nds 2004 : caad : arch : ethz : ch

Parametric Construction Stylesheets

Programmierung parametrisierter Konstruktions-Stylesheets und Generierung automatisierter Produktionszeichnungen

am Beispiel des NDS2004- Abschlussprojektes

Michelangelo Ribaudo
nds 2004 : caad : arch : ethz : ch

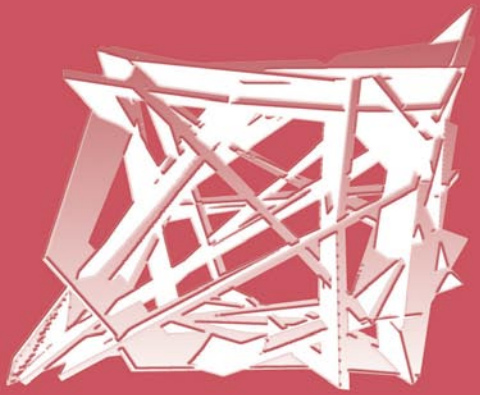
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ETH ZÜRICH
Chair of CAAD
Prof. Ludger Hovestadt
HIL E 15.1
ETH Hönggerberg
8093 Zürich
Switzerland

www.caad.arch.ethz.ch

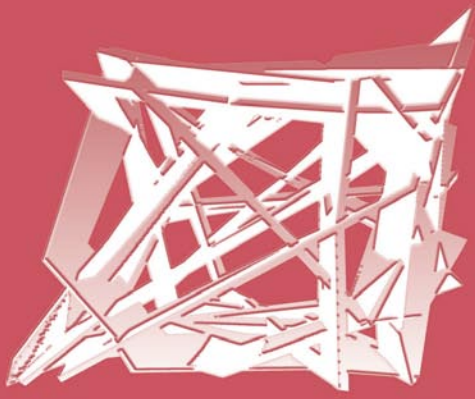
Layout | Li-hsuen, Yeh
Text & Drawings | Michelangelo Ribaudo

Zürich, October 2004



DANKSAGUNGEN

Professor Lehrstuhl CAAD	Ludger Hovestadt
Vom CAAD Leehrstuhl	Sibylla Spycher Oliver Fritz Odilo Schoch Markus Braach Russell Loveridge Karsten Droste Kai Strehlke Oskar Zieta
NDS Kurs Koordinator	Philipp Schaerer
ETHZ Department of Architecture Research/Postgraduate Studies ETHZ Betreuung Nachdiplomstudien	Bruno Dobler Anita Buchschacher
NDS2004	Christian Dürr If Ebnöther Jörg Grabfelder Anna Jach Jae Hwan, Jung Alexandre Kapellos Irene Logara Michelangelo Ribaudo Hanne Sommer Agnieszka Sowa Detlef Wingerath Thomas Wirsing Li-Hsuen, Yeh



Abstract English

Parametric Construction Stylesheets

Programming of parametric construction stylesheets and automatic production drawings on the example of NDS2004 final project.

During the postgraduate studies in CAAD at the department of architecture at ETH Zurich the research is mainly focused on computer based architectural design and its automatic production.
The final group work of NDS2004 students deals with generation, optimization and automatic production of a programmed complex structure. The generation of such a construction with the customary CAAD techniques is just possible with big efforts.
The present thesis is centred on the use of programmed tools as alternative or as support of the classical computer aided architectural design methods.
This thesis shows among other things how were programmed/generated the mathematical descriptions of the frames, the joints and the production drawings using MEL (Maya Embedded Language).
Further will be discussed pros and cons of the imported and exported digital data structures for their respective purpose like the generation of the joint details, the model visualizations, the different prototypes and the generation of the construction stylesheets.
The result of this work will be shown by visualizations of digital models as well as by using rapid prototyping methods and CNC machines.
Moreover this thesis will deal with the programming of stylesheets which were used to generate variants of constructions.
The NDS2004 prototype represents such a variant and was produced with the above mentioned programmed tools.

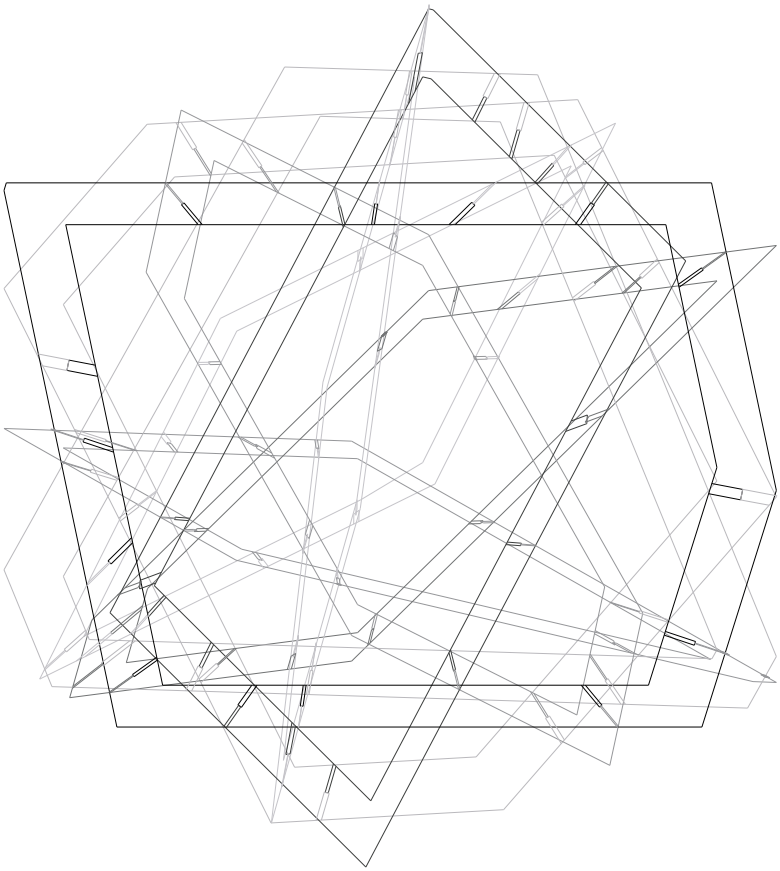
Abstract Deutsch

Parametric Construction Stylesheets

Erstellung parametrisierter Konstruktions-Stylesheets und Generierung automatisierter Produktionszeichnungen am Beispiel des NDS2004-Abschlussprojektes.

Während des CAAD-Nachdiplomstudiums am Lehrstuhl für CAAD des Architektur Departements an der ETH Zürich konzentriert sich die Forschung hauptsächlich auf computerunterstütztes Design und dessen automatisierten Produktion.
Das Endgruppenprojekt der Studenten des NDS2004 befasst sich mit der Generierung, Optimierung und automatisierten Produktion einer programmierten komplexen Struktur. Eine solche Konstruktion ist mit üblichen CAAD-Techniken nur mit sehr viel Aufwand herstellbar.
Die vorliegende These behandelt den Einsatz von programmierten Werkzeugen als Alternative oder zur Unterstützung des klassischen computerunterstützten Architekturdesigns.
In dieser These wird unter anderem veranschaulicht wie für dieses Projekt die mathematische Beschreibung der Rahmenkonstruktion, der Knotenpunkte und der Produktionszeichnungen in MEL (Maya Embedded Language) programmiert/generiert worden sind.
Behandelt werden auch die Vor- und Nachteile der importierten und exportierten digitalen Datenstrukturen für den jeweiligen Zweck, wie zum Beispiel für die Generierung des Knotendetails, die Modellvisualisierungen, die verschiedenen Prototypen und die Erstellung der Konstruktions-Stylesheets.
Das Ergebnis dieser Arbeit wird anhand von digitalen Modellvisualisierungen, wie auch durch die Benutzung von Rapid Prototyping Methoden und von CNC gesteuerten Maschinen vorgestellt.
Darüber hinaus befasst sich diese Thesis auch mit der Programmierung von Stylesheets, die für die Erstellung von Konstruktionsvarianten gedient haben.
Der NDS2004-Prototyp stellt eine solche Variante dar und ist mit den oben genannten programmierten Werkzeugen erzeugt und hergestellt worden.

Inhalt:



1.0 Kontext

1.1 Die Professur für CAAD

Die Professur für CAAD des Departements Architektur der ETH Zürich wendet aktuelle Informationstechnologien in der architektonischen Praxis an.
Das Interesse reicht von der Entwurfsunterstützung durch digitale Medien über die Produktion mit computergesteuerten Maschinen bis hin zum intelligenten Gebäudebetrieb.

1.2 Nachdiplomstudium CAAD

Das Nachdiplomstudium CAAD wendet sich an diplomierte und praktizierende Architektinnen und Architekten, sowie an Absolventinnen und Absolventen verwandter Studienrichtungen aus dem In- und Ausland.

Ausbildungsschwerpunkt ist der computergestützte architektonische Entwurf (CAD) und seine automatisierte Produktion (CAM).
Das Nachdiplomstudium ist ein Vollzeitstudium dessen Programm sich in etwa 7 -10 Modulen unterteilt, die in seminaristischer Form durchgeführt werden. In jedem Modul sollen spezifische Kenntnisse vermittelt werden.

Das Studium endet mit einer individuellen Thesis und einem Gruppenprojekt.

1.3 Einleitung

Beim diesjährigen Gruppenprojekt handelt es sich um ein Pavillon, der als Experiment die Forschungen des Nachdiplomkurses (NDS2004) CAAD bündelt.
Eines der Ziele der Forschungsarbeit ist es, dass weder Entwurfs- noch Konstruktionszeichnung von Hand aufgerissen, sondern mit Hilfe des Computers generiert werden.
Die Geometrie der Struktur ist programmiert und kann vom Entwerfenden mittels Parametern variiert werden. In Abhängigkeit dieser Parameter können beliebig viele Varianten erzeugt und im Modelmassstab mit einem 3D-Ausgabegerät räumlich geplottet werden.
Das Ideal einer kontinuierlichen digitalen Prozesskette, welche die erlernten Fähigkeiten auf dem Gebiet des computergestützten architektonischen Entwurfs (CAD) und seiner automatisierten Produktion (CAM) aufnimmt, konnte wegen der knappen Produktionszeit nur mit zusätzlicher Handarbeit eingehalten werden.
Das gebaute Ergebnis ist nicht als architektonisches Statement zu verstehen. Es dient keinem Zweck und seine Form folgt keiner Funktion. Der gebaute Prototyp veranschaulicht alleine die Umsetzung aktueller Informationstechnologien in Planung und Konstruktion.
Viele unterschiedliche Varianten werden generiert und die einzelnen Prototypen unter Verwendung von CNC-Maschinen materialisiert und gebaut. Die Prototypen sind massstabslos. Beim Hauptprototyp, mit einer Kantenlänge von 2,60m x 2,60m, handelt es sich um eine begehbare Struktur. Das Mass 2,60m x 2,60m resultiert aus der Raumhöhe des Ausstellungsortes. Zum Transport und Aufbau kann die gebaute Struktur in einzelne tragbare Elemente zerlegt werden.
Für die Produktion kommen die an der ETH-Hönggerberg zur Verfügung stehenden CNC-Maschinen zum Einsatz: Eine 3-achsige Fräse (Produktionstisch 2.40m x 1.55m), eine Industrie-Laserschneidmaschine (Produktionstisch 3.00m x 2.00m), einen 3D-Gipsdrucker (Produktionsgröße 0.20m x 0.20m x 0.25m) und eine Modellaserschneidmaschine (Produktionstisch 0.45m x 0.81m).


