = new MyTextField(6, bMinRange);
133     zMin.addKeyListener(new MyKeyListener(zMin));
134     double zMaxRange[] = {-1000, 1000};
135         zMax = new MyTextField(6, zMaxRange);
136     zMax.addKeyListener(new MyKeyListener(zMax));
137
138     z_range.add(zMinLabel);
139     z_range.add(zMin);
140     z_range.add(zMaxLabel);
141     z_range.add(zMax);
142     z_range.setBorder(
143               BorderFactory.createCompoundBorder(
144                   BorderFactory.createTitledBorder("Richtung z"),
145                   BorderFactory.createEmptyBorder(-20,0,0,0)));
146
147     parameterPanel.add(z_range);
148     // end z_range
149
150     // number_z
151     JPanel number_z = new JPanel(new FlowLayout(FlowLayout.LEFT,
152                               10, 10));
```

```
153      number_z.setPreferredSize(new Dimension(150, 150));
154
155      JLabel numZ = new JLabel("Anzahl (z):");
156
157      double zNumRange[] = {-1000, 1000};
158          numZText = new MyTextField(8, zNumRange);
159      numZText.addKeyListener(new MyKeyListener(numZText));
160
161      number_z.add(numZ);
162      number_z.add(numZText);
163
164      number_z.setBorder(
165                BorderFactory.createCompoundBorder(
166                    BorderFactory.createTitledBorder("Messpositionen"),
167                    BorderFactory.createEmptyBorder(0,0,0,0)));
168
169      parameterPanel.add(number_z);
170      //end number_z
171
172      }//end initParameter()
173
174
175
176      public void openFile_actionPerformed(ActionEvent e) {
177      fc.setFileFilter(new ScanFilter());
178      int returnVal = fc.showOpenDialog(Dialog_PlaneScan.this);
179
180      if (returnVal == JFileChooser.APPROVE_OPTION) {
181          myFile = fc.getSelectedFile();
182          monitoringArea.append( "Opening: " +myFile.getName()
183                      + "." + newline);
184          try {
185          importParameter(myFile);
186          }
187          catch (IOException e1) {
188          System.out.println("import nicht moeglich");
189          }
190      }// end if
191      else {
192          monitoringArea.append(" Open command cancelled by user." + newline);
193      }// end else
194      }// end openFile_actionPerformed
195
196
197      public void saveFile_actionPerformed(ActionEvent e) {
198      if (instrumentSet & setParameter()) {
199          int returnVal = fc.showSaveDialog(Dialog_PlaneScan.this);
200          if (returnVal == JFileChooser.APPROVE_OPTION) {
201              File file = fc.getSelectedFile();
202              PlaneScan myScan = new PlaneScan (file, myTable);
203              myScan.createGrid(x_range, z_range, nx, nz, yValue);
204              //    System.out.println("file schreiben");
205              //hier file speichern
206              monitoringArea.append(" Saving: " + file.getName() + "." + newline);
207          }// end if
208          else {
209              monitoringArea.append(" Save command cancelled by user." + newline);
210          }// end else
211      }// end if
212      else {
213          monitoringArea.append("Instrument waehlen!" + newline);
214          return;
215      }// end else
216      }// end saveFile_actionPerformed
217
218
219      public boolean setParameter() {
220      if (yField.valueSet && xMin.valueSet
221          && xMax.valueSet && numXText.valueSet && zMin.valueSet
222          && zMax.valueSet && numZText.valueSet) {
223          yValue = Transformation.double2long(yField.myValue);
224
225          x_range[0] = Transformation.double2long(xMin.myValue);
226          x_range[1] = Transformation.double2long(xMax.myValue);
227          z_range[0] = Transformation.double2long(zMin.myValue);
228          z_range[1] = Transformation.double2long(zMax.myValue);
```

```
229
230         nx = (long) numXText.myValue;
231         nz = (long) numZText.myValue;
232         return true;
233     }// end if
234     else {
235         monitoringArea.append("Fixpunkt wurde nicht spezifiziert!\n");
236         return false;
237     }// end else
238     }// end saveFile_actionPerformed
239
240     public void importParameter(File file) throws IOException {
241     BufferedReader in = new BufferedReader(new FileReader(file));
242         String input;
243         String output;
244     double myDouble;
245     StringTokenizer stringToken;
246     for (int i=0; i<18; i++ ) {
247         input = in.readLine();
248         stringToken = new StringTokenizer(input);
249         String str = new  String(stringToken.nextToken());
250
251         if (str.equals("$y_Wert:")) {
252         if (stringToken.hasMoreTokens()) {
253             output = stringToken.nextToken();
254             yField.myValue = Double.parseDouble(output)/1000;
255             yField.setText(Double.toString(yField.myValue));
256             yField.valueSet = true;
257         }// end if
258         }// end if
259
260
261         if (str.equals("$x_Min:")) {
262         if (stringToken.hasMoreTokens()) {
263             output = stringToken.nextToken();
264             xMin.myValue = Double.parseDouble(output)/1000;
265             xMin.setText(Double.toString(xMin.myValue));
266             xMin.valueSet = true;
267         }// end if
268         }// end if
269
270         if (str.equals("$x_Max:")) {
271         if (stringToken.hasMoreTokens()) {
272             output = stringToken.nextToken();
273             xMax.myValue = Double.parseDouble(output)/1000;
274             xMax.setText(Double.toString(xMax.myValue));
275             xMax.valueSet = true;
276         }// end if
277         }// end if
278
279         if (str.equals("$x_Num:")) {
280         if (stringToken.hasMoreTokens()) {
281             output = stringToken.nextToken();
282             numXText.myValue = Long.parseLong(output);
283             numXText.setText(Double.toString(numXText.myValue));
284             numXText.valueSet = true;
285         }// end if
286         }// end if
287
288         if (str.equals("$z_Min:")) {
289         if (stringToken.hasMoreTokens()) {
290             output = stringToken.nextToken();
291             zMin.myValue = Double.parseDouble(output)/1000;
292             zMin.setText(Double.toString(zMin.myValue));
293             zMin.valueSet = true;
294         }// end if
295         }// end if
296
297         if (str.equals("$z_Max:")) {
298         if (stringToken.hasMoreTokens()) {
299             output = stringToken.nextToken();
300             zMax.myValue = Double.parseDouble(output)/1000;
301             zMax.setText(Double.toString(zMax.myValue));
302             zMax.valueSet = true;
303         }// end if
304         }// end if
```

```
305
306         if (str.equals("$z_Num:")) {
307         if (stringToken.hasMoreTokens()) {
308             output = stringToken.nextToken();
309             numZText.myValue = Long.parseLong(output);
310             numZText.setText(Double.toString(numZText.myValue));
311             numZText.valueSet = true;
312         }// end if
313         }// end if
314
315         if (str.equals("$Instrument:")) {
316         if (stringToken.hasMoreTokens()) {
317             output = stringToken.nextToken();
318             if (output.equals("RTOF")) instrumentList.setSelectedIndex(1);
319             myTable.myInstrument = myTable.RTOF;
320             if (output.equals("DFMS")) instrumentList.setSelectedIndex(2);
321             myTable.myInstrument = myTable.DFMS;
322             if (output.equals("COPS")) instrumentList.setSelectedIndex(3);
323             myTable.myInstrument = myTable.COPS;
324         }// end if
325         }// end if
326     }// end for
327     in.close();
328     }// end importParameter
329
330 }// end class Dialog_AngleScan.java
331
```

```java
/**
 * Dialog_Scan.java
 *
 * Created: Sun Jun  4 20:37:07 2000
 *
 * @author Roland Schaefer
 *
 * Abstrakte Klasse als Superklasse f"ur die Userinterfaces der
 * Scandefinitionen.
 */


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.util.*;


public abstract class Dialog_Scan extends JFrame {

    // declaration of swing components
    JMenuBar menuBar1 = new JMenuBar();
    JMenu menuFile = new JMenu();
    JMenuItem menuFileExit = new JMenuItem();
    JMenu menuHelp = new JMenu();
    JMenuItem menuHelpAbout = new JMenuItem();
    FlowLayout mainLayout = new FlowLayout(FlowLayout.LEFT, 30, 30);
    JPanel mainPanel, parameterPanel;
    JTextArea monitoringArea;
    JComboBox instrumentList;
    // end declaration of swing components



    //declaration of other components
    private static String newline = "\n";
    boolean written, instrumentSet;
    Table myTable;
    File myFile;
    Dimension mySize = new Dimension(830, 680);
    String myTitle;
    final JFileChooser fc = new JFileChooser();
    // end declaration of other components

    // constructor of Frame
    public Dialog_Scan(Table table, String title) {
    myTable = table;
    myTitle = title;
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        init();
        initParameter();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
    }// constructor of Frame

    // declaration of abstract methods
    public abstract boolean setParameter();
    public abstract void importParameter(File file) throws IOException;
    public abstract void openFile_actionPerformed(ActionEvent e);
    public abstract void saveFile_actionPerformed(ActionEvent e);
    public abstract void initParameter();
    // end declaration of abstract methods


    //component initialization
    private void init() throws Exception {
    mainPanel =(JPanel) getContentPane();
    mainPanel.setLayout(mainLayout);
    this.setSize(mySize);
    this.setTitle(myTitle);

```

```
77      //Menu
78      menuFile.setText("File");
79      menuFileExit.setText("Exit");
80      menuFileExit.addActionListener(new ActionListener()  {
81          public void actionPerformed(ActionEvent e) {
82              fileExit_actionPerformed(e);
83          }
84          });
85      menuHelp.setText("Help");
86      menuHelpAbout.setText("About");
87      menuHelpAbout.addActionListener(new ActionListener()  {
88          public void actionPerformed(ActionEvent e) {
89              helpAbout_actionPerformed(e);
90          }
91          });
92      menuFile.add(menuFileExit);
93      menuHelp.add(menuHelpAbout);
94      menuBar1.add(menuFile);
95      menuBar1.add(menuHelp);
96      this.setJMenuBar(menuBar1);
97      // end Menu
98
99
100     // panels
101
102     // parameterPanel
103     parameterPanel = new JPanel(new FlowLayout(FlowLayout.LEFT,
104                                20, 20));
105         parameterPanel.setPreferredSize(new Dimension(360, 580));
106     parameterPanel.setBorder(
107               BorderFactory.createCompoundBorder(
108                   BorderFactory.createTitledBorder("ScanParameter"),
109                   BorderFactory.createEmptyBorder(-20,0,0,0)));
110     // end parameterPanel
111
112
113     //instrument_choice
114     JPanel instrument_choice = new JPanel(new FlowLayout(FlowLayout.CENTER,
115                                  30, 90));
116     instrument_choice.setPreferredSize(new Dimension(150, 200));
117     instrument_choice.add(instrument_select());
118
119     instrument_choice.setBorder(
120               BorderFactory.createCompoundBorder(
121                   BorderFactory.createTitledBorder("Instrument"),
122                   BorderFactory.createEmptyBorder(-20,0,0,0)));
123
124     parameterPanel.add(instrument_choice);
125     //end instrument_choice
126
127     // end parameterPanel
128
129
130
131     //infoPanel
132     JPanel infoPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 20));
133     infoPanel.setPreferredSize(new Dimension(380, 580));
134     //end infoPanel
135
136
137
138     // sketchPanel
139     JPanel sketchPanel = new JPanel();
140     sketchPanel.setPreferredSize(new Dimension(340, 210));
141     sketchPanel.setBorder(
142               BorderFactory.createCompoundBorder(
143                   BorderFactory.createTitledBorder("Hilfsskizze"),
144                   BorderFactory.createEmptyBorder(-20,0,0,0)));
145
146     infoPanel.add(sketchPanel);
147     // end sketchPanel
148
149
150     // filePanel
151     fc.setFileFilter(new ScanFilter());
152     JPanel filePanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));
```

