Programmierpraktikum
im Hauptstudium

DISPO2

Graphische Darstellung der Indexräume vernesteter Schleifen

Alexander Wüst

02.11.1995

Universität Passau
Fakultät für Mathematik und Informatik

Inhaltsverzeichnis :

1 Einleitung 3

2 DISPO2: Files 3
2.1 C++ 4
2.1.1 index_struct.h 4
2.1.2 dispo2.cc 5
2.1.3 convert.cc 5
2.1.4 index_func.cc 5
2.1.5 canvas.cc 5
2.1.6 deps.cc 5
2.1.7 groups.cc 5
2.1.8 aux.cc 5
2.1.9 dispo2_win.cc 6
2.1.10 convert_win.cc 6
2.2 Tcl/Tk 6
2.2.1 Dispo2.tcl 6
2.2.2 Convert.tcl 6

3 3D - Ansicht 6
3.1 Indexpunkte 7
3.2 Abhängigkeiten 7

4 Transformationen 8
4.1 Verschiebung 9
4.2 Vergrößerung/Verkleinerung 9
4.3 Rotation 9
4.4 Komplexität 9

5 Erweiterung und Optimierung 10

6 Zusammenfassung 10

Literaturangabe 11

1 Einleitung

DISPO2 ('DIsplay System for POlyhedra 2') ist ein in das Projekt LooPo ('Loop Parallelisation in the Polytope Model') eingebettetes Programm zur Visualisierung des Indexraumes perfekt vernesteter Schleifen bis zur Dimension 3 und der Abhängigkeiten zwischen den Indexpunkten. Außerdem werden zur Betrachtung des Indexraumes zahlreiche Funktionen wie Zoom und Rotation zur Verfügung gestellt.
DISPO2 ist in der Programmiersprache 'C++' programmiert und besitzt ein Frontend in Tcl/Tk. Die Tcl/Tk-Fenster werden also von C++ - Routinen aufgerufen, und bei Anklicken eines Buttons in einem Tcl/Tk-Fenster wird eine C++ - Funktion gestartet, die an diesen Button gebunden ist.
Der Aufruf erfolgt mit

dispo2 tempdir projectname

2 DISPO2: Files

Es wird erwartet, daß die C++ - Files

index_struct.h	dispo2.cc
index_func.h	index_func.cc
convert.h	convert.cc
canvas.h	canvas.cc
deps.h	deps.cc
groups.h	groups.cc
aux.h	aux.cc
dispo2_win.h	dispo2_win.cc
convert_win.h	convert_win.cc

in einem Unterverzeichnis 'dispo2' des LooPo-Verzeichnisses stehen.

Die Files für das Tcl/Tk Frontend

Dispo2.tcl
Convert.tcl

müssen sich im Library-Unterverzeichnis des LooPo-Verzeichnisses befinden.
Mit dem Befehl

gmake dsp2

kann der Übersetzungsvorgang aus dem LooPo-Verzeichnis heraus gestartet werden.
Außerdem müssen, um einen korrekten Ablauf sicherzustellen, zur Laufzeit von DISPO2 das File mit den gefilterten Abhängigkeiten, das File mit der Gruppeneinteilung der Statements und das File mit den Indexräumen der einzelnen Statements für das jeweilige Projekt existieren. Diese Dateien werden üblicherweise durch den Aufruf der Programme 'ScanPars' [Günz] und 'Dependences' [Keim] erzeugt.

2.1 C++

Im folgenden werden ein kurzer Überblick über die C++ - Files und ggfs. einige Hinweise auf Besonderheiten gegeben.

2.1.1 index_struct.h

Die Datenstruktur, mit der die Indexräume verwaltet werden, ist eine doppelt verkettete Liste und sieht folgendermaßen aus:

```
typedef struct _index
{
        int i_loop;
        int j_loop;
        int k_loop;
        float x_val;
        float y_val;
        float z_val;
        int id;
        char* name;
        struct _index* next;
        struct _index* prev;
}
index;
```

Die ersten drei Felder enthalten die Schleifenkoordinaten des betreffenden Indexpunktes.
Die nächsten drei Felder enthalten die Koordinaten, an denen der Indexpunkt im Tk-Canvas gezeichnet werden soll.
Im Feld 'id' wird die eindeutige Nummer eingetragen, die das Canvas an jedes Objekt im Canvas vergibt, und dient dazu, einzelne Indexpunkte im Canvas zu identifizieren.
Das Feld 'name' hat bei Indexräumen keine Bedeutung. Dieselbe Datenstruktur wird jedoch auch für das sich mitdrehende Koordinatenkreuz verwendet. In diesem Fall wird hier der Name einer Schleifenvariablen gespeichert.
Die letzten beiden Felder enthalten den Nachfolger bzw. den Vorgänger des betrachteten Indexpunktes.
Die zweite Datenstruktur dient der Speicherung der Informationen aus dem Gruppen-File. Für jedes Statement wird ein Eintrag gemacht, der aus zwei Integerzahlen besteht, nämlich der Nummer des Statements und der dazugehörigen Gruppe.

```
typedef  struct _groupinfo
{
        int statement_nr;
        int group_nr;
        struct _groupinfo* next;
}
groupinfo;
```