3-achsige fräse


Industrie-Laserchneidemaschine


Modellaserchneidemaschiene


3D-Gipsdrucker

2.0 Konzept

Ausgangslage ist ein gedachtes Volumen in Form eines Würfels. Tragstruktur und Hülle ist ein irreguläres Flechtwerk von zueinander abgewinkelten und sich kreuzenden Stegen. Lage und Abwinklung der Stege werden durch eine Anzahl sich verkeilender Ebenen beschrieben. Ein inneres Volumen wird subtrahiert und definiert die Tiefe der Stege. *Bild 2.0*
Am Computer werden unterschiedlich viele Varianten erzeugt und einzelne Prototypen anhand von 3D-Ausgabegeräten materialisiert und gebaut. Hauptprototyp ist eine begehbare Struktur. Die genaue Abmessung wird durch die Raumhöhe des Ausstellungsortes bestimmt.
Der Pavillon soll zum Transport und Aufbau in einzelne tragbare Grössen zerlegt werden und einfach demontierbar sein. Für die Produktion kommen die an der ETH-Hönggerberg zur Verfügung stehenden CNC-Maschinen zum Einsatz.

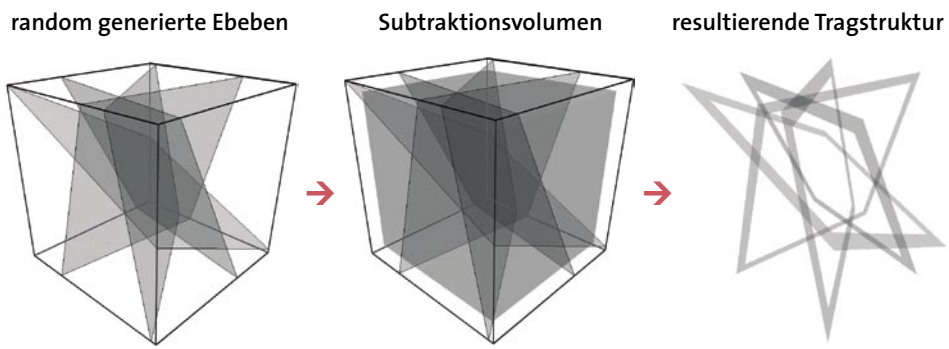


Bild 2.0

2.1 Prolog

„ Programmieren statt zeichnen “

Kohärent zum Forschungsbereich des Lehrstuhls für CAAD wurde der Pavillon programmiert und nicht gezeichnet. Die traditionellen Entwurfsmethoden mit konventionellen CAAD-Techniken und die Verwendung von selbst programmierten Werkzeugen im architektonischen Entwurf weisen sehr viele Analogien auf. Deshalb können diese programmierten Werkzeuge parallel als arbeitserleichternde Unterstützung in der Entwurfsphase oder in der Werkplanung eingesetzt werden, diese aber auch komplett ersetzen.
Der Prozess der Lösungsfindung für das Gestaltungsproblem basierte auf die Methode von Versuch und Irrtum. Es wurden Regeln benutzt, um versuchsweise Lösungen zu generieren und dann Prädikate zu berechnen und um festzustellen, ob diese Lösungen akzeptabel sind. Es handelt sich dabei um nichts anderes als um einen Prozess von Generieren und Testen in einem Lösungsraum. Der Gestalter beobachtet gewissermassen von aussen den Prozess der Formgenese und greift bei Gelegenheit steuernd ein.
Der ideale IT-Gestalter handelt nicht mehr in einer konkreten Situation, sondern in einer abstrakten Welt, in der er alles Notwendige spezifiziert hat, sich dann zurücklehnt und zusieht, wie die von ihm geschaffene «Maschine» ein Problem löst. Befriedigung in dieser Art des Entwerfens ist die einwandfreie Funktion der Maschine und das Ineinandergreifen wie Zahnräder der Begriffe, mit denen die Prototypen beschrieben werden.
Die syntaktischen Regeln, die die abstrakte Welt der Entwürfe beherrschen etablieren einen architektonischen Typus. Die in einer kritischen Sprache ausgedrückten Prädikate etablieren die Anforderungen eines bestimmten Moments und Kontexts.
Die Aufgabe des Entwerfens ist es, den Typus in einer Form zu instantiieren, die diesem gegebenen Moment und Kontext gerecht wird. Um einen Typus in einem konkreten Kontext instantiieren zu können, muss ein intelligentes Design-System über eine kritische Sprache verfügen, die den Zusammenhang zwischen den Formen und ihrem Funktionieren im jeweiligen Kontext beschreibt und damit den Entwurfsprozess steuert.
So sehr Formengrammatiken (Algorithmen) als Werkzeuge zur spielerischen Erzeugung von Formen ihren Wert haben, so wenig lässt sich aber der Entwurfsprozess insgesamt auf sie reduzieren.
Man kann hier eine Verschiebung des kreativen Prozesses auf verschiedene Weise konstatieren: Erstens entsteht der Entwurf in der computergestützten Kreativität durch einen Algorithmus,

d.h. durch eine maschinengestützte Kreativität und zweitens verlagert sich die persönliche Kreativität von der Produktion von Formen auf die Ebene der Interpretation der Bedeutung von Gebilden. Mit anderen Worten ist die architektonische Kreativität nicht nur ein Formenspiel, sondern auch ein Sprachspiel. In diesem Zusammenhang wird der Computer nicht zur Reproduktion bzw. Vorkonfiguration von Formen verwendet.
Das Ziel ist es den Computer als Katalysator des Neuen zu verstehen. Der Computer wird zu einem Werkzeug der Unschärfe im Entwurfsprozess, d.h. der Gestalter ist nun befähigt die Algorithmen für sich zu nutzen, um mit den unerwarteten Ergebnissen Neues zu provozieren.
Eine andere Art der Wechselbeziehung zwischen Computer und Gestalter definiert einen Prozess, der an jedem Punkt unterbrochen werden kann, um die Parameter neu zu ordnen und den Prozess erneut fortzusetzen.
Um am Ende zu einem Produkt zu gelangen, wählt der Gestalter aus einer Bandbreite möglicher Lösungen aus. Das Endergebnis kann als ein fixierter Zustand dieser Möglichkeiten gesehen werden.

2.2 Digitale Prozesskette

Die gebaute Instanz entstand aus den vier Prozessen folgender digitalen Kette:

Generation

Ein eigens in der CAD-Software Maya geschriebenes Programm generiert unter Angabe der Würfelabmessung, der Ebenenanzahl und des subtrahierenden Volumens die Geometrie der Struktur. Die Werte können vom Entwerfenden variiert werden. In Abhängigkeit dieser Parameter können beliebig viele Varianten (Instanzen) erzeugt werden.
Jeder Variante liegt ein dreidimensionales Datenmodell zugrunde. Dieses Datenmodell kann zur Visualisierung gerendert und an verschiedene Ausgabegeräte wie Drucker oder 3D-Printer übermittelt werden.

Optimization

In einem zweiten Schritt wird die Geometrie der Struktur optimiert. Je nach konstruktiven oder materialspezifischen Vorüberlegungen können Bedingungen für die Geometrie festgelegt werden: Der Abstand benachbarter Knoten soll ein Mindestmass nicht unterschreiten, die maximale Steglänge hingegen soll zwischen zwei Knoten aus statischen Gründen ein gewisses Mass nicht überschreiten.
Die Rahmengeometrie wird wie folgt optimiert: Für jeden Stegrahmen – als durchstanzte Ebene verstanden - wird ein Fitnesswert errechnet. Bei jedem Durchgang wird der Rahmen mit dem schlechtesten Wert eliminiert und gleichzeitig durch einen neuen ersetzt. Die Fitnesswerte der einzelnen Stegrahmen werden für jede Variante aufsummiert und als globale Fitness beschrieben.
Die Variante mit dem besten Wert wird vom Programm für die Weiterverarbeitung empfohlen. Es handelt sich um einen evolutionären Prozess, der beliebig wiederholt und manuell abgebrochen werden kann.

Automated Construction Drawings

Sämtliche Rahmen und die entworfenen Steckverbindungen sollen automatisch aus Platten gefräst werden. Dieser Produktionsweg erfordert die Erstellung eines Zuschnittsplans.
Hier setzt ein drittes Programmskript ein. Es übernimmt das dreidimensionale optimierte Datenmodell aus der vorhergehenden Phase und projiziert jeden Stegrahmen mit den zugehörigen Schlitzverbindungen massstabsgetreu in eine Zeichnungsebene.
Diese Zeichnung dient als Werkplan für die bauliche Realisation. Die format- und transportbedingten Steckverbindungen werden nachträglich manuell ergänzt.

Computer aided manufactoring

Der Plan wird anschliessend mit dem Programm SurfCam in einen maschinenverständlichen Code übersetzt und an die CNC-Fräse übermittelt.
Die im Plan enthaltenen Linien werden von der Maschine als Fräspfade interpretiert.

3.0 Programmierung

Das Programm sollte nach dem Oben in *Bild 2.0* abgebildeten Prinzip in der Lage sein einen Kubischen Pavillon zu generieren.
Da weder Konstruktionssystem, Massstab, Beschaffenheit noch Bauteildimensionen Definiert waren, sollte der Programmcode so flexibel wie möglich aufgebaut werden.
Nach unserer Idealvorstellung sollte dann das Programm beliebig viele Lösungsvorschläge (Instanzen) liefern, so dass die zu bauende Instanz nur noch aus dem Satz vorgeschlagener Geometrien hätte ausgewählt werden können.
Es war uns Bewusst, dass aus dem Generierungsprozess keine 'baubaren' Instanzen resultiert wären, ohne zuvor oder danach eine an das Konstruktionssystem angepasste Optimierung durchzuführen.
Nach dem Optimierungsprozess sollten dann die Konstruktionsspezifischen Daten hinzugefügt werden, so dass ein detailliertes dreidimensionales Digitalmodell hätte visualisiert werden können.
Verschiedene Konstruktions-Systeme sollten aus den optimierten Daten durch das verwenden von 'Stylesheets' erzeugt werden.
Die Werkpläne des ausgewählten digitalen 3D-Modell sollten anschliessend voll automatisiert generiert werden. Wenn möglich sollten auch die für die CNC-Fertigung benötigten maschinenspezifischen Programmcodes erzeugt werden können und einen Konfigurator oder 'Userinterface' geschaffen werden.

In den Folgenden Paragraphen werden die Programmstruktur, die Funktionsweise und der mathematische Aufbau näher beschrieben.
Es werden ausserdem die aufgetretenen Schwierigkeiten erläutert, die Entwicklungszeit (zwei Monate) bedingten Kürzungen am Programmumfang und die eingegangenen Kompromisse begründet.

3.1 Mel (Maya Embedded Language)

Die Suche nach einer geeigneten Software für das Oben erläuterte Vorhaben hat die Verantwortlichen für die Programmierung, Agnieszka Sowa und der Autor dieser These dazu bewogen mehrere 3D-Software aus dem Bereich Architektur, Mechanik und der Filmindustrie in Betracht zu nehmen. Besonderes Augenmerk wurde auf dessen Funktionsumfang und Skriptsprache gelegt.
Ausser *modeling*, *animation*, *dynamics* und *rendering* sollte die Software eine funktionsreiche Skriptsprache besitzen und sehr schnell 3D-Geometrien verarbeiten und visualisieren können.