```
153     filePanel.setPreferredSize(new Dimension(340, 130));
154
155
156     //Create the open button
157         ImageIcon openIcon = new ImageIcon("images/open.gif");
158         JButton openButton = new JButton("Open Scan", openIcon);
159         openButton.addActionListener(new ActionListener() {
160         public void actionPerformed(ActionEvent e) {
161             openFile_actionPerformed(e);
162         }
163         });
164     filePanel.add(openButton);
165     // end create open button
166
167     //Create the save button
168         ImageIcon saveIcon = new ImageIcon("images/save.gif");
169         JButton saveButton = new JButton("Save Scan", saveIcon);
170         saveButton.addActionListener(new ActionListener() {
171         public void actionPerformed(ActionEvent e){
172             saveFile_actionPerformed(e);
173         }
174         });
175
176     filePanel.add(saveButton);
177     // end create save button
178
179     //Create take-over button
180     JButton take_overButton = new JButton("finished...");
181     take_overButton.addActionListener(new ActionListener() {
182         public void actionPerformed(ActionEvent e) {
183             fileExit_actionPerformed(e);
184         }
185         });
186
187     filePanel.add(take_overButton);
188     // end create take-over button
189
190     filePanel.setBorder(
191               BorderFactory.createCompoundBorder(
192                   BorderFactory.createTitledBorder("File Manipulationen"),
193                   BorderFactory.createEmptyBorder(0,0,0,0)));
194
195     infoPanel.add(filePanel);
196     //end filePanel
197
198     //sysPanel
199     JPanel sysPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 20));
200     sysPanel.setPreferredSize(new Dimension(340, 160));
201
202     //Layout for monitoringArea
203     monitoringArea = new JTextArea();
204     monitoringArea.setEditable(false);
205     JScrollPane areaScrollPane = new JScrollPane(monitoringArea);
206     areaScrollPane.setVerticalScrollBarPolicy(
207                   JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
208     areaScrollPane.setPreferredSize(new Dimension(300, 120));
209     sysPanel.add(areaScrollPane);
210
211     sysPanel.setBorder(
212               BorderFactory.createCompoundBorder(
213                   BorderFactory.createTitledBorder("Systeminfos"),
214                   BorderFactory.createEmptyBorder(-20,0,0,0)));
215
216     infoPanel.add(sysPanel);
217     // end layout for monitoringArea
218     // end sysPanel
219     // end infoPanel
220
221     mainPanel.add(parameterPanel);
222     mainPanel.add(infoPanel);
223     }// end component initialisation
224
225    //instrument_select with combobox
226     public JComboBox instrument_select() {
227     String[] instrumentStrings = {"Instrument", "RTOF", "DFMS", "COPS"};
228         instrumentList = new JComboBox(instrumentStrings);
```

```
229      instrumentList.setSelectedIndex(0);
230      instrumentList.addActionListener(new ActionListener()  {
231          public void actionPerformed(ActionEvent e) {
232              instrumentSet_actionPerformed(e);
233          }// end actionPerformed
234          });// end addActionListener
235      return instrumentList;
236      }//end instrument_select
237
238      //File | Exit action performed
239      public void fileExit_actionPerformed(ActionEvent e) {
240      this.dispose();
241      }// end fileExit_actionPerformed
242
243
244      //Help | About action performed
245      public void helpAbout_actionPerformed(ActionEvent e) {
246      ProtoFrame_AboutBox dlg = new ProtoFrame_AboutBox(this);
247      Dimension dlgSize = dlg.getPreferredSize();
248      Dimension frmSize = getSize();
249      Point loc = getLocation();
250      dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x,
251              (frmSize.height - dlgSize.height) / 2 + loc.y);
252      dlg.setModal(true);
253      dlg.show();
254      }// end helpAbout_actionPerformed
255
256      //Overridden so we can exit when window is closed
257      protected void processWindowEvent(WindowEvent e) {
258      super.processWindowEvent(e);
259      if (e.getID() == WindowEvent.WINDOW_CLOSING) {
260          fileExit_actionPerformed(null);
261      }// end if
262      }// end processWindowEvent
263
264
265      public void instrumentSet_actionPerformed(ActionEvent e) {
266      written = false;
267      JComboBox cb = (JComboBox)e.getSource();
268      int  inst = (int)cb.getSelectedIndex();;
269      if (inst==0) {
270          instrumentSet = false;
271          myTable.myInstrument = myTable.NONE;
272          monitoringArea.append("Kein Instrument gewaehlt\n");
273      }
274      else if (inst==1) {
275          instrumentSet = true;
276          myTable.myInstrument = myTable.RTOF;
277          monitoringArea.append("RTOF gewaehlt\n");
278          cb.transferFocus();
279      }
280      else if (inst==2) {
281          instrumentSet = true;
282          myTable.myInstrument = myTable.DFMS;
283          monitoringArea.append("DFMS gewaehlt\n");
284          cb.transferFocus();
285      }
286      else if (inst==3) {
287          instrumentSet = true;
288          myTable.myInstrument = myTable.COPS;
289          monitoringArea.append("COPS gewaehlt\n");
290          cb.transferFocus();
291      }
292       }// end instrumentSet_actionPerformed
293
294  }// end class Dialog_AngleScan.java
295
296
```

```java
/**
 * Dialog_SphereScan1.java
 *
 * Created: Sun Jun  4 20:37:07 2000
 *
 * @author Roland Schaefer
 *
 * Userinterface zur Definition eines Kugelscans.
 */


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.util.*;


public class Dialog_SphereScan extends Dialog_Scan {

    // declaration of swing components
    MyTextField xField, yField, zField, radiusField;
    MyTextField alphaMin, alphaMax, numAlphaText;
    MyTextField betaMin, betaMax, numBetaText;
    // end declaration of swing components



    //declaration of other components
    private static String newline = "\n";
    Point3D center;
    long a_range[]= new long[2];
    long b_range[]= new long[2];
    long na, nb, radius;
    // end declaration of other components


    // constructor of Frame
    public Dialog_SphereScan(Table table) {
    super(table, "KugelScan");
    }// end constructor of Frame



    public void initParameter() {

    // fixPoint choice
    JPanel fixPoint_choice = new JPanel(new FlowLayout(FlowLayout.CENTER,
                                10, 20));
    fixPoint_choice.setPreferredSize(new Dimension(150, 200));
    fixPoint_choice.setBorder(
              BorderFactory.createCompoundBorder(
                  BorderFactory.createTitledBorder("FixPunkt"),
                  BorderFactory.createEmptyBorder(-20,0,0,0)));

    JLabel xLabel = new JLabel("x:      ");
    JLabel yLabel = new JLabel("y:      ");
    JLabel zLabel = new JLabel("z:      ");
    JLabel radiusLabel = new JLabel("Radius:");

    double xRange[] = {-1000, 1000};
        xField = new MyTextField(6, xRange);
    xField.addKeyListener(new MyKeyListener(xField));

    double yRange[] = {-1000, 1000};
        yField = new MyTextField(6, yRange);
    yField.addKeyListener(new MyKeyListener(yField));

    double zRange[] = {-1000, 1000};
        zField = new MyTextField(6, zRange);
    zField.addKeyListener(new MyKeyListener(zField));

    double radiusRange[] = {-1000, 1000};
        radiusField = new MyTextField(4, radiusRange);
    radiusField.addKeyListener(new MyKeyListener(radiusField));
```

```
77
78      fixPoint_choice.add(xLabel);
79      fixPoint_choice.add(xField);
80      fixPoint_choice.add(yLabel);
81      fixPoint_choice.add(yField);
82      fixPoint_choice.add(zLabel);
83      fixPoint_choice.add(zField);
84      fixPoint_choice.add(radiusLabel);
85      fixPoint_choice.add(radiusField);
86      parameterPanel.add(fixPoint_choice);
87      // end fixPoint_choice
88
89      //alpha_range
90      JPanel alpha_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
91                               10, 30));
92      alpha_range.setPreferredSize(new Dimension(150, 150));
93
94      JLabel alphaMinLabel = new JLabel("min:");
95      JLabel alphaMaxLabel = new JLabel("max:");
96
97      double aMinRange[] = {-1000, 1000};
98          alphaMin = new MyTextField(6, aMinRange);
99      alphaMin.addKeyListener(new MyKeyListener(alphaMin));
100
101     double aMaxRange[] = {-1000, 1000};
102         alphaMax = new MyTextField(6, aMaxRange);
103     alphaMax.addKeyListener(new MyKeyListener(alphaMax));
104
105     alpha_range.add(alphaMinLabel);
106     alpha_range.add(alphaMin);
107     alpha_range.add(alphaMaxLabel);
108     alpha_range.add(alphaMax);
109
110     alpha_range.setBorder(
111               BorderFactory.createCompoundBorder(
112                   BorderFactory.createTitledBorder("Winkel A"),
113                   BorderFactory.createEmptyBorder(-20,0,0,0)));
114
115     parameterPanel.add(alpha_range);
116     //end alpha_range
117
118     // number_alpha
119     JPanel number_alpha = new JPanel(new FlowLayout(FlowLayout.LEFT,
120                               10, 10));
121     number_alpha.setPreferredSize(new Dimension(150, 150));
122
123         JLabel numAlpha1 = new JLabel("Anzahl (A):");
124
125     double alphaNumRange[] = {-1000, 1000};
126         numAlphaText = new MyTextField(8, alphaNumRange);
127     numAlphaText.addKeyListener(new MyKeyListener(numAlphaText));
128
129     number_alpha.add(numAlpha1);
130     number_alpha.add(numAlphaText);
131
132     number_alpha.setBorder(
133               BorderFactory.createCompoundBorder(
134                   BorderFactory.createTitledBorder("Messpositionen"),
135                   BorderFactory.createEmptyBorder(0,0,0,0)));
136
137     parameterPanel.add(number_alpha);
138     //end number_alpha
139
140     // beta_range
141     JPanel beta_range = new JPanel(new FlowLayout(FlowLayout.CENTER,
142                               10, 30));
143     beta_range.setPreferredSize(new Dimension(150, 150));
144
145     JLabel betaMinLabel = new JLabel("min:");
146     JLabel betaMaxLabel = new JLabel("max:");
147
148     double bMinRange[] = {-1000, 1000};
149         betaMin = new MyTextField(6, bMinRange);
150     betaMin.addKeyListener(new MyKeyListener(betaMin));
151     double bMaxRange[] = {-1000, 1000};
152         betaMax = new MyTextField(6, bMaxRange);
```

```
153      betaMax.addKeyListener(new MyKeyListener(betaMax));
154
155      beta_range.add(betaMinLabel);
156      beta_range.add(betaMin);
157      beta_range.add(betaMaxLabel);
158      beta_range.add(betaMax);
159      beta_range.setBorder(
160                BorderFactory.createCompoundBorder(
161                    BorderFactory.createTitledBorder("Winkel B"),
162                    BorderFactory.createEmptyBorder(-20,0,0,0)));
163
164      parameterPanel.add(beta_range);
165      // end beta_range
166
167      // number_beta
168      JPanel number_beta = new JPanel(new FlowLayout(FlowLayout.LEFT,
169                               10, 10));
170      number_beta.setPreferredSize(new Dimension(150, 150));
171
172      JLabel numBeta1 = new JLabel("Anzahl (B):");
173
174      double betaNumRange[] = {-1000, 1000};
175          numBetaText = new MyTextField(8, betaNumRange);
176      numBetaText.addKeyListener(new MyKeyListener(numBetaText));
177
178      number_beta.add(numBeta1);
179      number_beta.add(numBetaText);
180
181      number_beta.setBorder(
182                BorderFactory.createCompoundBorder(
183                    BorderFactory.createTitledBorder("Messpositionen"),
184                    BorderFactory.createEmptyBorder(0,0,0,0)));
185
186      parameterPanel.add(number_beta);
187      //end number_beta
188
189      }//end initParameter()
190
191
192      public void openFile_actionPerformed(ActionEvent e) {
193      fc.setFileFilter(new ScanFilter());
194      int returnVal = fc.showOpenDialog(Dialog_SphereScan.this);
195
196      if (returnVal == JFileChooser.APPROVE_OPTION) {
197          myFile = fc.getSelectedFile();
198          monitoringArea.append( "Opening: " +myFile.getName()
199                      + "." + newline);
200          try {
201          importParameter(myFile);
202          }
203          catch (IOException e1) {
204          System.out.println("import nicht moeglich");
205          }
206      }// end if
207      else {
208          monitoringArea.append(" Open command cancelled by user." + newline);
209      }// end else
210      }// end openFile_actionPerformed
211
212
213      public void saveFile_actionPerformed(ActionEvent e) {
214      if (instrumentSet & setParameter()) {
215          int returnVal = fc.showSaveDialog(Dialog_SphereScan.this);
216          if (returnVal == JFileChooser.APPROVE_OPTION) {
217          File file = fc.getSelectedFile();
218          SphereScan myScan = new SphereScan (file, myTable, radius);
219          myScan.createGrid(center, a_range, b_range, na, nb);
220          monitoringArea.append(" Saving: " + file.getName() + "." + newline);
221          }// end if
222          else {
223          monitoringArea.append(" Save command cancelled by user." + newline);
224          }// end else
225      }// end if
226      else {
227          monitoringArea.append("Instrument waehlen!" + newline);
228          return;
```