2.1.2 dispo2.cc

Dies ist das Hauptmodul von DISPO2. Von hier aus werden alle Submodule aufgerufen. Außerdem befinden sich hier die Einstellungen für das Aussehen der Indexkugeln, wobei die Variable 'node_radius' den Radius einer Indexkugel, und 'abstand' den Abstand zweier benachbarter Indexkugeln angibt.

2.1.3 convert.cc

Dieses Modul übernimmt das Einlesen des Ungleichungssystems eines Indexraumes aus dem Indexspace-File und eliminiert evtl. Strukturparameter. Auch die Abhängigkeiten werden hier aus dem Filter-File eingelesen.

2.1.4 index_func.cc

Hier befinden sich die Funktionen, die aus einem Ungleichungssystem die Datenstruktur für den Indexraum aufbauen.

2.1.5 canvas.cc

Hier befinden sich alle Funktionen, die das Tk-Canvas betreffen, wie z.B. das Zeichnen der Indexkugeln und die Funktionen für Rotation, Verschiebung und Vergrößerung/Verkleinerung.

2.1.6 deps.cc

Hier geschieht die Behandlung der Abhängigkeiten zwischen den Indexpunkten. Das File enthält u.a. Funktionen zur Bestimmung der Quelle und des Zieles einer Abhängigkeit und die Funktion zum Zeichnen der Abhängigkeitspfeile.

2.1.7 groups.cc

Hier wird die Information aus dem Gruppen-File ausgelesen und in der dafür vorgesehenen Datenstruktur gespeichert.

2.1.8 aux.cc

In diesem Modul befinden sich verschiedene Hilfsfunktionen, u.a. zur Umwandlung von Float- oder Integerzahlen in Strings und zur Anzeige von Fehlermeldungen.

2.1.9 dispo2_win.cc

Hier befindet sich die Definition des Datentyps für das DISPO2-Hauptfenster. Außerdem werden hier die C++ - Aufrufe aus dem DISPO2-Hauptfenster behandelt. D.h. die Spezifikation aller Aktionen, die ausgeführt werden sollen, wenn im Fenster ein Button gedrückt wird, befindet sich ebenfalls hier.

2.1.10 convert_win.cc

Analog zu dispo2_win.cc befinden sich hier die Definitionen für das Convert-Fenster.

2.2 Tcl/Tk

Die nächsten beiden Abschnitte beschreiben die Tcl/Tk-Files für das Frontend.

2.2.1 Dispo2.tcl

Dieses File generiert das Hauptfenster von DISPO2.
Hier verdient der Slider für den Scaling-Faktor besondere Beachtung, da er eine Fließkommazahl darstellt, und deswegen kein Standard-Slider verwendet werden konnte. Er läuft von 100 - 300, die Anzeige muß gesondert auf 1.00 - 3.00 gesetzt werden.

2.2.2 Convert.tcl

Dieses File enthält das Gerüst für das Fenster zur Eliminierung der Strukturparameter.
Da die Anzahl der Strukturparameter zur Compile-Zeit nicht bekannt ist, müssen die Fensterelemente zur Anzeige der Strukturparameter und die Eingabefelder für die Werte dynamisch zur Laufzeit generiert werden.