Die in die engere Wahl genommenen Skript-Sprachen waren *VectorScript* (für *VectorWorks*), *MAXScript* (für *3Dstudio MAX*) und *MEL* (*Maya Embedded Language*), wobei die Entscheidung auf die Letztere Programmiersprache fiel.
Der größere Funktionsumfang und die schnelle *3D-engine* versprachen mehr Flexibilität, Effizienz und schnelle *NURBS* Visualisierungen.Leider konnte eines in der Anfangsphase in Betracht genommenes Programm, *TopSolid*, nicht eingesetzt werden, da keine *scripting* Schnittschnelle für den Benutzter vorhanden ist.
TopSolid überzeugte vor allem weil die von Hand gezeichneten Geometrien auf einfacher Weise zu parametrisieren sind. Ausserdem bietet es hilfsreiche Werkzeuge für die CNC-Fertigung an.

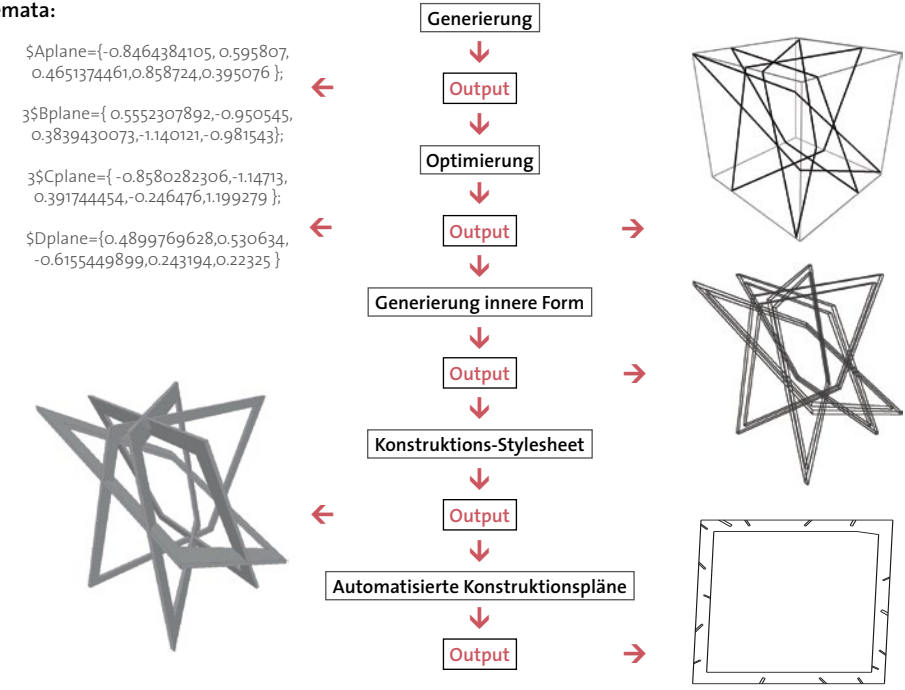
Alle Scripts, die für Generierung, Optimierung und Produktion des NDS2004 Studienobjekts benutzt wurden, sind ausschliesslich in MEL programmiert worden um sicherzustellen, dass keine Komplikationen durch den Datenaustausch zwischen den Prozessen entstanden wären. Bei abgeschlossener Programmierung kann man nun bestätigen, dass MEL eine sehr effiziente Skriptsprache ist. Das Programm ist jedoch nicht für lange und komplizierte Rechenvorgänge geeignet, wie sie für die Optimierungsprozesse des NDS2004 Prototyp nötig waren. Perl oder Java wären zu diesem Zweck besser geeignet gewesen.

Bild 3.0 http://www.alias.com

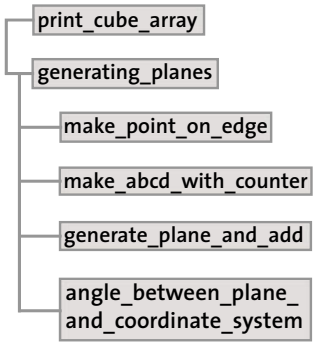
3.2 Programmstruktur

Für den Entwurf des NDS2004 Studienobjekt und für die Generierung der Werkpläne sind verschiedene Teilprogramme geschrieben worden.
Diese Teilprogramme werden Prozeduren genannt. Jede Prozedur funktioniert nach dem Blackbox-Prinzip, welches typisch ist für objektorientiertes Programmieren.
Die im Code enthaltenen mathematischen Beschreibungen der Geometrien werden in Form von Algorithmen, Parameter und in eine für das Programm verständliche Sprache, in diesem Fall MEL, geschrieben.

Schemata:



3.3 Generierung



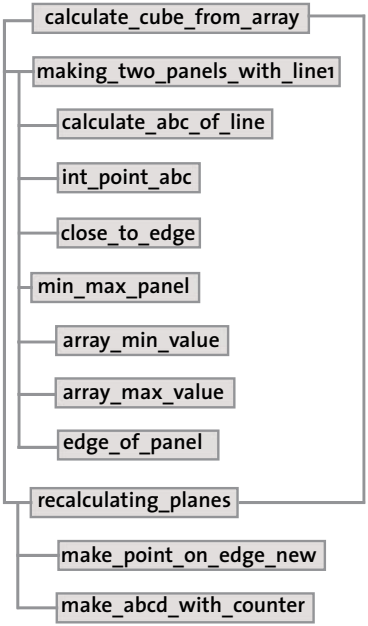
Die Generierungsprozedur des NDS2004 Prototyp enthält die mathematische Beschreibung der Struktur. Nach dieser Beschreibung soll der Computer einen ersten Output liefern. Der Designer überlässt dabei dem Computer eine gewisse gestalterische Freiheit, da er nur die „Randbedingungen" für das Output vorgibt.
Dies ist ein wichtiges Merkmal der computergenerierten Architektur. Der Designer kann dabei über die delegierte Gestaltungsfreiheit entscheiden. Dies kann er durch schrittweise, immer mehr einschränkenden Vorgaben, präzisere Beschreibungen oder Parameter steuern.

Im Oben genannten Studienobjekt sind folgende Einschränkungen in den Generierungsprozeduren enthalten:

- Die Ebenen sollen zufällig innerhalb des Kubus generiert werden
- Die Ebenen sollen nicht parallel zueinander oder zu den Seiten des Kubus sein.

Der Gestallter konnte hingegen die Anzahl der zu erstellenden Ebenen durch den Parameterwert *$numOfPlates* selber definieren.
Der Leser kann Näheres zur Generierung in der These von Sowa Agnieszka *Generation and optimization of complex and irregularconstruction/structure on the example of NDS 2004 final project* erfahren.

3.4 Optimierung



In einem zweiten Schritt wird die Geometrie der Struktur optimiert.
Je nach konstruktiven oder materialspezifischen Vorüberlegungen können Bedingungen für die Geometrie festgelegt werden:

-Der Abstand benachbarter Knoten soll ein Mindestmass nicht unterschreiten, die maximale Steglänge hingegen soll zwischen zwei Knoten aus statischen Gründen ein gewisses Mass nicht überschreiten.

Die Rahmengeometrie wird wie folgt optimiert:

Für jeden Stegrahmen – als durchstanzte Ebene verstanden - wird ein Fitnesswert errechnet. Bei jedem Durchgang wird der Rahmen mit dem schlechtesten Wert eliminiert und gleichzeitig durch einen Neuen ersetzt.
Die Fitnesswerte der einzelnen Stegrahmen werden für jede Variante aufsummiert und als globale Fitness beschrieben.
Die Variante mit dem besten Wert wird vom Programm für die Weiterverarbeitung empfohlen.
Es handelt sich um einen evolutionären Prozess, der beliebig wiederholt und manuell abgebrochen werden kann.

Funktionsweise der Optimierungsprozeduren:

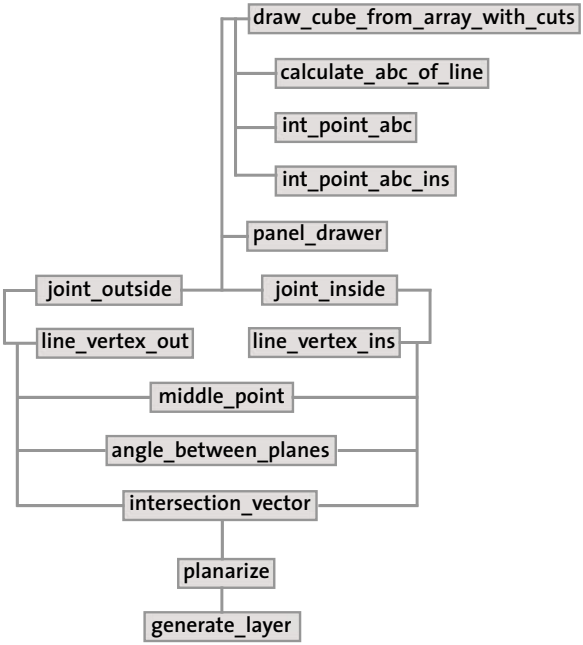
- Panel Findung
- Analyse der Schnittpunkte
- Analyse der Segmentlänge
- Berechnung des Fitnesswerte für jede Ebene
- Entfernen der Ebene mit dem niedrigsten Fitnesswerte
- Erzeugen einer neuen Ebene

fitnessPlane = {-123, -256, -89, -244, -156, -289, -266, -146}

Der niedrigste Wert der Fitnessfunktion entspricht der Ebene, die die schlechtesten Schnittpunkte zwischen den Ebenen verursacht. Diese Ebene wird gelöscht und durch eine neue ersetzt.

Der Leser kann Näheres zur Generierung in der These von Sowa Agnieszka *Generation and optimization of complex and irregular construction/structure on the example of NDS2004 final project* erfahren.

3.5 Konstruktion Stylesheet



Stylesheet, ist der neudeutsche Ausdruck für Stilvorlage, Formatvorlage oder Druckformatvorlage. In der EDVhat eine Formatvorlage die Aufgabe die Eingabe von Daten zu vereinfachen und soll zu einer einheitlichen und übersichtlichen Darstellung der Daten führen.
Bei Textverarbeitungs- oder DTP-Programmen ist eine Formatvorlage eine Kombination beliebiger Formatmerkmale, die mit einem gemeinsamen Namen belegt, gespeichert und bei Bedarf in einem Schritt einem Textteil zugewiesen werden können.