```
229      }// end else
230      }// end saveFile_actionPerformed
231
232      public boolean setParameter() {
233      if (xField.valueSet && yField.valueSet && zField.valueSet && radiusField.valueSet
234          && alphaMin.valueSet && alphaMax.valueSet && numAlphaText.valueSet
235          && betaMin.valueSet && betaMax.valueSet && numBetaText.valueSet) {
236          center = new Point3D(Transformation.double2long(xField.myValue),
237                   Transformation.double2long(yField.myValue),
238                   Transformation.double2long(zField.myValue));
239
240          radius = Transformation.double2long(radiusField.myValue);
241          a_range[1] = Transformation.double2long(alphaMin.myValue);
242          a_range[0] = Transformation.double2long(alphaMax.myValue);
243          b_range[1] = Transformation.double2long(betaMin.myValue);
244          b_range[0] = Transformation.double2long(betaMax.myValue);
245
246          na = (long) numAlphaText.myValue;
247          nb = (long) numBetaText.myValue;
248          return true;
249      }// end if
250      else {
251          monitoringArea.append("Fixpunkt wurde nicht spezifiziert!\n");
252          return false;
253      }// end else
254      }// end setParameter
255
256      public void importParameter(File file) throws IOException {
257      BufferedReader in = new BufferedReader(new FileReader(file));
258          String input;
259          String output;
260      double myDouble;
261      StringTokenizer stringToken; // = new StringTokenizer();
262      for (int i=0; i<18; i++ ) {
263          input = in.readLine();
264          stringToken = new StringTokenizer(input);
265          String str = new  String(stringToken.nextToken());
266          if (str.equals("$x_Wert:")) {
267          if (stringToken.hasMoreTokens()) {
268              output = stringToken.nextToken();
269              xField.myValue = Double.parseDouble(output)/1000;
270              xField.setText(Double.toString(xField.myValue));
271              xField.valueSet = true;
272          }// end if
273          }// end if
274
275          if (str.equals("$y_Wert:")) {
276          if (stringToken.hasMoreTokens()) {
277              output = stringToken.nextToken();
278              yField.myValue = Double.parseDouble(output)/1000;
279              yField.setText(Double.toString(yField.myValue));
280              yField.valueSet = true;
281          }// end if
282          }// end if
283
284          if (str.equals("$z_Wert:")) {
285          if (stringToken.hasMoreTokens()) {
286              output = stringToken.nextToken();
287              zField.myValue  = Double.parseDouble(output)/1000;
288              zField.setText(Double.toString(zField.myValue));
289              zField.valueSet= true;
290          }// end if
291          }// end if
292
293          if (str.equals("$Radius:")) {
294          if (stringToken.hasMoreTokens()) {
295              output = stringToken.nextToken();
296              radiusField.myValue  = Double.parseDouble(output)/1000;
297              radiusField.setText(Double.toString(radiusField.myValue));
298              radiusField.valueSet= true;
299          }// end if
300          }// end if
301
302          if (str.equals("$a_Min:")) {
303          if (stringToken.hasMoreTokens()) {
304              output = stringToken.nextToken();
```

```
305             alphaMin.myValue = Double.parseDouble(output)/1000;
306             alphaMin.setText(Double.toString(alphaMin.myValue));
307             alphaMin.valueSet = true;
308         }// end if
309         }// end if
310         if (str.equals("$a_Max:")) {
311         if (stringToken.hasMoreTokens()) {
312             output = stringToken.nextToken();
313             alphaMax.myValue = Double.parseDouble(output)/1000;
314             alphaMax.setText(Double.toString(alphaMax.myValue));
315             alphaMax.valueSet = true;
316         }// end if
317         }// end if
318 
319         if (str.equals("$a_Num:")) {
320         if (stringToken.hasMoreTokens()) {
321             output = stringToken.nextToken();
322             numAlphaText.myValue = Long.parseLong(output);
323             numAlphaText.setText(Double.toString(numAlphaText.myValue));
324             numAlphaText.valueSet = true;
325         }// end if
326         }// end if
327 
328         if (str.equals("$b_Min:")) {
329         if (stringToken.hasMoreTokens()) {
330             output = stringToken.nextToken();
331             betaMin.myValue = Double.parseDouble(output)/1000;
332             betaMin.setText(Double.toString(betaMin.myValue));
333             betaMin.valueSet = true;
334         }// end if
335         }// end if
336 
337         if (str.equals("$b_Max:")) {
338         if (stringToken.hasMoreTokens()) {
339             output = stringToken.nextToken();
340             betaMax.myValue = Double.parseDouble(output)/1000;
341             betaMax.setText(Double.toString(betaMax.myValue));
342             betaMax.valueSet = true;
343         }// end if
344         }// end if
345 
346         if (str.equals("$b_Num:")) {
347         if (stringToken.hasMoreTokens()) {
348             output = stringToken.nextToken();
349             numBetaText.myValue = Long.parseLong(output);
350             numBetaText.setText(Double.toString(numBetaText.myValue));
351             numBetaText.valueSet = true;
352         }// end if
353         }// end if
354 
355         if (str.equals("$Instrument:")) {
356         if (stringToken.hasMoreTokens()) {
357             output = stringToken.nextToken();
358             if (output.equals("RTOF")) instrumentList.setSelectedIndex(1);
359             myTable.myInstrument = myTable.RTOF;
360             if (output.equals("DFMS")) instrumentList.setSelectedIndex(2);
361             myTable.myInstrument = myTable.DFMS;
362             if (output.equals("COPS")) instrumentList.setSelectedIndex(3);
363             myTable.myInstrument = myTable.COPS;
364         }// end if
365         }// end if
366     }// end for
367     in.close();
368     }// end importParameter
369 
370 }// end class Dialog_AngleScan.java
371 
```

```java
/**
 * Dialog_TabelControl.java
 *
 * Created: Tue Oct  10 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Haupt-Userinterface der Applikation.
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.*;
import javax.swing.*;
import javax.swing.text.*;
import javax.swing.event.*;
import java.io.*;


public class Dialog_TableControl extends JFrame  {

    // declaration of swing components
    JMenuBar menuBar1 = new JMenuBar();
    JMenu menuFile = new JMenu();
    JMenuItem menuFileExit = new JMenuItem();
    JMenu menuHelp = new JMenu();
    JMenuItem menuHelpAbout = new JMenuItem();
    FlowLayout mainLayout = new FlowLayout(FlowLayout.LEFT, 30, 30);
    JPanel mainPanel;
    JPanel scanPanel = new JPanel();
    JPanel directPanel = new JPanel();
    JPanel startPanel = new JPanel();
    FlowLayout startLayout = new FlowLayout();
    JPanel infoPanel = new JPanel();
    FlowLayout infoLayout = new FlowLayout();
    JLabel actionLabel_pos;
    JLabel actionLabel_speed;
    JTextField  current_mode;
    MyTextField positionField, speedField;
    JRadioButton absButton, relButton;
    public  JTextArea monitoringArea;
    private JButton startButton;
    File myFile;
    JTextField instrument_info;
    // end declaration of swing components



    //declaration of other components
    Thread sleepThread;
    MyFileThread fileThread;
    private static String newline = "\n";
    boolean axisSet = false;
    boolean written = false;
    boolean instrumentSet = false;
    boolean rel_abs = false;
    boolean pos_abs;
    String axis_code;
    Axis myAxis;
    Table myTable;
    SingleMove mySingleMove = new SingleMove();
    boolean directDefined = false;
    boolean scanDefined = false;
    PosDisplay  z_info, y_info, x_info, a_info, b_info;
    // end declaration of other components


    // constructor of Frame
    public Dialog_TableControl() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        init();
    }
    catch(Exception e) {
```

```
77          e.printStackTrace();
78      }
79      }// end constructor of Frame
80
81      //component initialization
82      private void init() throws Exception {
83      mainPanel =(JPanel) getContentPane();
84      mainPanel.setLayout(mainLayout);
85      this.setSize(new Dimension(800, 600));
86      this.setTitle("TableControl");
87      myTable = new Table();
88      sleepThread = new Thread();
89      sleepThread.setPriority(1);
90
91      //Menu
92      menuFile.setText("File");
93      menuFileExit.setText("Exit");
94      menuFileExit.addActionListener(new ActionListener()  {
95          public void actionPerformed(ActionEvent e) {
96              fileExit_actionPerformed(e);
97          }// end actionPerformed
98          });// end addActionListener
99      menuHelp.setText("Help");
100     menuHelpAbout.setText("About");
101     menuHelpAbout.addActionListener(new ActionListener()  {
102         public void actionPerformed(ActionEvent e) {
103             helpAbout_actionPerformed(e);
104         }// end actionPerformed
105         });//end addActionListener
106
107     menuFile.add(menuFileExit);
108     menuHelp.add(menuHelpAbout);
109     menuBar1.add(menuFile);
110     menuBar1.add(menuHelp);
111     this.setJMenuBar(menuBar1);
112     // end Menu
113
114
115     // panels
116
117     // scanPanel
118         scanPanel.setPreferredSize(new Dimension(200, 200));
119     FlowLayout scanLayout = new FlowLayout(FlowLayout.CENTER, 30, 15);
120         scanPanel.setLayout(scanLayout);
121
122     JButton angleScanButton = new JButton("Winkelscan");
123         angleScanButton.addActionListener(new ActionListener()  {
124             public void actionPerformed(ActionEvent e) {
125                     angleScan_actionPerformed(e);
126             }// end actionPerformed
127             });// end addActionListener
128
129     scanPanel.add(angleScanButton);
130
131     JButton sphereScanButton = new JButton("Kugelscan");
132     sphereScanButton.addActionListener(new ActionListener()  {
133         public void actionPerformed(ActionEvent e) {
134             sphereScan_actionPerformed(e);
135         }// end actionPerformed
136         });// end addActionListener
137
138     scanPanel.add(sphereScanButton);
139
140     JButton planeScanButton = new JButton("Ebenenscan");
141     planeScanButton.addActionListener(new ActionListener()  {
142         public void actionPerformed(ActionEvent e) {
143             planeScan_actionPerformed(e);
144         }// end actionPerformed
145         });// end addActionListener
146
147     scanPanel.add(planeScanButton);
148
149     scanPanel.setBorder(
150               BorderFactory.createCompoundBorder(
151                   BorderFactory.createTitledBorder("Scan Auswahl"),
152                   BorderFactory.createEmptyBorder(-10,5,5,5)));
```

```
153
154
155     //Create the open button
156     final JFileChooser fc = new JFileChooser();
157         ImageIcon openIcon = new ImageIcon("images/open.gif");
158         JButton openButton = new JButton("Open Scan", openIcon);
159         openButton.addActionListener(new ActionListener() {
160         public void actionPerformed(ActionEvent e) {
161             fc.setFileFilter(new ScanFilter());
162             int returnVal = fc.showOpenDialog(Dialog_TableControl.this);
163
164                 if (returnVal == JFileChooser.APPROVE_OPTION) {
165             myFile = fc.getSelectedFile();
166
167             current_mode.setText("Scan: " + myFile.getName());
168             monitoringArea.append( "Opening: " +myFile.getName()
169                       + "." + newline);
170             directDefined = false;
171             scanDefined = true;
172             if (fileThread != null)
173             fileThread.setEnded();
174                 }// end if
175         else {
176                     monitoringArea.append(" Open command cancelled by user." + newline);
177                 }// end else
178         }// end actionPerformed
179         });// end addActionLstener
180
181     scanPanel.add(openButton);
182     //end scanPanel
183
184     //directPanel
185     JPanel choice = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));
186     choice.setPreferredSize(new Dimension(490, 50));
187
188     JPanel position = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 0));
189     position.setPreferredSize(new Dimension(490, 50));
190
191     JPanel speed = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 0));
192     speed.setPreferredSize(new Dimension(490, 50));
193
194     FlowLayout directLayout = new FlowLayout(FlowLayout.CENTER);
195         directPanel.setLayout(directLayout);
196     directPanel.setPreferredSize(new Dimension(500, 200));
197
198     // JLabel positionLabel = new JLabel("        Position:     ");
199         JLabel positionLabel = new JLabel("        Position(mm):     ");
200     positionLabel.setForeground(Color.black);
201
202     // JLabel speedLabel = new JLabel("Geschwindigkeit:");
203         JLabel speedLabel = new JLabel("Geschwindigkeit(mm/s):");
204     speedLabel.setForeground(Color.black);
205
206     //Create text field.
207     double posRange[] = {-1000, 1000};
208         positionField = new MyTextField(12, posRange);
209     positionField.addKeyListener(new MyKeyListener(positionField));
210
211
212     //Create text field
213     double speedRange[] = {0,100};
214     speedField = new MyTextField(12, speedRange);
215     speedField.addKeyListener(new MyKeyListener(speedField));
216
217     // create radio buttons
218     absButton = new JRadioButton("abs", false);
219     absButton.addActionListener(new ActionListener()  {
220         public void actionPerformed(ActionEvent e) {
221             abs_actionPerformed(e);
222         }// end actionPerformed
223         });// end addActionListener
224
225     relButton = new JRadioButton("rel", false);
226     relButton.addActionListener(new ActionListener()  {
227         public void actionPerformed(ActionEvent e) {
228             rel_actionPerformed(e);
```