3 3D - Ansicht

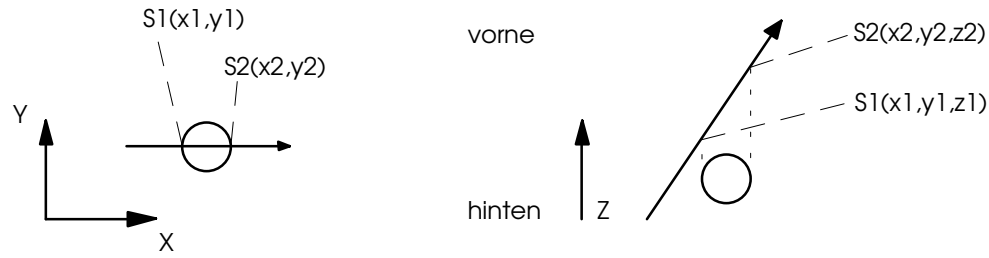
Ein großes Problem von DISPO2 ist die 3-dimensionale Darstellung der Indexräume und der Abhängigkeiten der einzelnen Indexpunkte. Da das Tk-Canvas nur 2-dimensional ist, mußte 'von Hand' dafür gesorgt werden, daß die Darstellung 3-dimensional richtig erfolgt.
Zur Anzeige der Elemente in einem Canvas verwendet Tk eine sog. 'Display-Liste' (siehe [Oust]). D.h., wenn Elemente neu kreiert werden, werden sie hinten an die Display-Liste angehängt und es gilt: Elemente weiter hinten in der Display-Liste verdecken Elemente weiter vorn. Zusätzlich existieren Befehle um die Reihenfolge der Display-Liste zu verändern.
Unabhängig davon gilt für DISPO2: Elemente mit größerer Z-Koordinate verdecken Elemente mit kleinerer Z-Koordinate, d.h. die Z-Achse verläuft aus dem Bildschirm heraus auf den Betrachter zu.

3.1 Indexpunkte

Korrekte 3D-Ansicht der Indexpunkte ist leicht dadurch zu erreichen, daß die verkettete Liste, in der die Indexpunkte gespeichert werden, nach aufsteigender Z-Koordinate sortiert wird. In dieser Reihenfolge werden nun die Indexkugeln gezeichnet (sie stehen dann auch in dieser Reihenfolge in der Display-Liste), und man erreicht den gewünschten Effekt, daß Indexkugeln mit größerer Z-Koordinate Indexkugeln mit kleinerer Z-Koordinate verdecken (Hier wird unterschieden zwischen 'Indexpunkt' und 'Indexkugel'. Jeder Indexpunkt wird im Canvas von einer Indexkugel dargestellt. Bei diesen Indexkugeln handelt es sich allerdings in Wirklichkeit um 2-dimensionale Gebilde, was die Berechnung der Schnitte mit Abhängigkeitspfeilen beschleunigt. Richtiger wäre also 'Indexscheibe'.).