Textverarbeitungs- und DTP-Programme könnten theoretisch so programmiert sein, dass der Anwender in der Lage wäre, zu allen erdenklichen Formatmerkmalen Vorlagen anzulegen. In der Praxis ist das aber nicht der Fall: Üblich sind Zeichenformatvorlagen und Absatzformatvorlagen. Daneben wären zum Beispiel noch Abschnitts-, Tabellen- oder Listenformatvorlagen denkbar. Abgelegt werden Formatvorlagen normalerweise in einer so genannten Dokumentvorlage, also einer Musterdatei, die der jeweiligen Arbeitsdatei zugewiesen wird.
Ein Konstruktions Stylesheet ist, analog zu dem in der EDV beschriebenen üblichen Gebrauch, als Konstruktionssystemvorlage oder Konstruktionstypvorlage zu verstehen. Der selbe Datensatz, z.B. Konstruktionsraster oder Achskoordinaten von konstruktiven Bauteilen, können je nach aufgerufenen Stylesheet einen anderen Output erzeugen. Zum Beispiel könnte eine Konstruktion, die aus Rundrohren besteht per Knopfdruck, in eine aus Quadratrohren bestehende Konstruktion umgewandelt werden, oder die geometrischen Konstruktionsraster verändern werden.
Ein ähnliches Prinzip ist in gewissen CAD-Programmen wie z.B. Architectural Desktop, Nemetschek Allplan, oder ACAD-BAU zu finden.
In diesen Programmen gibt es so genannte „intelligente Objekte“. Es handelt sich um parametrisierte Planelemente die einen spezifischen Bautyp darstellen. Durch das Selektieren und durch das Anwählen eines anderen Bautyps können mit einem Mausklick alle selektierten Elemente in den ausgewählten Bautyp umgewandelt werden.
Ein einschaliges Mauerwerk kann z.B. durch diese Operation in ein zweischaliges Mauerwerk umgewandelt werden. Die an den Wänden angrenzenden oder integrierten Objekte würden sich jedoch nicht an der geänderten Wandstärke anpassen, da nicht die ganze Planzeichnung in sich parametrisiert ist.
Im NDS2004 Studienobjekt ist ein Stylesheet, ähnlich einem intelligenten Objekt parametrisiert, stellt aber darüber hinaus eine für die ganze Planzeichnung übergeordnete Zeichnungsanleitung dar.
Für jeden Konstruktionstyp werden gesonderte Stylesheets erstellt. Diese enthalten die Zeichnungsanleitungen der kompletten Planzeichnung und können in ein einziges Programm eingebettet werden. Die Auswahl des zu verwendenden Stylesheet kann dann durch eine Variable gesteuert werden.
In dieser These ist der für den gebauten NDS2004 Prototyp erstellten Stylesheet als austauschbare Prozedur programmiert worden. Es beinhaltet Informationen über Assemblierungsreihenfolge, Bauteilbezeichnung, Detail-Topologie wie auch über zeichnungsbezogenen Anleitungen.

Die Programmierung des Stylesheets erfogte Hand in Hand mit dem Fortschritt der Generierungs- und Optimierungsprozeduren, Material- und fertigungsrelevanten Entscheidungen.
Das Operieren als Schnittstelle zwischen Generierung und Produktion ist mit viel Aufwand verbunden, da jegliche Änderung am Generierungscode oder an der Konstruktion notwendige Anpassungen an der Schnittstelle erfordert. Besonders, die an den Details erbrachten Änderungen führten oft zur Entwicklung neuer Algorithmen bis hin zur Umgestalltung des Generierungscodes.

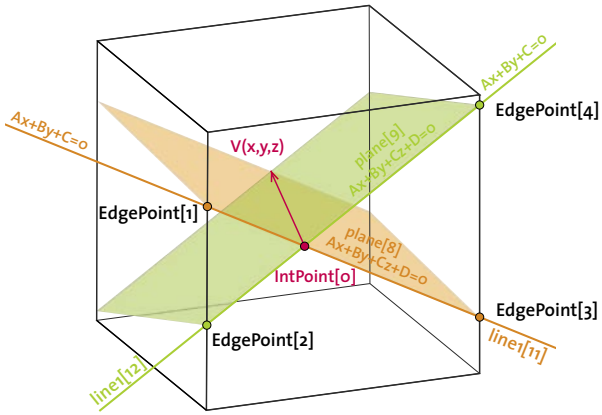
Im Folgenden sind zusammenfassend Teilbereiche des Projekts erläutert um einen weiteren Überblick über dessen Auswirkungen und Rückwirkungen auf dem Programmcode des Stylesheets zu erlangen.

3.6 Speicherung der Daten

Maya speichert in der *History* eines Zeichnungsfiles keine Einträge der *script editor* Schnittstelle, in der das MEL-Skript eingegeben wird.
Die errechneten Werte mussten somit in einem Text-File auf geeigneter Weise gespeichert werden, so dass die Daten für Wiederherstellungs- oder Weiterverarbeitungszwecken wieder verwendbar gewesen wären.
Die Daten des Generierungsskripts, Schnittpunkte, Nummern der Geraden und Ebenen, sowie Koordinatenpunkte von Geraden beziehen sich jeweils auf die Seiten des Kubus. Man kann von 2.5D Daten sprechen da die genaue Anordnung im Raum nur durch die Referenz zur Würfelseite bestimmbar ist. Diese Referenz war für die Erstellung der 3D Details von fundamentaler Bedeutung.

Der Leser kann Näheres zur Generierung in der These von Sowa Agnieszka *Generation and optimization of complex and irregular construction/structure on the example of NDS2004 final project* erfahren.

In *Bild 3.6* , die in Zusammenarbeit mit Agnieszka Sowa vereinbarte Datenstrukturierung.



INTERSECTION POINT

	x coordinate	y coordinate	z coordinate	first crossing line	second crossing
$IntPoint[o]	$XintPoint1[o]	$YintPoint1[o]	$ZintPoint1[o]	$firstLineIntPoint1[o]=11	$secLineIntPoint1[o]=12

EDGE POINTS

	x coordinate	y coordinate	z coordinate	which line make it
EdgePoint[1]	$XedgePoint1[1]	$YedgePoint1[1]	$ZedgePoint1[1]	$LedgePoint1[1]=11
EdgePoint[4]	$XedgePoint1[4]	$YedgePoint1[4]	$ZedgePoint1[4]	$LedgePoint1[4]=12

L - for Line

LINES

				which plane make it	
line1[11]	$Aline1[11]	$Bline1[11]	$Cline1[11]	$Pline1[11]=8	Ax+By+C=o
line1[12]	$Aline1[12]	$Bline1[12]	$Cline1[12]	$Pline1[12]=9	Ax+By+C=o

P - for Plane

PLANES

plane[8]	$Aplane[8]	$Bplane[8]	$Cplane[8]	$Dplane[8]	Ax+By+Cz+D=o
plane[9]	$Aplane[9]	$Bplane[9]	$Cplane[9]	$Dplane[9]	Ax+By+Cz+D=o

Bild 3.6

Arrays-Schemata in denen die Koordinatenpunkte x, y, z der Schnittpunkte, die a, b, c, d Koeffizienten der Linien und Ebenengleichungen gespeichert sind.

Beispiele:
Anhand des IntPoint[o] ist es möglich auf die Linien 11 und 12 zu schliessen, die diesen Schnittpunkt generieren.
Auf analogem Wege kann man anhand der Linien z.B. $Pline1[11] auf die Ebene schliessen, in diesem Fall Ebene 8.

4.0 Produktion

4.1 Material und Konstruktion

Die für das Gruppenprojekt durchgeführten Material- und Konstruktionsstudien sind unter der Regie von Ebnöther If in Kooperation mit dem Autor der vorliegenden These durchgeführt worden. Die Konstruktionssysteme und Details sind dem Programmcode subordiniert entwickelt worden. Wegen des engen Zeitplans musste man schon von Anfang an nicht nur dessen schnelle Produktion garantieren, sondern auch dessen Integration in den Programmcode.
Die folgende Tabelle fasst die für die Tragkonstruktion und Verblendungselemente in Betracht genommenen Materialien.

Verblendungs-elemente	**Tragkonstruktion (Stab- Flächentragwerke)**				
	Holz	GFK	PVC	Eternit	...
Holz	●				
GFK		○			
PVC	○				
Eternit					
Stahlblech	○				
Textil/Membran				○	
...					

Nach folgenden projektrelevanten Kriterien sollte die Konstruktion entwickelt werden:

- Kostensparende Konstruktion
- Leicht transportierbar
- Leichte, schnelle Assemblierung
- Schnelle CNC-Verarbeitung
- Keine Nachbearbeitung
- Platzsparende Lagerung

Nach der Auswertung von verschiedenen Materialien und Konstruktionsprinzipien sind mehrere Prototypen erstellt worden. Diese sind mit den ETH eigenen CNC-Gesteuerten Maschinen angefertigt worden. In dieser These wird jedoch nur auf die tatsächlich gebaute Konstruktionsvariante eingegangen. Es soll lediglich als Beispiel zur Veranschaulichung der in der Programmierung berücksichtigten Kriterien dienen.

Hersteller	Material	Farbe	Stärke [mm]	Abmessungen [m]	Fr./m2	kg/m2	Urteil
Argolite	solid core	weiss	8	2.6 x 1.3	80	11.2	Kleine Abmessungen
Küchler	solid core	weiss	8	2.75 x 2.04	56	11	Kleine Abmessungen
Badertscher	Trespa	–	10	3.65 x 1.86	101	14	Hoher Preis
Maagtechnik	Polypropyierve	weiss	8	2 x 1	–	leicht	Zu biegsam
Küchler	Multiplex Okumé	natura	10	3.1 x 1.83	24.1	5	Unschöne Beschichtung
Sax-Zim	Multiplex Phenol	braun	12	3.05 x 1.52	40.9	8.5	Kleine Abmessungen
Sax-Zim	3-ply	weiss	18	5.2 x 2.1	36.5	7.3	Nicht Ideal für Fräse
Küchler	OSB-Agepan	beige	10	2.5 x 1.25	9	6.5	Nicht Ideal für Fräse
Küchler	MDF	weiss furniert	8	2.8 x 2-07	11.9	6.8	Kleine Abmessungen
Küchler	MDF	weiss Furniert	16	4.2 x 2.0	17.4	13	Leicht zerbrechlich

Tabellle 4.1
Materialauswertung

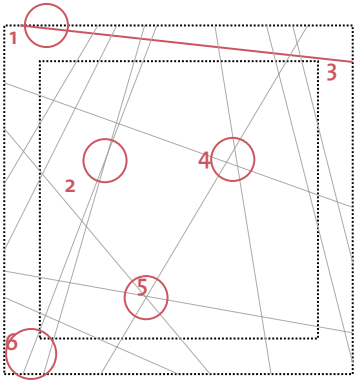


Bild 4.1
Konstruktive Kriterien

4.2 Konstruktive Kriterien:

1 Keine all zu flachen Winkel in der Nähe der Würfelkanten
2 Keine all zu flachen Winkel zwischen den Elementen
3 Keine Elemente vollständig in einer Kubusseite
4 Keine all zu kleine Paneelflächen
5 Nicht mehr als zwei Elemente sollen sich in einem Punkt Kreuzen
6 Element soll Innenseite und Aussenseite des Kubus nicht auf unterschiedliche Seite Kreuzen

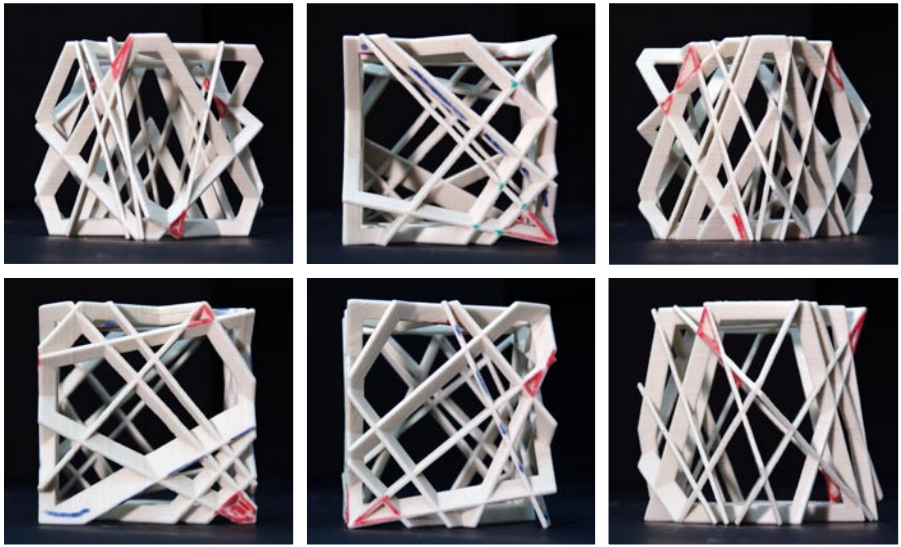


Bild 4.2
Für eine bessere Erfassung struktureller Schwachpunkte wurden 3D-Drucker Modelle angefertigt, in denen die kritischen Stellen markiert wurden.