```
229         }// end actionPerformed
230         });// end addActionListener
231     //end create radio buttons
232
233     // create button
234     JButton takeOverButton = new JButton("Uebernehmen");
235     takeOverButton.addActionListener(new ActionListener()  {
236         public void actionPerformed(ActionEvent e) {
237             takeOver_actionPerformed(e);
238         }// end actionPerformed
239             });// end addActionListener
240
241         // Grouping radio buttons.
242         ButtonGroup abs_rel = new ButtonGroup();
243         abs_rel.add(absButton);
244         abs_rel.add(relButton);
245     // end grouping radio buttons
246
247     choice.add(instrument_select());
248     choice.add(axis_select());
249
250     position.add(positionLabel);
251     position.add(positionField);
252     position.add(absButton);
253     position.add(relButton);
254
255
256     speed.add(speedLabel);
257     speed.add(speedField);
258     speed.add(takeOverButton);
259
260     directPanel.add(choice);
261     directPanel.add(position);
262     directPanel.add(speed);
263     directPanel.setBorder(
264               BorderFactory.createCompoundBorder(
265                   BorderFactory.createTitledBorder("Direkt Steuerung"),
266                   BorderFactory.createEmptyBorder(0,5,5,5)));
267
268     //end directPanel
269
270
271     //startPanel
272     startPanel.setPreferredSize(new Dimension(200, 250));
273     startPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 30, 30));
274     current_mode = new JTextField("*Betriebmodus*");
275     current_mode.setPreferredSize(new Dimension(140, 30));
276
277     // create button
278     startButton = new JButton("START");
279     startButton.addActionListener(new ActionListener()  {
280         public void actionPerformed(ActionEvent e) {
281             try {
282             start_actionPerformed(e);
283             }
284             catch (IOException e1){}
285         }// end actionPerformed
286         });// end addActionListener
287
288     // create button
289     JButton stopButton = new JButton("STOP");
290     stopButton.addActionListener(new ActionListener()  {
291         public void actionPerformed(ActionEvent e) {
292             stop_actionPerformed(e);
293         }// end actionPerformed
294         });// end addActionListener
295
296     startPanel.add(current_mode);
297     startPanel.add(startButton);
298     startPanel.add(stopButton);
299     startPanel.setBorder(
300               BorderFactory.createCompoundBorder(
301                   BorderFactory.createTitledBorder("Steuerung"),
302                   BorderFactory.createEmptyBorder(0,5,5,5)));
303     //end startPanel
304
```

```
305      //infoPanel
306      infoPanel.setPreferredSize(new Dimension(500, 250));
307      infoPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
308
309      JPanel current_values = new JPanel(new FlowLayout(FlowLayout.LEFT,
310                                10, 10));
311      current_values.setPreferredSize(new Dimension(460, 100));
312
313      JLabel instrument_info_Label = new JLabel("Instrument:");
314      current_values.add(instrument_info_Label);
315
316      instrument_info = new JTextField(8);
317      instrument_info.setText("NONE");
318      current_values.add(instrument_info);
319
320      // JLabel z_info_Label = new JLabel("      z-Achse:");
321          JLabel z_info_Label = new JLabel(" z-Achse (um):");
322      current_values.add(z_info_Label);
323
324          z_info = new PosDisplay(myTable.z_axis);
325      current_values.add(z_info);
326      z_info.start();
327
328      // JLabel x_info_Label = new JLabel("    x-Achse:");
329          JLabel x_info_Label = new JLabel("x-Achse (um):");
330      current_values.add(x_info_Label);
331
332          x_info = new PosDisplay(myTable.x_axis);
333      current_values.add(x_info);
334      x_info.start();
335
336      JLabel a_info_Label = new JLabel("      a-Achse:");
337      current_values.add(a_info_Label);
338
339          a_info = new PosDisplay(myTable.a_axis);
340      current_values.add(a_info);
341      a_info.start();
342
343
344      // JLabel y_info_Label = new JLabel("    y-Achse:");
345          JLabel y_info_Label = new JLabel("y-Achse (um):");
346      current_values.add(y_info_Label);
347
348      y_info = new PosDisplay(myTable.y_axis);
349      current_values.add(y_info);
350      y_info.start();
351
352      JLabel b_info_Label = new JLabel("      b-Achse:");
353      current_values.add(b_info_Label);
354
355      b_info = new PosDisplay(myTable.b_axis);
356      current_values.add(b_info);
357      b_info.start();
358
359      initAxisDisplay();
360
361      JLabel sysinfo = new JLabel("SystemInfos");
362      sysinfo.setBorder(BorderFactory.createEmptyBorder(5,0,0,0));
363
364      //Layout for monitotringArea
365      monitoringArea = new JTextArea();
366      monitoringArea.setEditable(false);
367      JScrollPane areaScrollPane = new JScrollPane(monitoringArea);
368      areaScrollPane.setVerticalScrollBarPolicy(
369                  JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
370      areaScrollPane.setPreferredSize(new Dimension(460, 80));
371      //end Layout for monitoringArea
372
373
374      infoPanel.add(current_values);
375      infoPanel.add(sysinfo);
376      infoPanel.add(areaScrollPane);
377      infoPanel.setBorder(
378                BorderFactory.createCompoundBorder(
379                    BorderFactory.createTitledBorder("Informationen"),
380                    BorderFactory.createEmptyBorder(-15,5,5,5)));
```

```
381      //end infoPanel
382      //end Panels
383
384      mainPanel.add(scanPanel);
385      mainPanel.add(directPanel);
386      mainPanel.add(startPanel);
387      mainPanel.add(infoPanel);
388      }// end component initalisation
389
390      //axis_select with combobox
391      public JComboBox axis_select() {
392      String[] axisStrings = {"Achse waehlen", "x-Achse", "y-Achse",
393                  "z-Achse", "a-Achse", "b-Achse"};
394      JComboBox axisList = new JComboBox(axisStrings);
395      axisList.setSelectedIndex(0);
396      axisList.addActionListener(new ActionListener()  {
397          public void actionPerformed(ActionEvent e) {
398              axisSet_actionPerformed(e);
399          }// end actionPerformed
400          });// end addActionListener
401      return axisList;
402      }
403      //end axis_select
404
405      //instrument_select with combobox
406      public JComboBox instrument_select() {
407      String[] instrumentStrings = {"Instrument waehlen", "RTOF", "DFMS", "COPS"};
408      JComboBox instrumentList = new JComboBox(instrumentStrings);
409      instrumentList.setSelectedIndex(0);
410      instrumentList.addActionListener(new ActionListener()  {
411          public void actionPerformed(ActionEvent e) {
412              instrumentSet_actionPerformed(e);
413          }// end actionPerformed
414          });// end addActionListener
415      return instrumentList;
416      }//end instrument_select
417
418
419  // action definitions-------------------------------------------------------------------
420
421
422      //File | Exit action performed
423      public void fileExit_actionPerformed(ActionEvent e) {
424          System.runFinalization();
425      System.exit(0);
426      }
427
428
429      //Help | About action performed
430      public void helpAbout_actionPerformed(ActionEvent e) {
431      ProtoFrame_AboutBox dlg = new ProtoFrame_AboutBox(this);
432      Dimension dlgSize = dlg.getPreferredSize();
433      Dimension frmSize = getSize();
434      Point loc = getLocation();
435      dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x,
436              (frmSize.height - dlgSize.height) / 2 + loc.y);
437      dlg.setModal(true);
438      dlg.show();
439      }
440
441
442      //Overridden so we can exit when window is closed
443      protected void processWindowEvent(WindowEvent e) {
444      super.processWindowEvent(e);
445      if (e.getID() == WindowEvent.WINDOW_CLOSING) {
446          fileExit_actionPerformed(null);
447      }
448      }
449
450
451      public void takeOver_actionPerformed(ActionEvent e) {
452      if (axisSet  && instrumentSet && rel_abs &&
453          positionField.valueSet && speedField.valueSet){
454          //   transferFocus();
455          current_mode.setText(" Direktsteuerung");
456          scanDefined = false;
```

```
457         directDefined = true;
458     }// end if
459     else {
460             current_mode.setText(" *Betriebsmodus*");
461         monitoringArea.append("Angaben fuer direkte Steuerung nicht vollstaendig\n");
462     }// end else
463     }// end takeOver_actionPerformed
464
465     public synchronized void start_actionPerformed(ActionEvent e) throws IOException {
466     if (directDefined) {
467         if (!myAxis.isActive()) {
468         myAxis.setActive();
469         sendMove();
470             while(myAxis.inPos() == false) ;
471         }
472     }// end if
473     else if (scanDefined) {
474         //  System.out.println("scan defined");
475         if (fileThread == null || fileThread.ended()) {
476         fileThread = new MyFileThread(this);
477         fileThread.start();
478         System.out.println("fileThread started");
479         }// end if
480         //   else { // thread already started
481         //  if (fileThread.isInterrupted())
482         //  fileThread.setUninterrupted();
483         //  }// end else
484     }// end else if
485     else {
486         monitoringArea.append("Bitte alle erforderlichen Angaben eingeben!\n");
487     }// end else
488     }// end start_actionPerformed
489
490     public void stop_actionPerformed(ActionEvent e) {
491     if (fileThread != null ){
492         fileThread.setInterrupted();
493         current_mode.setText("*Betriebsmodus*");
494     }
495     if ((myAxis != null)){ //&& (myAxis.isActive())) { // better !myAxis.inPos()
496             myAxis.myTable.amplifierCom.waitForPort();
497         myTable.amplifierCom.writeToCom(myAxis.myId);
498         myTable.amplifierCom.writeToCom("STOP");
499         myTable.amplifierCom.writeToCom("DIS");
500             myAxis.myTable.amplifierCom.unlock();
501         myAxis.setInactive();
502         monitoringArea.append("Bewegung gestoppt!\n");
503     }
504     }// end stop_actionPerformed
505
506     public void angleScan_actionPerformed(ActionEvent e) {
507     Dialog_AngleScan mydialogAngleScan = new Dialog_AngleScan(this.myTable);
508     mydialogAngleScan.show();
509     }// end angelScan_actionPerformed
510
511     public void sphereScan_actionPerformed(ActionEvent e) {
512     Dialog_SphereScan mydialogSphereScan = new Dialog_SphereScan(this.myTable);
513     mydialogSphereScan.show();
514     }// end sphereScan_actionPerformed
515
516     public void planeScan_actionPerformed(ActionEvent e) {
517     Dialog_PlaneScan mydialogPlaneScan = new Dialog_PlaneScan(this.myTable);
518     mydialogPlaneScan.show();
519     }// end planeScan_actionPerformed
520
521     public void abs_actionPerformed(ActionEvent e) {
522     pos_abs = true;
523         rel_abs = true;
524         monitoringArea.append("abs\n");
525     }// end abs_actionPerformed
526
527     public void rel_actionPerformed(ActionEvent e) {
528     pos_abs = false;
529     monitoringArea.append("rel\n");
530         rel_abs = true;
531     }// end rel_actionPerformed
532
```

```
533      public void instrumentSet_actionPerformed(ActionEvent e) {
534      written = false;
535      JComboBox cb = (JComboBox)e.getSource();
536      int  inst = (int)cb.getSelectedIndex();;
537      instrumentSet = (inst != 0);
538      if (inst==0) {
539          myTable.myInstrument = myTable.NONE;
540          monitoringArea.append("Kein Instrument gewaehlt\n");
541      }// end if
542      else if (inst==1) {
543          myTable.myInstrument = myTable.RTOF;
544          monitoringArea.append("RTOF gewaehlt\n");
545      }// end else if
546      else if (inst==2) {
547          myTable.myInstrument = myTable.DFMS;
548          monitoringArea.append("DFMS gewaehlt\n");
549      }// end else if
550      else if (inst==3) {
551          myTable.myInstrument = myTable.COPS;
552          monitoringArea.append("COPS gewaehlt\n");
553      }// end else if
554      instrument_info.setText(myTable.myInstrument.myName);
555      }// end instrumentSet_actionPerformed
556
557      public void axisSet_actionPerformed(ActionEvent e) {
558      written = false;
559      JComboBox cb = (JComboBox)e.getSource();
560      int  axis = (int)cb.getSelectedIndex();;
561      if (axis==0) {
562          axisSet = false;
563      }// end if
564      else if (axis==1) {
565          axisSet = true;
566          myAxis = myTable.x_axis;
567          monitoringArea.append("x-Achse gewaehlt\n");
568      }// end else if
569      else if (axis==2) {
570          axisSet = true;
571          myAxis = myTable.y_axis;
572          monitoringArea.append("y-Achse gewaehlt\n");
573      }// end else if
574      else if (axis==3) {
575          axisSet = true;
576          myAxis = myTable.z_axis;
577          monitoringArea.append("z-Achse gewaehlt\n");
578      }// end else if
579      else if (axis==4) {
580          axisSet = true;
581          myAxis = myTable.a_axis;
582          monitoringArea.append("a-Achse gewaehlt\n");
583      }// end else if
584      else if (axis==5) {
585          axisSet = true;
586          myAxis = myTable.b_axis;
587          monitoringArea.append("b-Achse gewaehlt");
588      }// end else if
589      }// end axisSet_actionPerformed
590
591  //end action definition-----------------------------------------------------------
592
593  //usefull methods-----------------------------------------------------------------
594
595      public void all_disable() {
596      // disables all axis
597          myAxis.myTable.amplifierCom.waitForPort();
598      myTable.x_axis.disable();
599      myTable.y_axis.disable();
600      myTable.z_axis.disable();
601      myTable.a_axis.disable();
602      myTable.b_axis.disable();
603          myAxis.myTable.amplifierCom.unlock();
604      }// end all_disable
605
606      public void sendMove() {
607      if (directDefined) {
608          mySingleMove.setMove(myAxis,
```