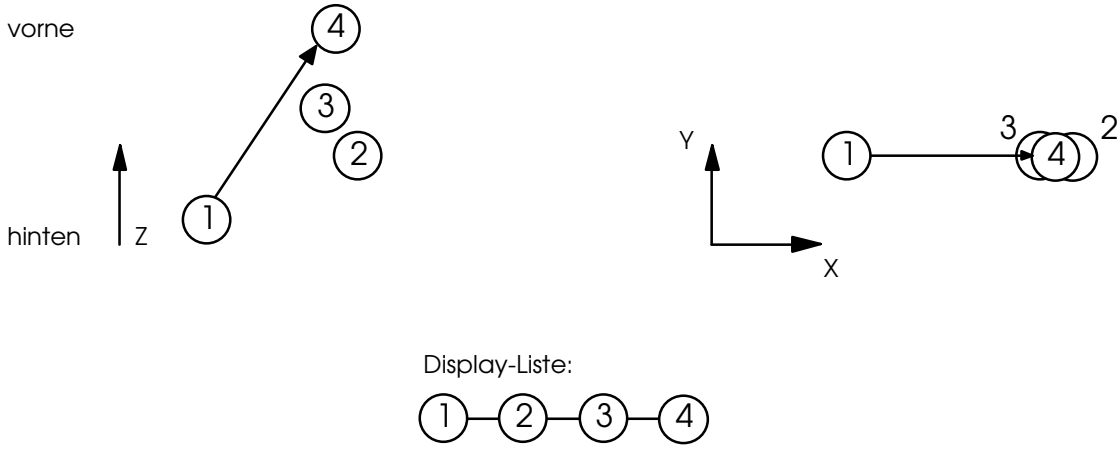
3.2 Abhängigkeiten

Abhängigkeitspfeile stellen nun insofern ein Problem dar, als man ihnen keine feste Z-Koordinate und somit keinen festen Platz in der Display-Liste zuordnen kann, da sie von einem Indexpunkt mit Z-Koordinate z_1 zu einem Indexpunkt mit Z-Koordinate z_2 verlaufen, und i.a. gilt: $z_1 \neq z_2$.
Unproblematische Ausnahmen sind nur Abhängigkeitspfeile, die die Abhängigkeit eines Indexpunktes von sich selbst darstellen, da diesen Abhängigkeitspfeilen eine eindeutige Z-Koordinate zugeordnet werden kann.
Die Lösung dieses Problems sieht folgendermaßen aus:
1. Schritt: Zunächst wird der Abhängigkeitspfeil in der Display-Liste direkt hinter seinem Quell-/Zielpunkt platziert, je nachdem, welcher von beiden Indexpunkten die kleinere Z-Koordinate besitzt. Danach wird die Liste der Indexpunkte vom Quellpunkt der Abhängigkeit bis zum Zielpunkt durchlaufen, und bei jedem Indexpunkt geprüft, ob er sich in Z-Richtung vor oder hinter dem Abhängigkeitspfeil befindet. Diese Prüfung geschieht dadurch, daß zunächst die Z-Koordinate vernachlässigt wird, und zuerst die Schnittpunkte des Abhängigkeitspfeiles mit der Indexkugel in der XY-Ebene berechnet werden. Existieren keine Schnittpunkte kann zum nächsten Indexpunkt übergegangen werden. Existieren jedoch Schnittpunkte, so werden dann die Z-Koordinaten dieser Schnittpunkte auf dem Abhängigkeitspfeil berechnet.



Ist eine der Z-Koordinaten größer als die der Indexkugel, wird angenommen, daß sich die Indexkugel hinter dem Abhängigkeitspfeil befindet, andernfalls vor ihm. Befindet sie sich vor dem Abhängigkeitspfeil, kann zum nächsten Indexpunkt übergegangen werden, wenn nicht, wird der betreffende Indexpunkt in der Display-Liste vor den Abhängigkeitspfeil verschoben. Dabei kann dieser Indexpunkt andere Indexpunkte in der Display-Liste überspringen, hinter denen er eigentlich liegen müßte, und vor ihnen zu liegen kommen. Dieser Fehler muß nachträglich in einem 2. Schritt korrigiert werden.

2. Schritt: Hier wird die Liste der Indexpunkte nocheinmal vom Quell- bzw. Zielpunkt der Abhängigkeit, je nachdem, welcher die kleinere Z-Koordinate besitzt, bis zum eben verschobenen Indexpunkt durchlaufen, und bei jedem Indexpunkt geprüft, ob er sich in der XY-Ebene mit dem im 1. Schritt verschobenen Indexpunkt überschneidet. Ist dies der Fall, muß auch dieser Indexpunkt in der Display-Liste vor dem eben verschobenen Indexpunkt platziert werden. Für den in Schritt 2 verschobenen Indexpunkt muß natürlich auch wieder Schritt 2 ausgeführt werden. Die Ausführung von Schritt 2 erfolgt also rekursiv.



4 Transformationen

Auf die Realisierung der Transformationen durch Multiplikation von Transformationsmatrizen und Gleichbehandlung der Verschiebung mit Vergrößerung/Verkleinerung und Rotation durch Verwendung von homogenen Koordinaten, wie in [FDFH] vorgeschlagen, wurde verzichtet, da der wesentliche Vorteil der beiden Methoden, nämlich die Zusammenfassung mehrerer Transformationen zu *einer* Transformationsmatrix, hier deswegen nicht ausgeschöpft werden kann, weil die Anwendung immer nur eine Transformation auf einmal zuläßt.
Die Verschiebung eines 3-dimensionalen Punktes benötigt bei Verwendung von Transformationsmatrizen und homogenen Koordinaten mindestens 9 Additionen und 12 Multiplikationen. Die einfachste und offensichtliche Methode, die auch verwendet wurde (siehe Kap. 4.1), kommt mit 3 Additionen aus.

4.1 Verschiebung

Die Verschiebung eines Indexpunktes wird einfach dadurch erreicht, daß zu den Koordinaten des Indexpunktes der Verschiebungswert addiert wird. Der Verschiebungswert ist variabel, jedoch für X- und Y-Richtung identisch, und kann zur Laufzeit durch einen Slider verändert werden. Eine Verschiebung in Z-Richtung ist nicht vorgesehen.

4.2 Vergrößerung/Verkleinerung

Auch diese Transformation wird einfach dadurch realisiert, daß die Koordinaten eines Indexpunktes mit dem Scaling-Faktor multipliziert werden. Dieser ist ebenfalls durch einen Slider einstellbar, jedoch für X-, Y- und Z-Richtung gleich. Das Zentrum einer Scaling-Transformation ist immer die Mitte des betrachteten Auschnittes des Canvas, der durch zwei Scrollbars am Rand des Canvas-Fensters verändert werden kann.