4.3 Details

Wie schon bei den Konstruktionskriterien erwähnt, war ein wichtiges Kriterium die werkzeugfreie Assemblierung des Prototyps. Aus diesem Grunde wurde anstelle des Flansch-Details *(Bild4.3-1)*, der nur die Programmierung eines Details für die ganze Struktur bedingt hätte, für die in *Bild 4.3-2* abgebildete programmieraufwendigere Lösung optiert.
Diese Variante erfordert nähmlich ein weiteres Detail für die nötigen Elementverlängerungen *(Bild 4.3-3)*.

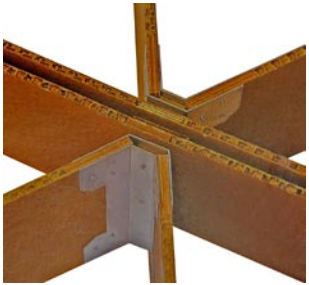


Bild4.3-1
Flansch-Detail

Bild 4.3-2

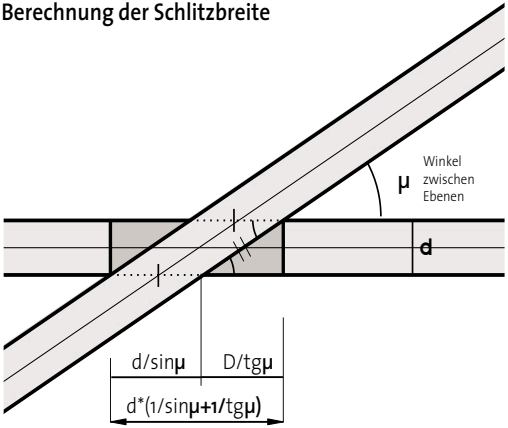
Bild 4.3-3
Realisierte Variante und Prototyp der Steckverbindung

4.4 Detail des Kreuzungspunktes zweier Elemente:

Damit dessen Fräspfad automatisch generiert werden konnte, bedingte dieses Detail die Integrierung folgender Parameter in den Programmcode:

- Materialstärke
- Winkel zwischen Bauelemente
- Fräskopf Durchmesser

Berechnung der Schlitzbreite



Zeichnen des Schlitzes mit Vektoren

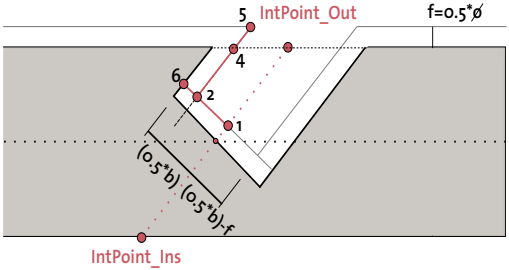
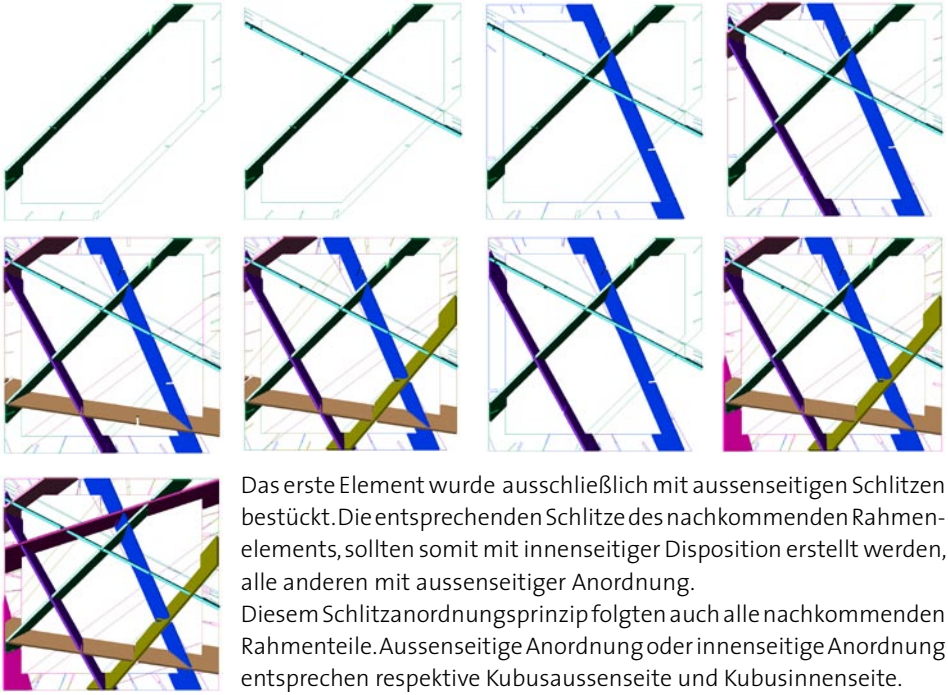


Bild 4.4
Berechnung der Schlitzbreite und zeichnen des Schlitzes mit Vektoren.

Der Schlitz wurde anhand von 3D-Vektoren gezeichnet.
Zuerst wurden die Punkte des Fräspfades berechnet und die Richtungsvektoren bestimmt.
Erst dann wurden die Daten in Form von Variablen an das Hauptprogramm DrawCubefromArray.mel übermittelt und gezeichnet.

Für die automatisierte Werkplanzeichnung dieses Details war des Weiteren die Bestimmung dessen Disposition, rahmeninnenseitige oder rahmenaussenseitige Anordnung von fundamentaler Bedeutung. Gleich bleibend der Generierungsreihenfolge der rahmenbestimmender Ebenen wurde die Generierungsreihenfolge der Details gesetzt.
Dies war so am einfachsten zu programmieren, stellte sich jedoch später bei der Nomenklatur der Details nicht als Optimum heraus, da eine der Montagereihenfolge entsprechenden Bezeichnung angebrachter gewesen wäre. Für die Anordnung der Schlitze war es jedoch von Vorteil.



Das erste Element wurde ausschließlich mit aussenseitigen Schlitzen bestückt. Die entsprechenden Schlitze des nachkommenden Rahmenelements, sollten somit mit innenseitiger Disposition erstellt werden, alle anderen mit aussenseitiger Anordnung.
Diesem Schlitzanordnungsprinzip folgten auch alle nachkommenden Rahmenteile. Aussenseitige Anordnung oder innenseitige Anordnung entsprechen respektive Kubusaussenseite und Kubusinnenseite.

Gleichzeitig zur Schlitzanordnung erfolgt in der Prozedur dessen Nomenklatur, die anfänglich Rahmennummer, Kubusseite und Schlitznummer enthalten sollte.
Kurz vor der Produktion, wurde jedoch aus gestalterischen Gründen vom Produktionsteam entschieden diese auf die in *Bild 4.4-1* dargestellte Weise von Hand zu ändern.

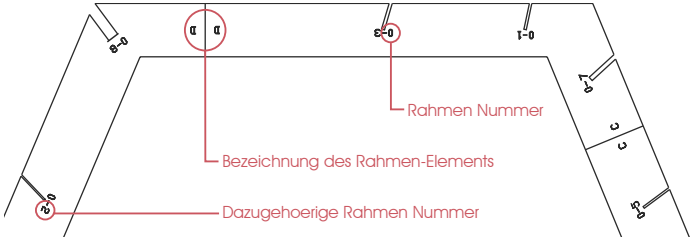


Bild 4.4-1
Die Bezeichnung der Rahmenelemente

Um die Montage zu vereinfachen sollte beiläufig untersucht werden, ob die Integrierung einer Sondervariante des Details möglich gewesen wäre. Die Sondervariante(*Bild 4.4-2*) hätte das standard Detail ersetzten sollen wann immer ein Schlitz in die entgegengesetzte Richtung zur Mehrheit der Schlitze orientiert gewesen wäre.
Die Implementierung dieser Anforderung in den Programmcode hätte in dieser Phase der Programmierung weitgreifende Reperkussionen in allen Prozeduren verursacht. Grund dafür war, dass die Koordinaten der Schnittpunkte in Arrays gespeichert worden sind und konnten deshalb nur nach dessen Position im Array abgerufen werden. In anderen Worten sind die Daten so strukturiert worden, dass man anhand der Arrayposition der sich in diesem Punkt schneidenden Geraden, auf die Arrayposition des Schnittpunktes schliessen kann und somit dessen Koordinaten abfragen. Mehrere Versuche dieses Problem zu umgehen, konnten keine zufriedenstellende Lösung herbeiführen.

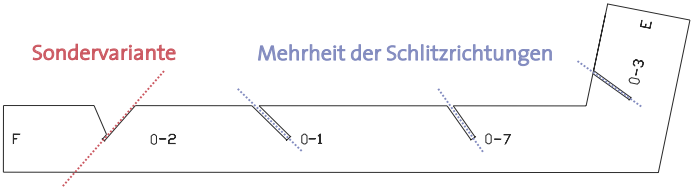
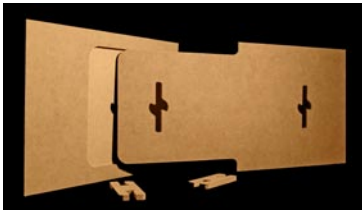


Bild 4.4-2
Die Sondervariante der Schlitzgeometrie

4.5 Detail der Elementverlängerungen



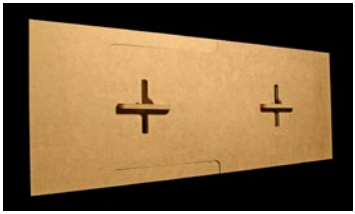


Bild 4.5-1
Prototyp Element-verlängerung

Die Rahmenelemente mussten aus mehreren Teilstücke angefertigt werden. Dabei waren die Abmessungen des Fräsenproduktionstisches und der Materialplatten entscheidend. Die Teilung des Rahmenelements hätte an Stellen stattfinden sollen, an denen genügend Platz vorhanden war um diese Steckverbindung einzufügen. Der Abstand zwischen zwei Schnittpunkte war das entscheidende Kriterium.
Anhand der Datenstrukturierung der Schnittpunkte wäre es sehr schwierig gewesen eine Analyseprozedur und den entsprechenden Algorithmus zu entwickeln um die best mögliche Anordnung des Details zu ermitteln und es dann an dieser Stelle einzufügen. Wegen der geringen Anzahl dieser Detailpunkte wurde ein Skript geschrieben, der in der Lage war anhand der Koordinaten des Einfügepunktes und der Rahmenbreite, das Detail an der richtigen Stelle und mit angepassten Abmessungen einzufügen.