```
609                     Transformation.double2long(positionField.myValue),
610                     Transformation.double2long(speedField.myValue),
611                     pos_abs);
612             myAxis.setActive();
613                 // myAxis.enable();
614                 myTable.amplifierCom.waitForPort();
615                 myAxis.disable();
616 
617             try {
618                 sleepThread.sleep(500);
619             } catch (InterruptedException e1) {
620                 System.out.println(e1.toString());
621             }
622 
623             myTable.amplifierCom.writeToCom(mySingleMove.writeToString());
624                 // myAxis.setOrderPos(Transformation.double2long(positionField.myValue));
625                 myAxis.enable();
626 
627             try {
628                  sleepThread.sleep(500);
629             } catch (InterruptedException e1) {
630                  System.out.println(e1.toString());
631                 }
632 
633                 myTable.amplifierCom.writeToCom("MOVE 1");
634                 myTable.amplifierCom.unlock();
635         }// end if
636         else {
637             monitoringArea.append("Angabe fuer Direktsteuerung unvollstaendig!");
638         }// end else
639         }// end sendMove
640 
641 
642         public void initAxisDisplay() {
643         System.out.println("init");
644         myTable.x_axis.init();
645         myTable.y_axis.init();
646         myTable.z_axis.init();
647         myTable.a_axis.init();
648         myTable.b_axis.init();
649         System.out.println("end-init");
650         }
651 
652         public void setActiveAxis(String input) {
653         if (input.charAt(2) == '0') {
654             myAxis = myTable.x_axis;
655         }// end if
656         else if (input.charAt(2) == '2') {
657             myAxis = myTable.y_axis;
658         }// end else if
659         else if (input.charAt(2) == '3') {
660             myAxis = myTable.z_axis;
661         }// end else if
662         else if (input.charAt(2) == '4') {
663             myAxis = myTable.a_axis;
664         }// end else if
665         else if (input.charAt(2) == '5') {
666             myAxis = myTable.b_axis;
667         }
668         }
669 
670         public void setActiveInstrument(char c) {
671         if (c == 'R')
672             myTable.myInstrument = myTable.RTOF;
673         else if (c == 'D')
674             myTable.myInstrument = myTable.DFMS;
675         else
676             myTable.myInstrument = myTable.COPS;
677         }
678 
679     // end usefull methods-----------------------------------------------------------------
680 
681 
682     }// end Dialog_TableControl
683     //*********************************************************************************
684 
```

```java
/**
 * FileChooser.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Ermoeglicht das oeffnen und speichern von Scanfiles via Userinterface.
 *
 */

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.*;

public class FileChooser extends JFrame {
    static private final String newline = "\n";

    public FileChooser() {
        super("FileChooser");

    final JTextArea log = new JTextArea(5,20);
        log.setMargin(new Insets(5,5,5,5));
        log.setEditable(false);
        JScrollPane logScrollPane = new JScrollPane(log);

        //Create a file chooser
        final JFileChooser fc = new JFileChooser();

        //Create the open button
        ImageIcon openIcon = new ImageIcon("images/open.gif");
        JButton openButton = new JButton("Open a File...", openIcon);
        openButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int returnVal = fc.showOpenDialog(FileChooser.this);

                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = fc.getSelectedFile();
            log.append("Opening: " + file.getName() + "." + newline);
                }// end if
        else {
                    log.append("Open command cancelled by user." + newline);
                }// end else
            }// end actionPerformed
        });// end addActionListener

        //Create the save button
        ImageIcon saveIcon = new ImageIcon("images/save.gif");
        JButton saveButton = new JButton("Save a File...", saveIcon);
        saveButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int returnVal = fc.showSaveDialog(FileChooser.this);
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = fc.getSelectedFile();
            log.append("Saving: " + file.getName() + "." + newline);
                }// end if
        else {
                    log.append("Save command cancelled by user." + newline);
                }// end else
            }// end actionPerformed
        });// end addActionListener
    // end creation of save button

        // This Panel is only for the buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(openButton);
        buttonPanel.add(saveButton);


        openButton.setNextFocusableComponent(saveButton);
        saveButton.setNextFocusableComponent(openButton);


        Container contentPane = getContentPane();
```

```
77          contentPane.add(buttonPanel, BorderLayout.NORTH);
78          contentPane.add(logScrollPane, BorderLayout.CENTER);
79      }// end constructor
80
81  }// end class FileChooser
82
```

```java
/**
 * Instrument.java
 *
 * Created: Sun Jun  4 20:25:54 2000
 *
 * @author Roland Schaefer
 *
 * Eine Instanz dieser Klasse ist ein Teil des 5-Achsentisches.
 * Die verschiedenen Instrumente werden in der Klasse `Table' erzeugt.
 *
 */

public class Instrument {
     public String myName;
     public Point3D  myRefPoint;
     public long alphaRef, betaRef;


    public Instrument(String name, long[] ref) {
    //constructor for the different existing instruments (3)
    myName = name;
        myRefPoint = new Point3D(ref[0],ref[1],ref[2]);
    alphaRef = ref[3];
        betaRef = ref[4];
    }

}//End class Instrument

```

```java
/**
 * MyFileThread.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Thread-Klasse, die das abarbeiten von Scanfiles ermoeglicht.
 *
 *///import javax.swing.text.*;

import javax.swing.*;
import java.awt.*;
import java.io.*;


public class MyFileThread  extends Thread {
    public File myFile;
    private boolean interrupted = false;
    private boolean ended = false;
    Thread sleepThread;
    Dialog_TableControl myDialog;
    Table myTable;
    Axis myAxis;


    public MyFileThread  (Dialog_TableControl dialog) {
    setPriority(1);
    sleepThread = new Thread();
    myDialog = dialog;
    myFile = myDialog.myFile;
    myTable = myDialog.myTable;
    this.setPriority(3);
    }// end constructor



    public  void run() {
    try {
        sendScan(myFile);
    }
    catch (IOException e) {
        System.out.println("File konnte nicht geoeffnet werden...");
    }
    ended = true;
    }

    public synchronized void sendScan(File file) throws IOException {
    BufferedReader in = new BufferedReader(new FileReader(file));
    String input;
    int count = 0;
    while((input = in.readLine()) != null) {
        while (interrupted) {
        try {
            wait();
        }
        catch (InterruptedException e) {
            System.out.println(e.toString());
        }
        }
        try {
        sleepThread.sleep(500);
        }
        catch (InterruptedException e2) {
        }
        while (!(myTable.allInPos()));
        if ((input.charAt(0) == '$') && (input.charAt(1) == 'I')) {
        char atpos13 = input.charAt(13);
        myDialog.setActiveInstrument(atpos13);
        myDialog.instrument_info.setText(myTable.myInstrument.myName);
        }// end if
        if ((input.charAt(0) != '#') && (input.charAt(0) != '$')
        && (input.charAt(0) != ';') && !interrupted) {
        myDialog.setActiveAxis(input);
        myDialog.myAxis.setActive();
        myTable.amplifierCom.writeToCom(input);
```

```
77          }// end if
78          else if (input.charAt(0) == ';' && !interrupted) {
79  //      myTable.instrumentsCom.writeToCom(input);
80          }// end else if
81      }// end while
82      in.close();
83      }// end sendScan
84
85
86      public void setInterrupted() {
87      interrupted = true;
88      }
89
90      public synchronized void setUninterrupted() {
91      interrupted = false;
92      notify();
93      }
94
95      public boolean isInterrupted() {
96      return interrupted;
97      }
98
99      public void setEnded() {
100     ended = true;
101     }
102
103     public boolean ended() {
104     return ended;
105     }
106
107 }//End class PosDisplay
108
109
110
111
112
113
114
115
116
117
118
119
120
```

```java
/**
 * MyKeyListener.java
 *
 *
 * Created: Sun Jun  4 20:37:07 2000
 *
 * @author Roland Schaefer
 *
 * Implementation von KeyListener. Zum Abfangen von Fehlerhaften Eingaben
 * in Textfeldern.
 *
 */

import java.awt.event.*;
import javax.swing.*;


class MyKeyListener implements KeyListener {
        MyTextField myTextField;

    public MyKeyListener(MyTextField field) {
    super();
    myTextField = field;
    }


    public void keyReleased(KeyEvent e) {
    try {
        myTextField.myValue = Double.parseDouble(myTextField.getText());
    }
    catch (NumberFormatException e1) {
        if (!myTextField.getText().equals("-"))
        JOptionPane.showMessageDialog(null, "Bitte Zahl eingeben");
        else
        return;
    }
    if ((myTextField.myValue <= myTextField.myRange[1]) &
        (myTextField.myValue >= myTextField.myRange[0])) {
        myTextField.valueSet = true;
    }// end if
    else {
        myTextField.valueSet =  false;
        JOptionPane.showMessageDialog(null,
                        "Bitte Zahl im Bereich eingeben");
    }// end else
    }// end keyReleased

    public void keyPressed(KeyEvent e) {}

    public void keyTyped(KeyEvent e) {}

}// end class MyKeyListener

```

```java
/**
 * MyTextField.java
 *
 * Created: Sun Jun  4 20:37:07 2000
 *
 * @author Roland Schaefer
 *
 * Angepasstes Textfeld, das ueber die Moeglichkeit verfuegt, einen
 * Bereich von zu akzeptierenden Werten anzugeben.
 *
 */

import javax.swing.*;

class MyTextField extends JTextField {
    public boolean valueSet = false;
    public double myValue;
    double myRange[] = new double[2];

    public MyTextField(int lenght, double[] range) {
    super(lenght);
    myRange = range;
    }// end constructor
}// end class MyTextField

```

```java

/**
 * PlaneScan.java
 *
 *
 * Created: Sun Jun 11 11:41:43 2000
 *
 * @author Roland Schaefer
 *
 * Schreibt aus den via Userinterface `Dialog_PlaneScan' eingegebenen
 * Werten den kompletten Scanablauf fuer einen Ebenenscan in ein
 * entsprechendes Scanfile.
 */


import java.io.*;

public class PlaneScan extends Scan {
    String fileName;
    long[] res_x;
    long[] res_z;
    long number_x, number_z;
    long y_value;
    SingleMove myMove = new SingleMove();


    public PlaneScan(File file, Table t) {
    super.init("PlaneScan", file, t);
    }


    public void createGrid(long[] x_range,long[] z_range,
                long nx,long nz,long y) {
    //calculates the x- and y-coordinates of the grid
    //and puts them in a x-vector and a y-vector
    int count = 0;
        long x_start = myTable.beam[0] + x_range[0];
        long z_start = myTable.beam[1] + z_range[0];
    long diff_x = Math.abs( x_range[0] - x_range[1]);
        long diff_z = Math.abs( z_range[0] - z_range[1]);
    y_value = y;
    long res_x;
    long res_z;
        number_x = nx;
        number_z = nz;
    b.println("#ScanParameter: ");
    b.println("#-----------------");
    b.println("$y_Wert: " + y_value);
    b.println("$x_Min: " + x_range[0]);
    b.println("$x_Max: " + x_range[1]);
    b.println("$x_Num: " + number_x);
    b.println("$z_Min: " + z_range[0]);
    b.println("$z_Max: " + z_range[1]);
    b.println("$z_Num: " + number_z);
    b.println("$Instrument: " + myTable.currentInstrument());

    myMove.setMove(myTable.y_axis, y_value, 1000, true);
    b.print(myMove.writeToString());
    for (int i=1; i<nz+1; i++) {
        res_z = z_start - (diff_z/(nz-1) * (i-1));
        myMove.setMove(myTable.z_axis, res_z, 1000, true);
        b.print(myMove.writeToString());
        for (int j=1; j<nx+1; j++) {
        res_x = x_start - (diff_x/(nx-1) * (j-1));
        myMove.setMove(myTable.x_axis, res_x, 1000, true);
        b.print(myMove.writeToString());
        count++;
        b.println("#point " + count + " reached");
        b.println("#please insert instrument orders: ");
        b.println(";");
        b.println(";");
        b.println(";");
        }// end for
    }// end for
    closeFile();
    }// end create Grid
```