4.3 Rotation

Wie schon bei den beiden vorangegangenen Transformationen beschrieben, werden Indexpunkte dadurch rotiert, daß ihre Koordinaten mit den entsprechenden Sinus- und Kosinustermen multipliziert werden. Es ist eine Rotation um alle drei Raumachsen möglich, der Rotationswinkel ist einstellbar. Drehachse der Rotation um eine bestimmte Raumachse ist eine, zur jeweiligen Raumachse parallele Gerade durch den Mittelpunkt des Indexraumes.

4.4 Komplexität

Bis auf die Rotation um die X- und Y-Achse haben alle Transformationen von DISPO2 lineare Komplexität, d.h. die Liste der Indexpunkte wird pro Funktionsaufruf konstant-oft durchlaufen (meistens 1x, aber im Höchstfall 2x).
Bei Rotation um die X- und Y-Achse liegt quadratische Komplexität vor, da hier nach der Rotation der Indexpunkte zusätzlich die Liste wieder nach aufsteigender Z-Koordinate sortiert werden muß. Hier besteht allerdings noch die Möglichkeit einer Optimierung (siehe Kap. 5).
Ein Problem stellt die Darstellung der Abhängigkeiten dar, denn hier liegt im schlimmsten Fall fakultative Komplexität in der Anzahl der Indexpunkte vor. Dieser Fall tritt in der Praxis jedoch nicht auf, und in der Regel ist die Komplexität linear zum Produkt aus der Anzahl der dargestellten Abhängigkeiten und der Anzahl der Indexpunkte. Auch hier besteht sicherlich noch Optimierungsbedarf.

5 Erweiterung und Optimierung

Als Erweiterung der Funktionalität wäre zum Beispiel ein sog. 'Alignment', d.h. die automatische Ausrichtung des Indexraumes in eine bestimmte Richtung denkbar, oder zusätzlich zum Source-Indexraum die Darstellung des Target-Indexraumes, und damit verbunden die Darstellung von Indexräumen mit 'Löchern', und schließlich die Darstellung von While-Schleifen, oder die Darstellung von Ausführungsräumen, wie in [Zwick].
Als Optimierung wäre eine Erhöhung der Ausführungsgeschwindigkeit besonders wünschenswert. Zwei mögliche Ansätze sind:
- Nach jeder Rotation werden die Indexpunkte einzeln wieder in die Liste der Indexpunkte, die ja nach aufsteigender Z-Koordinate sortiert ist, einsortiert. Durch den Einsatz von schnellen Sortieralgorithmen wäre sicher ein Zeitgewinn möglich.
- Das Zeichnen der Kreise für die Indexkugeln erfordert sehr viel Zeit. Durch die Verwendung von Bitmaps läßt sich die Zeit zum Zeichnen der Indexkugeln extrem reduzieren. Der Preis dafür ist allerdings der Verlust der stufenlosen Scaling-Funktion. Das ist auch der Grund, warum am Ende auf diese Optimierung verzichtet wurde.
Fände man allerdings eine Möglichkeit mehrfarbige (für eine vernünftige räumliche Darstellung der Indexkugeln sind mind. 3 Farben erforderlich) Bitmaps schnell und stufenlos zu skalieren, wäre dies eine sehr gute Möglichkeit zur Geschwindigkeits-steigerung.

6 Zusammenfassung

DISPO2 stellt ein äußerst komfortables Tool zur Darstellung von Indexräumen mit vielen Manipulationsmöglichkeiten dar. Bei steigender Zahl von Indexpunkten (ab etwa 50-100 Indexpunkten, stark abhängig von der Anzahl angezeigter Abhängigkeiten) wird die Darstellungsgeschwindigkeit leider etwas unbefriedigend. Bleiben die Beispiele allerdings einigermaßen beschränkt, so hat man mit DISPO2 eine gute Möglichkeit, um sich einen Überblick über das Aussehen von Indexräumen und Abhängigkeiten zu verschaffen.

Literaturangabe

[FDFH] Foley, van Dam, Feiner, Hughes: 'Computer Graphics: Principles and Practice', Addison Wesley, 1990.

[Oust] John Ousterhout: 'Tcl and the Tk Toolkit', Addison Wesley, 1994.

[Günz] Robert Günz: 'Der neue Scanner/Parser', Praktikum, Fakultät für Mathematik und Informatik der Universität Passau, noch nicht erschienen.

[Keim] Harald Keimer: 'Vergleich von Verfahren zur Datenabhängigkeits-analyse', Diplomarbeit, Fakultät für Mathematik und Informatik der Universität Passau, noch nicht erschienen.

[Zwick] Axel Zwicklbauer: 'Graphische Polyederdarstellung', Diplomarbeit, Fakultät für Mathematik und Informatik der Universität Passau, 1993, 80/ST 1000 Z 98.