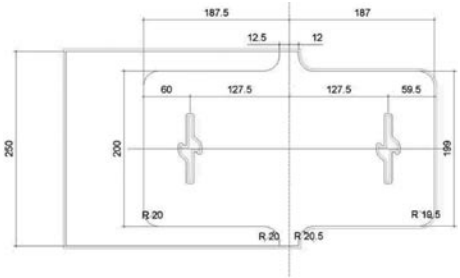


Bild 4.5-2
Variante Element-verlängerung

Für das Prototyp kam jedoch ein Plug-in zum Einsatz, der von Ebnöther If entwickelt wurde. Dieses wurde in VectorScript programmiert und war ausserdem fähig innerhalb der Geometrie die Fräspfade zu zeichnen. Die Fräspfade sind um die zu realisierende Vertiefung durch die CNC-gesteuerte Fräse zu bewirken nicht notwendig.
Surfcam, das Programm das den G-Code File für die Fräse schreibt, benötigt dafür nur eine Kontur. Durch die präzise Angabe des Fräspfades kann jedoch die Maschine dieselbe Operation schneller durchführen da dessen Fräskopf keine unnötigen Pfadwege zurückzulegen braucht.

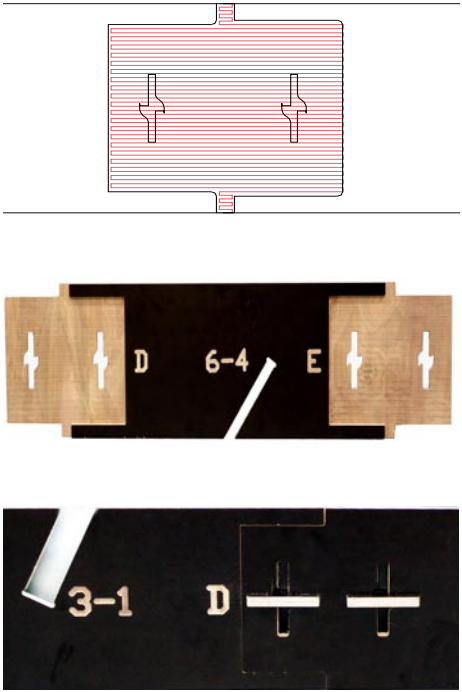
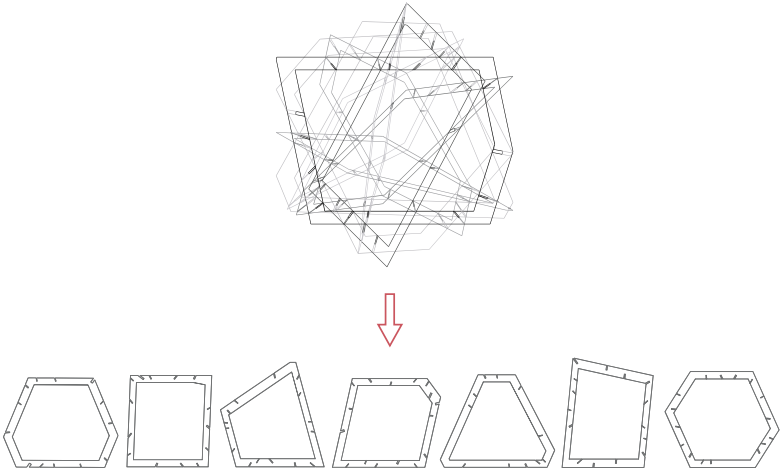


Bild 4.5-3
Fräspfaderzeugung durch Plug-in

4.6 Automatisierte Werkplanung

Input:
-Wandstärke
-Materialstärke

Da jedes Rahmenelement nie parallel zu den Würfelkanten liegt repräsentiert die Wandstärke eher eine fiktive Grösse.
Sämtliche Rahmen und die entworfenen Steckverbindungen sollen automatisch aus Platten gefräst werden. Dieser Produktionsweg erfordert die Erstellung eines Zuschnittsplans. Hier setzt ein weiteres Programmskript ein. Es übernimmt das dreidimensionale optimierte Datenmodell aus der vorhergehenden Phase und projiziert jeden Stegrahmen mit den zugehörigen Schlitzverbindungen massstabsgetreu in eine Zeichnungsebene. Diese Zeichnung dient als Basis für die Werkplanung. Die format- und transportbedingten Steckverbindungen wurden nachträglich ergänzt, wie es in den Detailbeschreibungen erleutert worden ist.



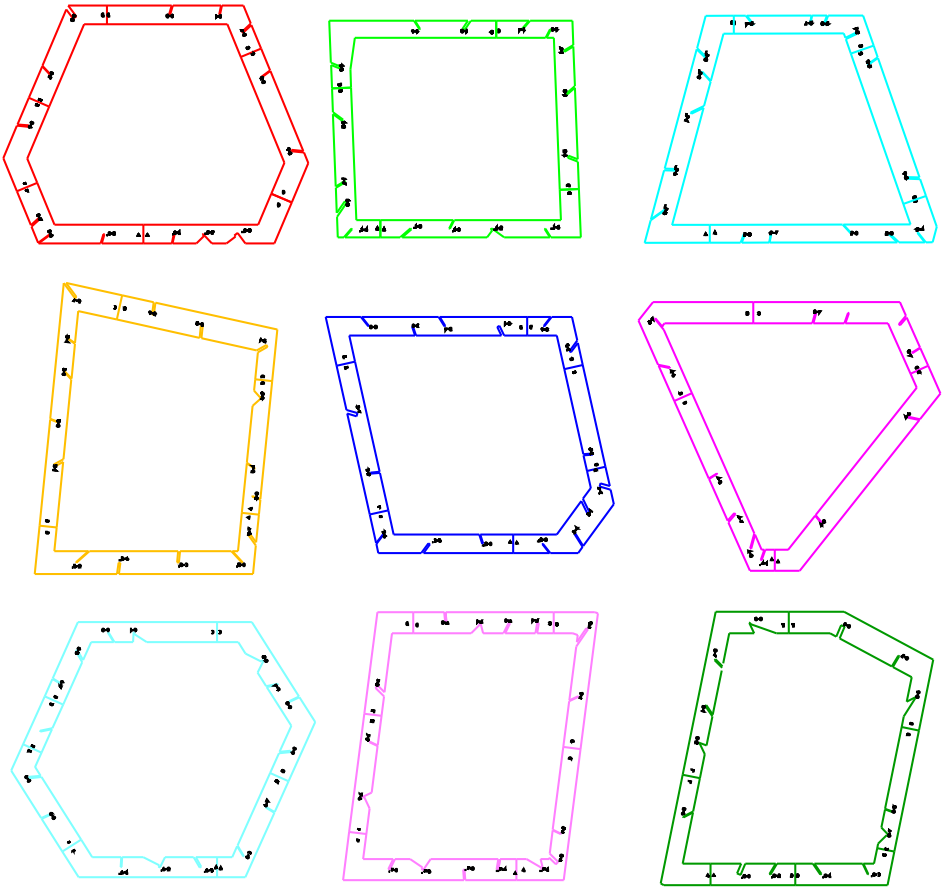


Bild 4.6
Basis für die Werkplanung

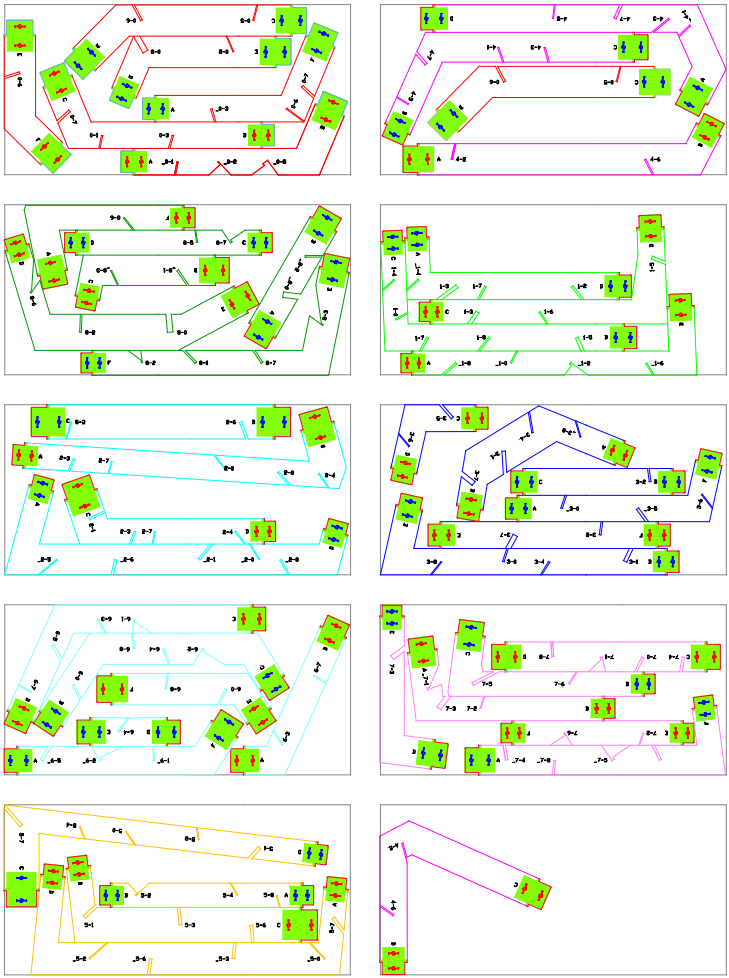
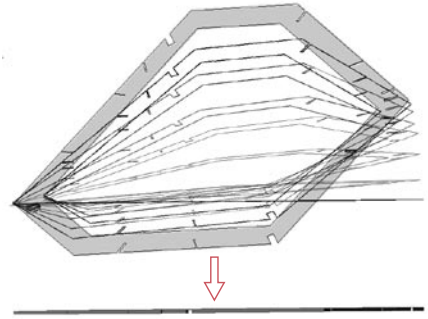


Bild 4.6-1
Werkpläne

Projektion der Stegrahmen mit den dazugehörigen Schlitzverbindungen in einer Zeichnungsebene:
In Maya kann man, im Gegensatz zu AutoCad, die Orientierung des verwendeten Koordinatensystem nicht ändern. Man kann jedoch über Umwege eine Kamera senkrecht auf das Element richten. Die Koordinaten und die Orientierung der Kamera können als *Attribute* Werte abgefragt werden und anhand dessen Zahlenwerte können die Winkel berechnet werden (ein Winkel für jede Koordinatensystem-Achse) um denen das Element gedreht werden soll damit es Flach im *3D World coordinate system* liegt. Anschliessend können mit einer einfachen *loop* alle Z-Koordinaten der in der Zeichnung vorhandenen Elementen abgefragt werden und auf null gesetzt werden.
Nach diesem Schritt kommt der *Extrude.mel* Script zum Einsatz. Dieser Schritt verwandelt das *wireframe* Model in einem *3D- Surface* Körper.

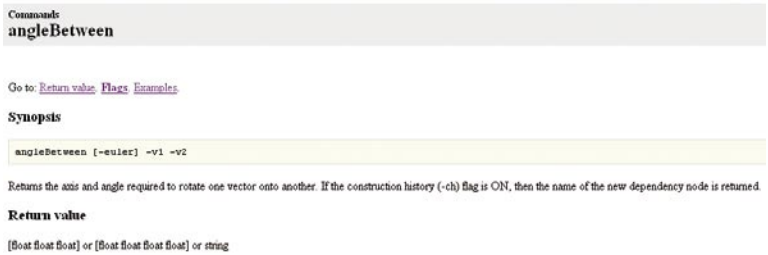
Planarize .mel



Extrude .mel

3D-Surface Modell

Der MEL-Befehl *angleBetween*, der erst nach Fertigstellung des *Planarize.mel* Skript in der üppigen *MEL Comand Referenz* entdeckt worden ist, hätte diesen Arbeitsschritt durch den *Flag -euler* , (Euler Winkel) möglicherweise vereinfachen können. Die Euler Winkeloption liefert durch die Eingabe zweier Vektoren drei auf das Koordinatensystem bezogenen Rotationswinkel um dessen x, y und z Achsen.