```
77
78  }// end class PlaneScan
79
```

```
 1
 2  /*
 3   * Point3D.java
 4   *
 5   *
 6   * Created: Mon Jun  19 15:58:22 2000
 7   *
 8   * @author Roland Schaefer
 9   *
10   * Mit diese Klasse wird ein Punkt im 3 dimensionalen Raum durch eine
11   * Matrix dargestellt.
12   */
13
14  import Jama.*;
15
16
17  public class Point3D {
18      // representation of 3d-point by Matrix
19      public Matrix vector;
20      public Matrix helpfullvector;// nur fuer test public
21      public double[] correction;
22      double offset_z = 1000;
23      public Point3D() {
24      vector = new Matrix(3,1);
25          helpfullvector = new Matrix(3,1);
26      }
27
28      public  Point3D(long x,long y,long z) {
29          double[][] vals = {{(double) x},{(double) y},{(double) z}};
30      vector = new Matrix(vals);
31      helpfullvector = new Matrix(3,1);
32      }
33
34      public  Point3D(double x,double y,double z) {
35          double[][] vals = {{x},{y},{z}};
36      vector = new Matrix(vals);
37      helpfullvector = new Matrix(3,1);
38      }
39      public Point3D(Point3D myPoint) {
40      vector = myPoint.vector.copy();
41      helpfullvector = myPoint.helpfullvector.copy();
42        }
43
44      public static  double[] toArray(Matrix v) {
45      double[] pointarray = new double[3];
46      for (int i=0; i<3; i++)
47      pointarray[i] =  v.get(i,0);
48      return pointarray;
49      }
50
51      public double  getCoord(int a) {
52      return vector.get(a,0);
53      }
54
55      public void setCoord(int a, double value) {
56      vector.set(a,0,(double) value);
57      }
58
59     //  public void translate_rel(long dx, long dy,long dz) {
60  //  //translates the point relative
61  //  //changes entries of vector
62  //      }
63
64  //      public void translate_abs(long x, long y, long z) {
65  //  //translates the point absolut
66  //  Point3D endpoint = new Point3D(x,y,z);
67  //  vector = endpoint.vector;
68  //      }
69
70      public void print(int a, int b) {
71      vector.print(a,b);
72      }
73
74      public void rotate_a(long alpha, long beta,  Point3D actPos,
75                Point3D relRef) {
76      //rotates the point around A
```

```
77       helpfullvector = vector.copy();
78       Point3D ref = new Point3D(relRef);
79       double a = Transformation.deg2rad(alpha);
80       double b = Transformation.deg2rad(beta);
81       double[][] vals_a = {{Math.cos(a),-Math.sin(a),0.},
82                            {Math.sin(a),Math.cos(a),0.},{0.,0.,1.}};
83       Matrix A = new Matrix(vals_a);
84       double[][] vals_b = {{1,0,0},{0, Math.cos(b),-Math.sin(b)},
85                            {0, Math.sin(b),Math.cos(b)}};//ok
86       Matrix B = new Matrix(vals_b);
87       System.out.println("ref:");
88       ref.print(3,3);
89       ref.setCoord(2, ref.getCoord(2) - offset_z);
90       ref.vector = B.times(ref.vector);
91       vector = actPos.vector.plus( A.times(ref.vector));
92       setCoord(2, getCoord(2) + offset_z);
93       helpfullvector = helpfullvector.minus(vector);//endpoint - startpoint
94       System.out.println("correction nach a\n");
95       helpfullvector.print(3,3);
96       correction = toArray(helpfullvector);
97      }// end rotate_a
98
99      public void rotate_b(long alpha, long beta, Point3D actPos,
100              Point3D relRef) {
101     //rotates the point around B
102     helpfullvector = vector.copy();
103     Point3D ref = new Point3D(relRef);
104     double a = Transformation.deg2rad(alpha);
105     double b = Transformation.deg2rad(beta);
106     double[][] vals_a = {{Math.cos(a),-Math.sin(a),0.},
107                          {Math.sin(a),Math.cos(a),0.},{0.,0.,1.}};
108     Matrix A = new Matrix(vals_a);
109     System.out.println("ref:");
110     ref.print(3,3);
111     ref.setCoord(2, ref.getCoord(2) - offset_z);
112     ref.vector = A.times(ref.vector);
113     double[][] vals_b = {{1,0,0},{0, Math.cos(b),-Math.sin(b)},
114                          {0, Math.sin(b),Math.cos(b)}};
115     Matrix B = new Matrix(vals_b);
116
117         vector = actPos.vector.plus(A.times(B).times(A.inverse()).
118                     times(ref.vector));
119         setCoord(2, getCoord(2) + offset_z);
120     helpfullvector = helpfullvector.minus(vector);
121     System.out.println("correction nach b\n");
122         helpfullvector.print(3,3);
123         correction = toArray(helpfullvector);
124     }// end rotate_b
125
126 }// end class Point3D
127
128
```

```java
/**
 * PosDisplay.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Thread-Klasse zur Anzeige der aktuellen Position einer Achse. Fuer
 * jede Achse gibt es einen einzelnen Thread.
 */

import javax.swing.*;
import java.awt.*;

public class PosDisplay  extends JTextField implements Runnable {
    //   private Thread myPosThread;
    private volatile Thread myThread = null;
    //   private boolean displayThreadSuspended = false;
    private boolean isActive = false;
    private Axis  myAxis ;
    public double actPos;
    Thread sleepThread;


    public PosDisplay (Axis a) {
    super(8);
    this.setBackground(Color.white);
        this.setEditable(false);
    myAxis  = a;
    sleepThread = new Thread();
    sleepThread.setPriority(1);
    }// end constructor


    public void start() {
    if(myThread == null) {
        myThread = new Thread(this);
        myThread.setPriority(3);
        myThread.start();
    }// end if
    }// end start()

    public void run() {
    while (true) {
        try {
        myThread.sleep(100);
        }
        catch (InterruptedException e) {
        System.out.println(e.toString());
        }
        myAxis.setPos(this);
    }// end while();
    }// end run



 //   public synchronized void displayPos() {
//        while (!myAxis.isActive()) {
//     try {
//         wait();
//     }
//     catch (InterruptedException e) {
//         System.out.println(e.toString());
//      }
//        }// end while
//        refresh();
//    }// end suspend


   //  public void resume() {
//  myPosThread.resumeMe();
//      if (!isActive)
//          myPosDisplay.notify();
//     }// end resume

```

```
77
78      // public void refresh(double d) {
79  //          this.setText(String.valueOf(d));
80  //  //  this.setText("hallo");
81  //  //  System.out.println("text wurde neu gesetzt");
82  //      }// end refresh
83
84      public void stop() {
85      Thread moribund = myThread;
86      myThread = null;
87      moribund.interrupt();
88      }// end stop
89
90
91
92      public synchronized void setActPos() {
93          myAxis.myTable.amplifierCom.waitForPort();
94      myAxis.myTable.amplifierCom.writeToCom(myAxis.myId);
95      myAxis.myTable.amplifierCom.writeToCom("PFB");
96      actPos = myAxis.myTable.amplifierCom.getFromCom();
97          // myAxis.myTable.amplifierCom.writeToCom(myAxis.myId);
98      // myAxis.myTable.amplifierCom.writeToCom("PFB");
99      // actPos = myAxis.myTable.amplifierCom.getFromCom();
100         myAxis.myTable.amplifierCom.unlock();
101     this.setText(String.valueOf(actPos));
102     }
103
104
105
106 }//End class PosDisplay
107
108
109
110
111
112
113
114
115
116
117
```

```java
/**
 * ProtoFrame_AboutBox.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Dialog-Klasse die Informationen zur Applikation liefert. Wird via
 * `Help'-Feld des Menus gestartet.
 *
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class ProtoFrame_AboutBox extends JDialog implements ActionListener {

    JPanel panel1 = new JPanel();
    JPanel panel2 = new JPanel();
    JPanel insetsPanel1 = new JPanel();
    JPanel insetsPanel2 = new JPanel();
    JPanel insetsPanel3 = new JPanel();
    JButton button1 = new JButton();
    JLabel imageControl1 = new JLabel();
    ImageIcon imageIcon;
    JLabel label1 = new JLabel();
    JLabel label2 = new JLabel();
    JLabel label3 = new JLabel();
    JLabel label4 = new JLabel();
    BorderLayout borderLayout1 = new BorderLayout();
    BorderLayout borderLayout2 = new BorderLayout();
    FlowLayout flowLayout1 = new FlowLayout();
    FlowLayout flowLayout2 = new FlowLayout();
    GridLayout gridLayout1 = new GridLayout();
    String product = "Tablecontroll";
    String version = "version 1.0";
    String copyright = "copyright by r.schaefer";
    String comments = "";

    public ProtoFrame_AboutBox(Frame parent) {
    super(parent);
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
    pack();
    }// end constructor

    private void jbInit() throws Exception  {
    this.setTitle("About");
    setResizable(false);
    panel1.setLayout(borderLayout1);
    panel2.setLayout(borderLayout2);
    insetsPanel1.setLayout(flowLayout1);
    insetsPanel2.setLayout(flowLayout1);
    insetsPanel2.setBorder(new EmptyBorder(10, 10, 10, 10));
    gridLayout1.setRows(4);
    gridLayout1.setColumns(1);
    label1.setText(product);
    label2.setText(version);
    label3.setText(copyright);
    label4.setText(comments);
    insetsPanel3.setLayout(gridLayout1);
    insetsPanel3.setBorder(new EmptyBorder(10, 60, 10, 10));
    button1.setText("Ok");
    button1.addActionListener(this);
    insetsPanel2.add(imageControl1, null);
    panel2.add(insetsPanel2, BorderLayout.WEST);
    this.getContentPane().add(panel1, null);
    insetsPanel3.add(label1, null);
    insetsPanel3.add(label2, null);
```

```
77      insetsPanel3.add(label3, null);
78      insetsPanel3.add(label4, null);
79      panel2.add(insetsPanel3, BorderLayout.CENTER);
80      insetsPanel1.add(button1, null);
81      panel1.add(insetsPanel1, BorderLayout.SOUTH);
82      panel1.add(panel2, BorderLayout.NORTH);
83      }// end jbInit
84
85      protected void processWindowEvent(WindowEvent e) {
86      if (e.getID() == WindowEvent.WINDOW_CLOSING) {
87          cancel();
88      }// end if
89      super.processWindowEvent(e);
90      }// end processWindowEvent
91
92      void cancel() {
93      dispose();
94      }// end cancel
95
96      public void actionPerformed(ActionEvent e) {
97      if (e.getSource() == button1) {
98          cancel();
99      }// end if
100     }// end actionPerformed
101
102 }// end class ProtoFrame_AboutBox
103
104
```

```
 1
 2  /**
 3   * Scan.java
 4   *
 5   *
 6   * Created: Sat Jun 10 22:34:15 2000
 7   *
 8   * @author Roland Schaefer
 9   *
10   * Abstracte Superklasse der Klassen `PlaneScan' und SphereScan'.
11   *
12   */
13
14  import java.io.*;
15  import java.util.*;
16
17  public abstract class Scan {
18      //abstracte class for the other scan classes
19
20      protected  Axis currentAxis;
21      protected  PrintWriter b;
22      protected  File myFile;
23      protected  Table myTable;
24      protected Instrument Instrument;
25
26
27      protected void init(String type,File file, Table t) {
28       myFile = file;
29       myTable = t;
30       Instrument = t.myInstrument;
31       openFile();
32       initializeFile(type);
33      }// end init
34
35
36      protected void openFile() {
37      try {
38          b = new PrintWriter(new FileOutputStream(myFile));
39      }
40      catch (IOException e)
41          {       System.out.println("File kann nicht geoeffnet werden");
42          System.out.println(e.toString());
43          return ;
44          }
45      }// end openFile
46
47
48      protected void initializeFile(String scantype) {
49      //writes the specific orders to the file
50      //each scan consists of a set of moves
51      GregorianCalendar myCalendar = new GregorianCalendar();
52      b.println("#" + scantype + " " + myFile.getName()
53            + "   saved: " + myCalendar.getTime());
54      b.println("#*********************************************************");
55      b.println("#");
56      b.println("#");
57      }// end initializeFile
58
59
60      protected void closeFile() {
61      b.println("#end of " + myFile.getName());
62      b.close();
63      }// end closeFile
64
65  }//End class Scan
66
```

```java
/**
 * ScanFilter.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Erweiterung der Klasse FileFilter. Dient als Filter fuer Files, deren
 * Extension in der Klasse Utils angegeben sind. (Hier bisher nur .scan).
 *
 */
import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;

public class ScanFilter extends FileFilter {

    // Accept all directories and all .scan files
    public boolean accept(File f) {
        if (f.isDirectory())
            return true;
    String extension = Utils.getExtension(f);
        if (extension != null) {
            if (extension.equals(Utils.scan))
        return true;
        else
                return false;
    }// end if
    return false;
    }// end accept

    // The description of this filter
    public String getDescription() {
        return "ScanFiles";
    }// end getDescription

}// end class ScanFilter

```

```java
/**
 * SingleMove.java
 *
 *
 * Created: Sun Jun  4 20:27:19 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Klasse zur Definition einzelner Fahrbefehle.
 */

public class SingleMove {
    private long  absPos;
    private long  speed;
    Axis myAxis;

    // boolean abs true if abs pos is given, else false


    public SingleMove() {
    speed = 0;
    }// end constructor


    public String writeToString() {
    //writes the order to the String
    // String orderString = new String(myAxis.myId + "\n"
    //              + "ORDER 1 " + ""+absPos+" "+speed+" "
    //              + "8192 1000 1000 500 500 0 0\n"
    //              + "MOVE 1\n");

        String orderString = new String( "ORDER 1 " + ""+absPos+" "+speed+" "
                    + "8192 1000 1000 500 500 0 0");
    return orderString;
    }// end write to String


    public void setMove(Axis a, long pos, long sp, boolean abs) {
    // boolean abs true if abs pos is given, else false
    speed = sp;
        myAxis = a;
    if (abs)
        absPos = pos;
        else
         // absPos = pos + Transformation.double2long(myAxis.getPos());
        absPos = pos + (long) myAxis.getPos();

    myAxis.setOrderPos(absPos);

    }// end setMove

}//End class SingleMove

```

```java
/**
 * SphereScan.java
 * AngleScan is a specialcase with radius=0
 *
 * Created: Sun Jun  4 20:26:58 2000
 *
 * @author Roland Schaefer
 *
 * Schreibt aus den via Userinterface `Dialog_SphereScan' oder
 * `Dialog_AngleScan'  eingegebenen Werten den kompletten Scanablauf
 * fuer einen Kugel- oder Winkelscan in ein entsprechendes Scanfile.
 *
 */

import java.lang.Math.*;
import Jama.*;
import java.io.*;


public class SphereScan extends Scan {
    String fileName;
    Point3D myCenter;//[x,y,z]
    long number_a, number_b, myRadius;
    long res_a;
    long res_b;//[min,max] relative to beam
    Instrument myInstrument;
    Point3D actRefPoint, refPointPos, relRefPoint;
    SingleMove myMove = new SingleMove();

  public SphereScan(File file, Table t, long radius) {
    //constructor
      if (radius>0) {
      super.init("Spherescan", file, t);
      }// end if
      else if (radius==0) {
      super.init("AngleScan", file, t);
      }// end else if
      else
      return;

      relRefPoint = new Point3D(t.myInstrument.myRefPoint);
      actRefPoint = new Point3D();
      actRefPoint.vector = relRefPoint.vector.plus(myTable.getCoord().vector);
      refPointPos = new Point3D(actRefPoint);
      actRefPoint.vector.print(3,1);
      myRadius = radius;
  }


    public void createGrid(Point3D center,long a_range[],
               long b_range[],long na, long nb) {
    //calculates the a- and b-coordinates of the grid
        //and puts them into a-vektor and b-vektor
    //beam is the reference-axis

    int count = 0;
        long a_start = a_range[0]; // + myInstrument.alphaRef;
        long b_start = b_range[0]; // + myInstrument.betaRef;
    long diff_a =Math.abs( a_range[0] - a_range[1]);
        long diff_b =Math.abs( b_range[0] - b_range[1]);
    myCenter = new Point3D(center);
    number_a = na;
        number_b = nb;

    //file initialisation
    b.println("#ScanParameter: ");
    b.println("#-----------------");
    b.println("$x_Wert: " + myCenter.getCoord(0));
    b.println("$y_Wert: " + myCenter.getCoord(1));
    b.println("$z_Wert: " + myCenter.getCoord(2));
    b.println("$Radius: " + myRadius);
    b.println("$a_Min: " + a_range[0]);
    b.println("$a_Max: " + a_range[1]);
    b.println("$a_Num: " + number_a);
    b.println("$b_Min: " + b_range[0]);
    b.println("$b_Max: " + b_range[1]);
```

```
77      b.println("$b_Num: " + number_b);
78      b.println("$Instrument: " + myTable.currentInstrument());
79      //end file initialisation
80
81      for (int i=1; i<na+1; i++) {
82          // rotation around a
83          b.println("#rotation around axis a");
84          res_a = a_start - (diff_a/(na-1) * (i-1));
85          res_b = Transformation.double2long(myTable.b_axis.getPos());// + myInstrument.betaRe
86
87          //test if angle is too big
88          if (res_a > 5000 ) {
89          b.println("#Winkel wird in Teilschritte zerlegt!");
90          long help = (long)(res_a / 5 + 0.5);
91          for (int k=1; k< help; k++) {
92              long dres_a = (long) (res_a/help + 0.5);
93              rot_a(dres_a, res_b);
94          }// end for
95          rot_a(res_a, res_b);
96          b.println("#Ende der Winkelunterteilung");
97          } // end if
98          else {
99          b.println("#Winkel ist klein genug!");
100         rot_a(res_a, res_b);
101         actRefPoint.print(3,3);
102         }// end else
103
104         for (int j=1; j<nb+1; j++) {
105         //rotation around b
106         res_b = b_start - (diff_b/(nb-1) * (j-1));
107         res_a = Transformation.double2long(myTable.b_axis.getPos());
108         b.println("#rotation around axis b");
109         //test if angle too big
110         if (res_b > 5000 ) {
111             b.println("#Winkel wird in Teilschritte zerlegt!");
112             long help_b = (long)(res_b / 5 + 0.5);
113             for (int l=1; l< help_b; l++) {
114             long dres_b = (long) (res_b/help_b + 0.5);
115             rot_b(res_a, dres_b);
116             }// end for
117             rot_b(res_a, res_b);
118             b.println("#Ende der Winkelunterteilung");
119         }// end if
120         else {
121             b.println("#Winkel ist klein genug!");
122             rot_b(res_a, res_b);
123         }// end else
124
125         // berechnen der Punkte auf der Kugeloberflaeche
126         double x, y, z;
127         double res_a_rad = Transformation.deg2rad(res_a);
128         double res_b_rad = Transformation.deg2rad(res_b);
129         x =  (myCenter.getCoord(0)/1000 - myRadius
130               * Math.cos(res_b_rad) * Math.sin(res_a_rad));
131         y =  (myCenter.getCoord(1)/1000 + myRadius
132               * Math.cos(res_b_rad) * Math.cos(res_a_rad));
133         z =  (myCenter.getCoord(2)/1000 + myRadius
134               * Math.sin(res_b_rad));
135         // end berechnung der punkte auf der Kugeloberflaeche
136
137         // final move to scanpoint
138         if (!(myRadius == 0) | (count==0)) {
139             b.println("#final move");
140             myMove.setMove(myTable.x_axis,
141                    Transformation.double2long(x
142                    - refPointPos.getCoord(0)), 1000, false);
143             b.print(myMove.writeToString());
144             myMove.setMove(myTable.y_axis,
145                    Transformation.double2long(y
146                    - refPointPos.getCoord(1)), 1000, false);
147             b.print(myMove.writeToString());
148             myMove.setMove(myTable.z_axis,
149                    Transformation.double2long(z
150                    - refPointPos.getCoord(2)), 1000, false);
151             b.print(myMove.writeToString());
152         }// end if
```

```
153         // end final move to scanpoint
154
155         // for instrument orders
156         count++;
157         b.println("#point " + count + " reached");
158         b.println("#please insert instrument orders: ");
159         b.println(";");
160         b.println(";");
161         b.println(";");
162         // end for instrument orders
163         }// end for rotate around b
164     }// end for rotate around a
165     closeFile();
166     }// end createCrid
167
168     private void rot_a(long alpha, long beta) {
169     actRefPoint.rotate_a(alpha, Transformation.double2long(myTable.b_axis.getPos())
170                 , myTable.getCoord(), relRefPoint);
171     myMove.setMove(myTable.a_axis, alpha, 1000, true);
172     b.print(myMove.writeToString());
173
174     // to move the reference point back to its initial position
175     myMove.setMove(myTable.x_axis,
176                 Transformation.double2long(actRefPoint.correction[0]),
177                 1000, false);
178     b.print(myMove.writeToString());
179     myMove.setMove(myTable.y_axis,
180                 Transformation.double2long(actRefPoint.correction[1]),
181                 1000, false);
182     b.print(myMove.writeToString());
183     myMove.setMove(myTable.z_axis,
184                 Transformation.double2long(actRefPoint.correction[2]),
185                 1000, false);
186     b.print(myMove.writeToString());
187     }// end rot_a
188
189
190     private void rot_b(long alpha, long beta) {
191     actRefPoint.rotate_b(Transformation.double2long(myTable.a_axis.getPos())
192                 , beta, myTable.getCoord(), relRefPoint);
193     myMove.setMove(myTable.b_axis, beta, 1000, true);
194     b.print(myMove.writeToString());
195
196     // to move the reference point back to its initial position
197     myMove.setMove(myTable.x_axis,
198                 Transformation.double2long(actRefPoint.correction[0]),
199                 1000, false);
200     b.print(myMove.writeToString());
201     myMove.setMove(myTable.y_axis,
202                 Transformation.double2long(actRefPoint.correction[1]),
203                 1000, false);
204     b.print(myMove.writeToString());
205     myMove.setMove(myTable.z_axis,
206                 Transformation.double2long(actRefPoint.correction[2]),
207                 1000, false);
208     b.print(myMove.writeToString());
209     }// end rot_b
210
211
212 }//End class SphereScan
213
```

```java
/**
 * Table.java
 *
 *
 * Created: Sun Jun  4 20:23:34 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Implementation des 5-Achsentisches mit den Definitionen der Achsen und
 * der Instrumente.
 */

public class Table {
    //Implemetation of real existing Positioning_Table
    Axis x_axis, y_axis, z_axis, a_axis, b_axis;// activeAxis;
    private final  long offset_z = 1000;
    Instrument RTOF, DFMS, COPS, NONE, myInstrument;
    long[] beam = {0,0};//[x,z]
    ComClass amplifierCom;
//    ComClass instrumentsCom;


    public Table() {
    //constructor
    x_axis = new Axis("\\ 0", this);
    y_axis = new Axis("\\ 1", this);
    z_axis = new Axis("\\ 2", this);
    a_axis = new Axis("\\ 3", this);
    b_axis = new Axis("\\ 4", this);
    long[] RTOF_ref = {50, 100, 1150, 0, 0};// von ursprung des tisches aus
    long[] DFMS_ref = {0, -200,1000, 0, 0};
    long[] COPS_ref = {50, 100, 1150, 0, 0};
    long[] NONE_ref = {0,0,0,0,0};
    RTOF = new Instrument("RTOF", RTOF_ref);
    DFMS = new Instrument("DFMS", DFMS_ref);
    COPS = new Instrument("COPS", COPS_ref);
    NONE = new Instrument("NONE",NONE_ref);
    myInstrument = NONE;
    amplifierCom = new ComClass("com1", 9600);
//      instrumentsCom = new ComClass("com2", 9600);
    }// end constructor

    public String currentInstrument() {
    //returns currently mounted instrument
        return myInstrument.myName;
    }// end currentInstrument

    public boolean allInPos() {
    if (x_axis.inPos() && y_axis.inPos()
        && z_axis.inPos() && a_axis.inPos()
        && b_axis.inPos())
        return true;
    else
        return false;
    }// end allInPos

    public Point3D getCoord() {
    Point3D myCoord = new Point3D(x_axis.myPos,y_axis.myPos,z_axis.myPos);
    return myCoord;
    }//end getCoord

}// End class Table
```

```java
/**
 * TableControl.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 * @author Stephan Graf
 *
 * Hauptklasse mit der Methode `main'.
 *
 */

import javax.swing.UIManager;
import java.awt.*;

public class TableControl {

    boolean packFrame = false;

    //Constructor
    public TableControl() {
    Dialog_TableControl frame = new Dialog_TableControl();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame) {
        frame.pack();
    }// end if
    else {
        frame.validate();
    }// end else

    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2,
              (screenSize.height - frameSize.height) / 2);
    frame.setVisible(true);
    // frame.all_disable();
    }// end constructor

    public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
        e.printStackTrace();
    }
    new TableControl();
    }// end main

}// end class TableControl


```

```java
/**
 * Transformation.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Hilfsklasse mit Methoden fuer Umrechnungen.
 */

import Jama.*;
import java.lang.Math.*;


public class Transformation {

    public static double deg2rad(long a) {
    //angle in mikrograd
    double res = ((double)a/180000)*Math.PI;
        return res;
    }// end deg2rad



    public static long double2long(double a) {
    //inkl faktor 1000
    if (a<0) {
        return (long)(1000*a-0.5);
    }// end if
        else {
        return (long) (1000*a+0.5);
    }// end else
    }// end double2long


}//End class Translation

```

```java
/**
 * Utils.java
 *
 * Created: Sat Jun  3 15:58:22 2000
 *
 * @author Roland Schaefer
 *
 * Hilfsklasse zum Herausfinden der Extension von Files.
 *
 */

import java.io.File;

public class Utils {

    public final static String scan = "scan";

    public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');
        if (i > 0 &&  i < s.length() - 1) {
            ext = s.substring(i+1).toLowerCase();
        }// end if
        return ext;
    }// end getExtension

}// end class Utils


```

B Mathematica

Berechnungen fuer die Applikation "TableControl"