Commands
angleBetween

Go to: Return value, Flags, Examples.

Synopsis

```
angleBetween [-euler] -v1 -v2
```

Returns the axis and angle required to rotate one vector onto another. If the construction history (-ch) flag is ON, then the name of the new dependency node is returned.

Return value

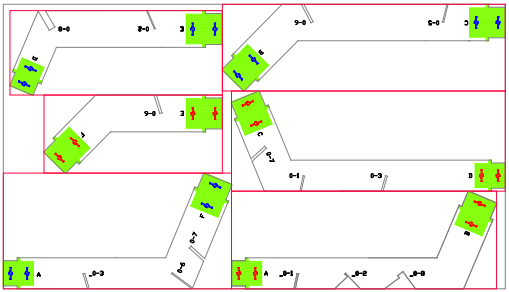
[float float float] or [float float float float] or string

Bild 4.6-2
MEL Command Reference

Beispiel:
angleBetween -euler -v1 0.0 1.0 2.0 -v2 1.0 2.0 0.0;
// Result: -63.434949 16.60155 -26.565051 //

In der gleichen Prozedur werden alle Rahmenelemente auf separaten Layer und im positiven Koordinatenbereich verschoben. Diese Operation war für die Software der Fräse wichtig.
Die Anordnung der Rahmenelemente auf einem einzigen Layer, hätte auf einfacher Weise mit dem, aus der Experimentierphase der MEL-Programmierung vertrautem Befehl *objectCenter* erfolgen können. Dieser Befehl gibt den Schwerpunkt der *Boundingbox* zurück. Eine solche Disposition wäre jedoch für die CNC-Produktion nicht zweckgemässig gewesen.

Anordnung anhand Boundingbox Befehl



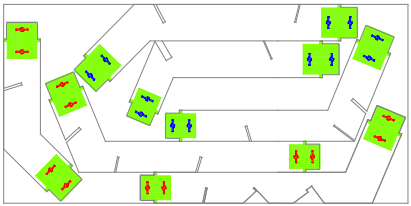
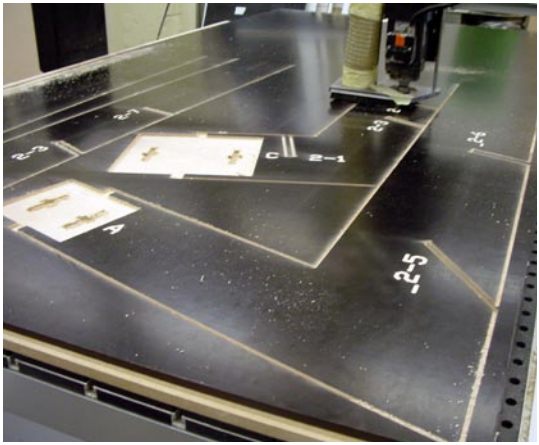
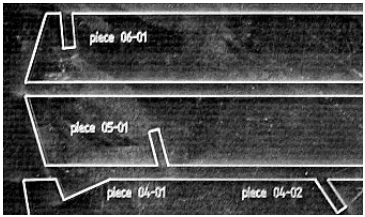


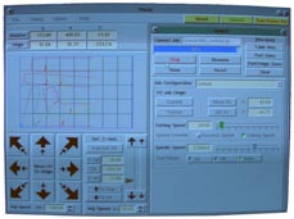
Bild 4.6-3
Von Hand auf Materialplatte angeordnete Rahmenteile

4.7 Computer Aided Manufacturing

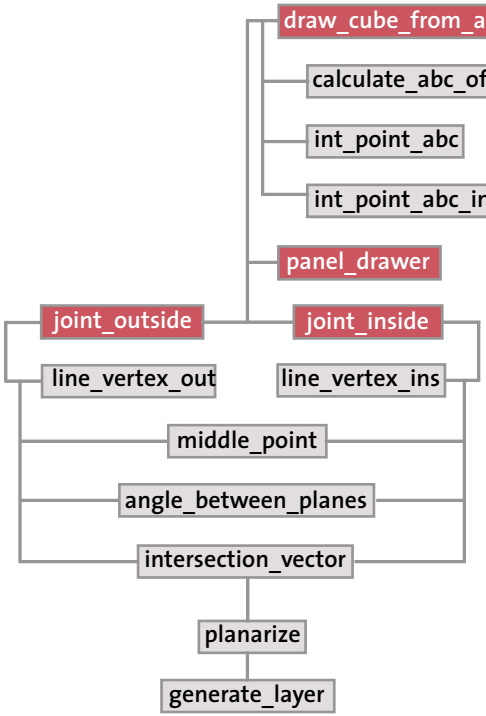
Die beim Detail der Elementverlängerungen beschriebenen Problematiken *(Seite 46)* haben die Erstellung eines nachbearbeitungsfreien Zuschnittsplans in der zur Verfügung stehenden Zeit nicht ermöglicht. Es ist dabei zu präzisieren das es nicht unmöglich gewesen wäre auch diesen Arbeitsschritt durch Mel-Skripte zu automatisieren. Bei der geringen Anzahl an Elementen entschied sich das Produktionsteam jedoch die Disposition der Elemente für den Zuschnittplan von Hand vorzubereiten.
Als Alternative hätte BySoft (Software für die ETH eigene Laserschneidemaschine) eingesetzt werden können. Diese Software ist in der Lage eine grosse Anzahl von Geometrien in Sekundenschnelle in einem materialsparenden Zuschnittplan für die CNC-Maschine einzuordnen.
Die auf diesem Wege präparierten Werkpläne können anschliessend mit dem Programm SurfCam in einen maschinenverständlichen Code übersetzt und an die CNC-Fräse übermittelt werden. Die im Plan enthaltenen Linien werden von der Maschine als Fräspfade interpretiert.







5.0 MEL-scripts



⇨ **Hauptprozedur**

⇨ **Konstruktions Stylesheet**
Tragende Paneele

⇨ **Konstruktions Stylesheet**
des Prototyp NDS2004

Bild 5.0
Prozeduren der automatisierten Werkplanung

Ausgehend von den unten aufgelisteten Arrays, in denen die optimierten Daten (A-,B-,C- und D-Koeffizienten) der Kubus schneidenden Ebenen enthalten, wurden anhand der in *Bild 5.0* abgebildeten Prozeduren die Automatisierten Werkpläne erstellt.
Im folgenden wird nur ein Auszug der Hauptprozedur und der beiden Stylesheets-Prozeduren wiedergegeben.

```
$Aplane=
{-0.8464384105,
0.595807,
0.5461631397,
-0.7931392819,
-0.7075098671,
0.153747,
0.4651374461,
0.858724,
0.395076 };

$Bplane=
{ 0.5552307892,
-0.097311,
0.4829963586,
0.4004663735,
0.8736476983,
-0.950545,
0.3839430073,
-1.140121,
-0.981543};

$Cplane=
{ -0.8580282306,
-1.14713,
-0.2349151238,
0.4380900804,
-0.9447640897,
-1.036641,
0.391744454,
-0.246476,
1.199279 };

$Dplane=
{ 0.4899769628,
0.530634,
-0.5304495668,
0.0374235896,
0.6483634222,
1.036054,
-0.6155449899,
0.243194,
-0.22325 };
```

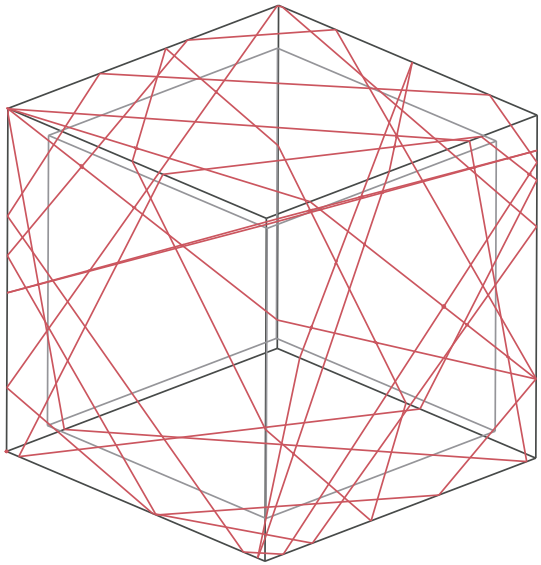
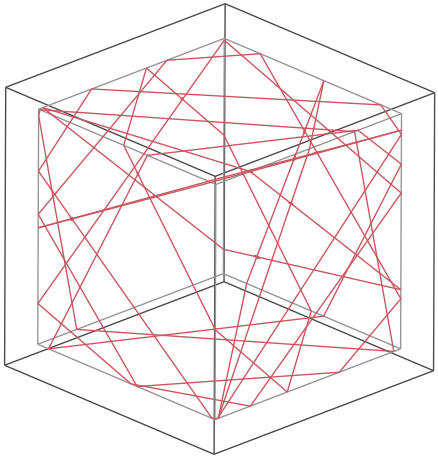


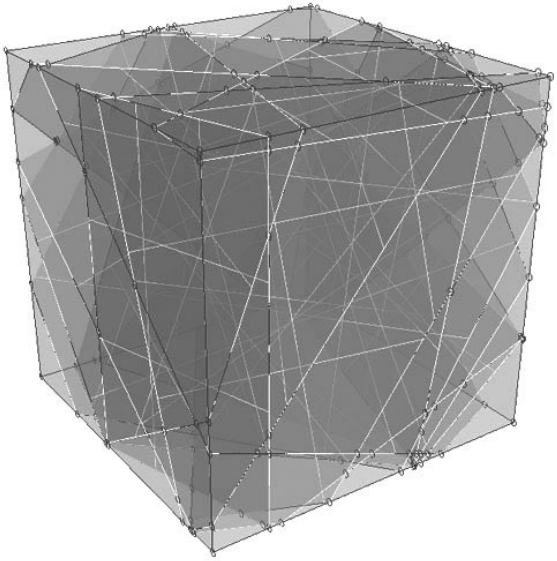
Bild 5.0-1
A, B, C, D Koeffiziente der Ebenen-Gleichung gespeichert in Arrays