```
Off[General::spell, General::spell1]
```

Rotationsmatrik fuer Drehungen um A:

$$\text{rotz}[\alpha\_] := \begin{pmatrix} \text{Cos}[\alpha] & -\text{Sin}[\alpha] & 0 \\ \text{Sin}[\alpha] & \text{Cos}[\alpha] & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
a = rotz[π / 2].{1, 0, 0}
```

```
{0, 1, 0}
```

Hilfsmatrik fuer Drehung um B:

$$\text{rotx}[\beta\_] := \begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}[\beta] & -\text{Sin}[\beta] \\ 0 & \text{Sin}[\beta] & \text{Cos}[\beta] \end{pmatrix}$$

```
c = rotx[π / 4].{0., 0., 141.}
```

```
{0., -99.7021, 99.7021}
```

Rotationsmatrik fuer Drehungen um B:

```
Rotmat[α_, β_] = rotz[α].rotx[β].Inverse[rotz[α]] // Simplify
```

```
{{Cos[α]^2 + Cos[β] Sin[α]^2, -Cos[α] (-1 + Cos[β]) Sin[α], Sin[α] Sin[β]},
 {-Cos[α] (-1 + Cos[β]) Sin[α], Cos[α]^2 Cos[β] + Sin[α]^2, -Cos[α] Sin[β]},
 {-Sin[α] Sin[β], Cos[α] Sin[β], Cos[β]}}
```

```
d = Rotmat[π / 2, π / 4].{100., 100., 100}
```

```
{141.421, 100., 0.}
```

```
spherepoint[radius_, center_, α_, β_] := {center[[1]] - radius Cos[β] Sin[α],
  center[[2]] + radius Cos[β] Cos[α], center[[3]] + radius Sin[β]}


actrefpointα[α_, actpos_, refpoint_] := actpos + rotz[α].refpoint

actrefpointβ[α_, β_, actpos_, refpoint_] := actpos + Rotmat[α, β].refpoint
```

■

Kontrolle anhand der Berechnung von verschiedenen Korrekturen und Moves

Parameter:

```
α := π / 2

β := π / 2

center := {0., -300., 1500.}

radius := 0.
```

```
offsetz := 1000
```

Punkt auf der Kugeloberflaeche:

```
scanpoint = spherepoint[radius, center, α, β]
```

```
{0., -300., 1500.}
```

Referenzpunkt des Instruments vom Schnittpunkt der Drehachsen aus:

```
relref := {50., 100., 1150.}
```

Aktuelle Position der 3 linearen Achsen:

```
actpos := {0., 0., 0.}
```

Aktueller Referenzpunkt (vor den Drehungen):

```
actrefpointbefore = actpos + relref
```

```
{50., 100., 1150.}
```

Referenzpunkt nach der Drehung um α:

```
actrefpoint = actrefpointα[α, actpos, relref]
```

```
{-100., 50., 1150.}
```

Korrektur, um den Referenzpunkt wieder auf seinen urspruenglichen Wert zu bringen:

```
correction1 = actrefpointbefore - actrefpoint
```

```
{150., 50., 0.}
```

```
(*actpos +=correction1*)
```

```
(*actrefpoint+=correction1*)
```

```
relref = rotz[α].relref
```

```
{-100., 50., 1150.}
```

Referenzpunkt nach der Drehung um β:

```
relrefβ = {relref[[1]] , relref[[2]], relref[[3]] - offsetz}
```

```
{-100., 50., 150.}
```

```
a = actrefpointβ[α, β, actpos , relrefβ]
```

```
{150., 50., 100.}
```

```
actrefpoint = {a[[1]], a[[2]], a[[3]] + offsetz}
```

```
{150., 50., 1100.}
```

Korretur, um Referenzpunkt wieder auf seine urspruengliche Position zu bringen:

```
correction2 = actrefpointbefore - actrefpoint
```

```
{-100., 50., 50.}
```

(*actpos += correction2*)

Abschliessende Bewegung um den Referenzpunkt auf den Scanpunkt zu bringen:

```
finalmove = scanpoint - actrefpointbefore
```

```
{-50., -400., 350.}
```

C Screenshots

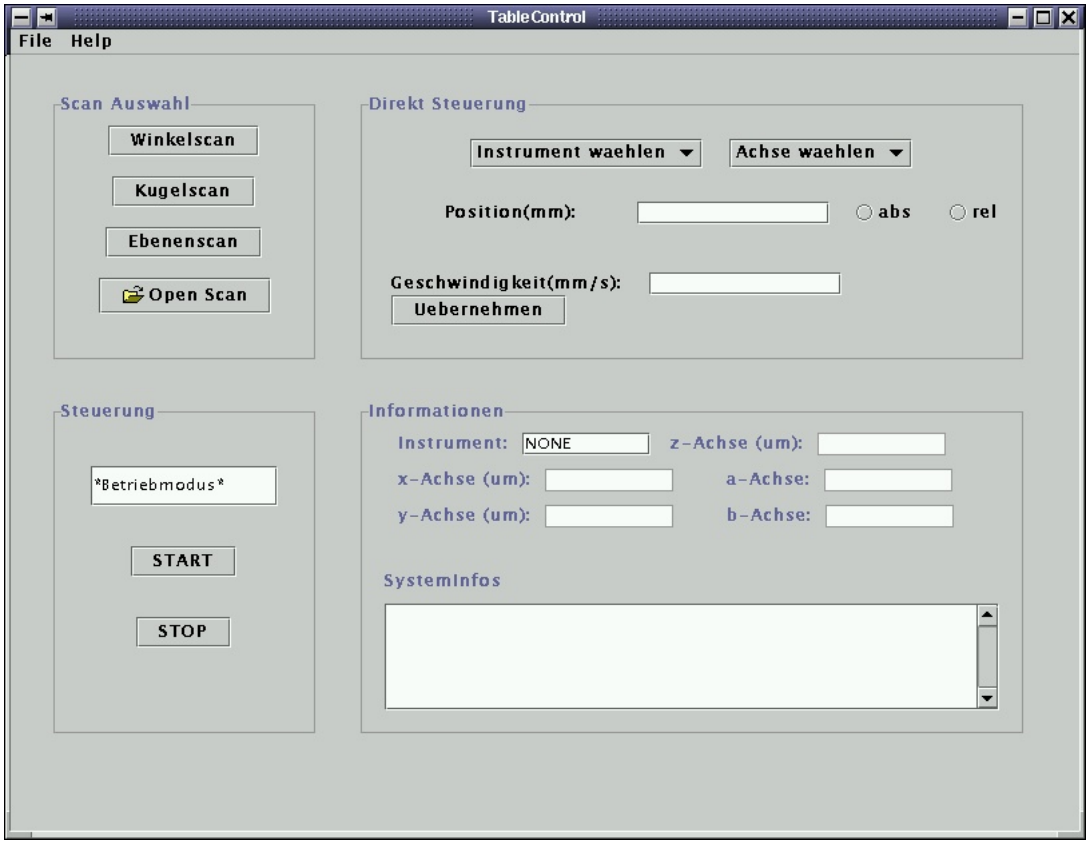

Figure 1: Screenshot des Hauptfensters

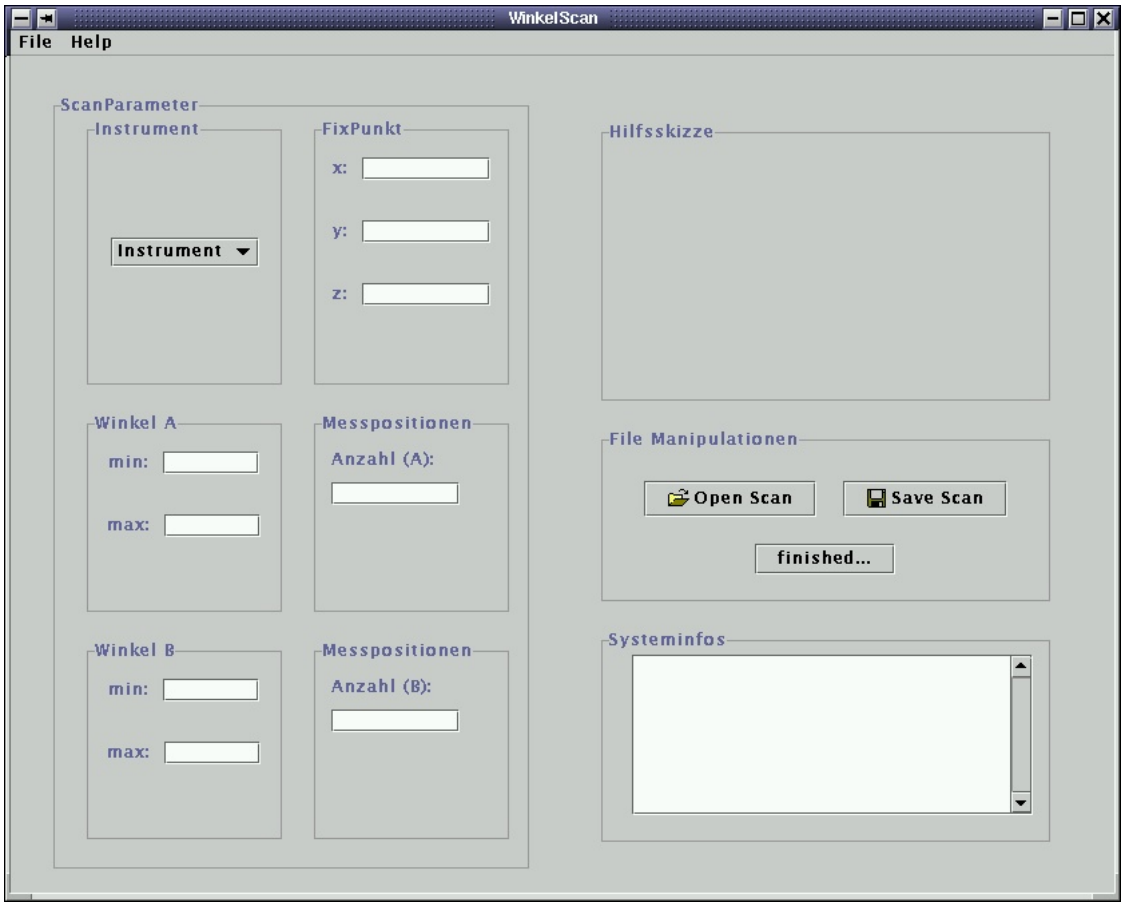

Figure 2: Screenshot des Fensters zur Eingabe eines scans.

D Scanfile

```
#AngleScan winkel2.scan   saved: Tue Jul 17 09:13:45 CEST 2001
#*********************************************************
#
#
#ScanParameter:
#-----------------
$x_Wert: 1000.0
$y_Wert: 2000.0
$z_Wert: 3000.0
$Radius: 0
$a_Min: 4000
$a_Max: 1000
$a_Num: 2
$b_Min: 5000
$b_Max: 1000
$b_Num: 3
$Instrument: COPS
#rotation around axis a
#Winkel ist klein genug!
\ 4
ORDER 1 4000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 7097 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -3244 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 0 1000 8192 1000 1000 500 500 0 0
MOVE1
#rotation around axis b
#Winkel ist klein genug!
\ 5
ORDER 1 5000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 -7097 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 16698 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 -8145 1000 8192 1000 1000 500 500 0 0
MOVE1
#final move
\ 0
ORDER 1 -49000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -98000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 -1147000 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 1 reached
#please insert instrument orders:
;
;
;
#rotation around axis b
#Winkel ist klein genug!
\ 5
ORDER 1 3000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 0 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -5466 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 3117 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 2 reached
#please insert instrument orders:
```

```
;
;
;
#rotation around axis b
#Winkel ist klein genug!
\ 5
ORDER 1 1000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 0 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -5354 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 3306 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 3 reached
#please insert instrument orders:
;
;
;
#rotation around axis a
#Winkel ist klein genug!
\ 4
ORDER 1 1000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 1753 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -3490 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 1722 1000 8192 1000 1000 500 500 0 0
MOVE1
#rotation around axis b
#Winkel ist klein genug!
\ 5
ORDER 1 5000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 -1753 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 14311 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 -8145 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 4 reached
#please insert instrument orders:
;
;
;
#rotation around axis b
#Winkel ist klein genug!
\ 5
ORDER 1 3000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 0 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -5466 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 3117 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 5 reached
#please insert instrument orders:
;
;
;
#rotation around axis b
#Winkel ist klein genug!
```

```
\ 5
ORDER 1 1000 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 0
ORDER 1 0 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 2
ORDER 1 -5354 1000 8192 1000 1000 500 500 0 0
MOVE1
\ 3
ORDER 1 3306 1000 8192 1000 1000 500 500 0 0
MOVE1
#point 6 reached
#please insert instrument orders:
;
;
;
#end of winkel2.scan

```

Literatur

[1] *Inbetriebnahme-Software SR600.EXE für SERVOSTAR 600*, 08 1999.

[2] *Lageregelung/Fahrsatzverwaltung*, 09 1999.