5.1 Draw_cube_from_array_with_ cuts.mel



```
global proc drawCubeFromArrayCuts (float $PlaneArrayA[],float $PlaneArrayB[],float $PlaneArrayC[],
float $PlaneArrayD[], float $wow, float $d, float $f)
int $counter = -1; global int $squareCounter; $squareCounter = 1; global int $circleCounter; $circleCounter = 1;
global float $Aline1[]; global float $Bline1[]; global float $Cline1[]; global int $Pline1[];
clear ($Aline1); clear ($Bline1); clear ($Cline1); clear ($Pline1);
global float $Aline2[]; global float $Bline2[]; global float $Cline2[]; global int $Pline2[];
clear ($Aline2); clear ($Bline2); clear ($Cline2); clear ($Pline2);

global float $Aline1ins[]; global float $Bline1ins[]; global float $Cline1ins[]; global int $Pline1ins[];
clear ($Aline1ins); clear ($Bline1ins); clear ($Cline1ins); clear ($Pline1ins);
global float $Aline2ins[]; global float $Bline2ins[]; global float $Cline2ins[]; global int $Pline2ins[];
clear ($Aline2ins); clear ($Bline2ins); clear ($Cline2ins); clear ($Pline2ins);

$Aline1={0,1,0,1}; $Bline1={1,0,1,0}; $Cline1={0,-1,-1,0}; $Pline1={-1,-1,-1,-1};
$Aline2={0,1,0,1}; $Bline2={1,0,1,0}; $Cline2={0,-1,-1,0}; $Pline2={-1,-1,-1,-1};

$Aline1ins={0,1,0,1};  $Bline1ins={1,0,1,0};  $Cline1ins={((-1)*$WoW),((-1)*$WoW1),((-1)*$WoW1),((-1)*$WoW)};  $Pline1ins={-
1,-1,-1,-1};
$Aline2ins={0,1,0,1}; $Bline2ins={1,0,1,0}; $Cline2ins={((-1)*$WoW),((-1)*$WoW1),((-1)*$WoW1),((-1)*$WoW)}; $Pline2ins={-
1,-1,-1,-1};
```

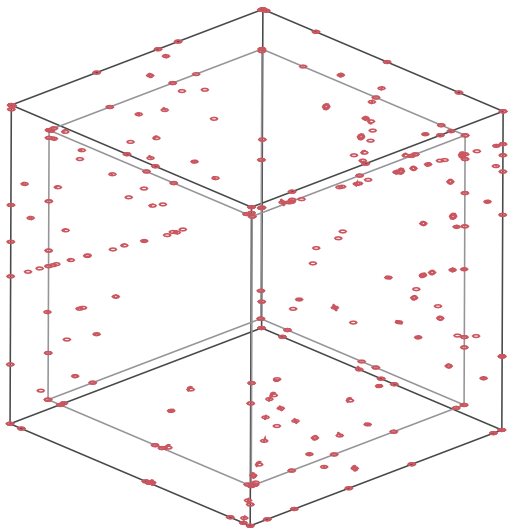
```
global int $numOfLineSide1;
global int $numOfLineSide1ins;
global int $numOfLineSide2;
global int $numOfLineSide2ins;

$numOfLineSide1 = 4;
$numOfLineSide1ins=4;
$numOfLineSide2 = 4;
$numOfLineSide2ins=4;

int $howManyPlanes= size ($PlaneArrayA);//how many times should a plane be drawn

for ($counter=0; $counter<$howManyPlanes; $counter=$counter+1)
{
//points used in calculating the y=Ax+B of lines on sides;
global float $a1;
global float $b1;
global float $a2;
global float $b2;

//PLANE - Ax+By+Cz+D = 0;
float $A = $PlaneArrayA[$counter];
float $B = $PlaneArrayB[$counter];
float $C = $PlaneArrayC[$counter];
float $D = $PlaneArrayD[$counter];

int $sID = 1; int $sIDins = 1;

//side 1
//$R11 0 0; 1 0 $R12; $R13 0 1; 0 0 $R14;

int $sideNR=1;

global float $R11;
$R11 = (-1)*((($B*0)+($C*0)+$D)/$A);
global float $R12;
$R12 = (-1)*((($A*1)+($B*0)+$D)/$C);
global float $R13;
$R13 = (-1)*((($B*0)+($C*1)+$D)/$A);
global float $R14;
$R14 = (-1)*((($A*0)+($B*0)+$D)/$C);


if (($R11 > 0)&&($R11 < 1))
```



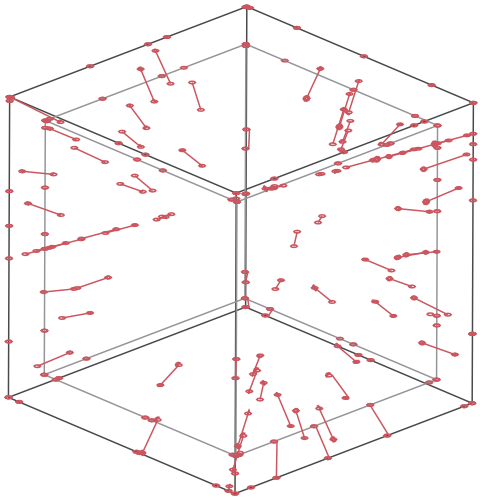
```
                {
                if ( ($R12 > 0)&&($R12 < 1) )
                        {curve -d 1 -p $R11 0 0 -p 1 0 $R12 -k 0 -k 1 -n ("l"+$sID+"p"+$counter);
                        $sID += 1;

//remember always, that a is sth like x and b is sth like y
$a1 = $R11;
$b1 = 0;
$a2 = 1;
$b2 = $R12;

calculateABCofLine $a1 $b1 $a2 $b2 $numOfLineSide1 $counter $sideNR 0;
$numOfLineSide1+=1;
}
                else if ( ($R13 > 0)&&($R13 < 1) )
                        {curve -d 1 -p $R11 0 0 -p $R13 0 1 -k 0 -k 1 -n ("l"+$sID+"p"+$counter) ;
                        $sID += 1;
$a1 = $R11;
$b1 = 0;
$a2 = $R13;
$b2 = 1;

//calculateABCofLineSide1 $R11 0 $R13 1 $numOfLineSide1 $counter;
calculateABCofLine $a1 $b1 $a2 $b2 $numOfLineSide1 $counter $sideNR 0;

$numOfLineSide1+=1;
}
                else if ( ($R14 > 0)&&($R14 < 1) )
                        {curve -d 1 -p $R11 0 0 -p 0 0 $R14 -k 0 -k 1 -n ("l"+$sID+"p"+$counter) ;
                        $sID += 1;
$a1 = $R11;
$b1 = 0;
$a2 = 0;
$b2 = $R14;

//calculateABCofLineSide1 $a1 $b1 $a2 $b2 $numOfLineSide1 $counter;
calculateABCofLine $a1 $b1 $a2 $b2 $numOfLineSide1 $counter $sideNR 0;

$numOfLineSide1+=1;
}
                }

...
                        if ( ($R13 > 0)&&($R13 < 1) )
```

5.2 Joint_Outside & Joint_Inside

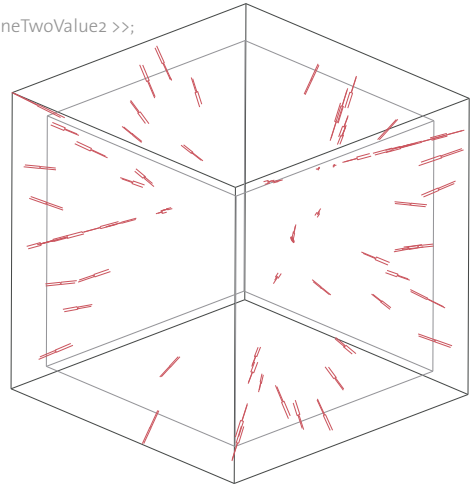
```
global proc middle_point( float $XintP, float $YintP, float $ZintP, float $XintPins, float $YintPins, float $ZintPins )
{
/*
Returns the mid point between two 3D points, p1 and p2.
*/

global float $midPt[3];
$midPt[0] = ($XintP+$XintPins)/2.0;  $midPt[1] = ($YintP+$YintPins)/2.0; $midPt[2] = ($ZintP+$ZintPins)/2.0;
}

global proc joint_Outside (int $PointNr, int $PointNrins, float $f, float $d)
{
global float $XintPoint2[]; global float $YintPoint2[];global float $ZintPoint2[];
global float $XintPoint2ins[]; global float $YintPoint2ins[];global float $ZintPoint2ins[];

//taking values from arrays
float $XintPOut=$XintPoint2[$PointNr];
float $YintPOut=$YintPoint2[$PointNr];
float $ZintPOut=$ZintPoint2[$PointNr];

float $XintPins=$XintPoint2ins[$PointNrins];
float $YintPins=$YintPoint2ins[$PointNrins];
float $ZintPins=$ZintPoint2ins[$PointNrins];

//Vector of middle line
float $Xvalue=($XintPOut-$XintPins);
float $Yvalue=($YintPOut-$YintPins);
float $Zvalue=($ZintPOut-$ZintPins);

global float $Aplane[]; global float $Bplane[];
global float $Cplane[]; global float $Dplane[];

//numbers of lines wich make a point
global int $firstLineIntPoint2[];
global int $secLineIntPoint2[];
global int $Pline2[];

//middle line values stored in unit vector
vector $InsOut=<<$Xvalue, $Yvalue, $Zvalue >>;
vector $InsOutunit=unit ($InsOut);
```



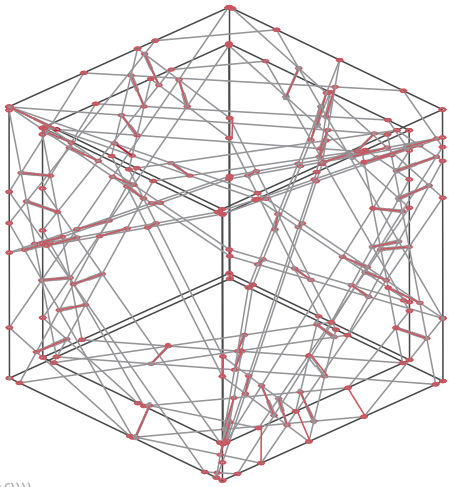
```
// Number of the plane
int $firstLine=$firstLineIntPoint2[$PointNr];
int $firstPlane=$Pline2[$firstLine];

int $secLine=$secLineIntPoint2[$PointNr];
int $secPlane=$Pline2[$secLine];

//creating Normalvector for first Plane
vector $firstNormal=<<$Aplane[$firstPlane],
$Bplane[$firstPlane], $Cplane[$firstPlane] >>;

vector $crossproduct1=cross($InsOut, $firstNormal);
vector $crossproduct1=unit ($crossproduct1);

float $Xcp1=$crossproduct1.x;
float $Ycp1=$crossproduct1.y;
float $Zcp1=$crossproduct1.z;

//creating Normalvector for second Plane
vector $secNormal=<<$Aplane[$secPlane],
$Bplane[$secPlane], $Cplane[$secPlane] >>;

vector $crossproduct2=cross($InsOut, $secNormal);
vector $crossproduct2=unit ($crossproduct2);

float $Xcp2=$crossproduct2.x;
float $Ycp2=$crossproduct2.y;
float $Zcp2=$crossproduct2.z;

//calculating middlepoint of middle line
middle_point $XintPOut $YintPOut $ZintPOut $XintPins $YintPins $ZintPins;
global float $midPt[] ;
global float $xedge1;global float $yedge1; global float $zedge1; global float $xedge2; global float $yedge2;
global float $zedge2;

LineVertexOut $firstPlane 2;
$PointOneLineOut={$xedge1,$yedge1, $zedge1}; $PointTwoLineOut={$xedge2,$yedge2, $zedge2};

float $XOneTwoValue=($PointOneLineOut[0] - $PointTwoLineOut[0]);
float $YOneTwoValue=($PointOneLineOut[1] - $PointTwoLineOut[1]);
float $ZOneTwoValue=($PointOneLineOut[2] - $PointTwoLineOut[2]);

vector $Border=<< $XOneTwoValue, $YOneTwoValue, $ZOneTwoValue >>;
vector $Borderunit=unit ($Border);
```



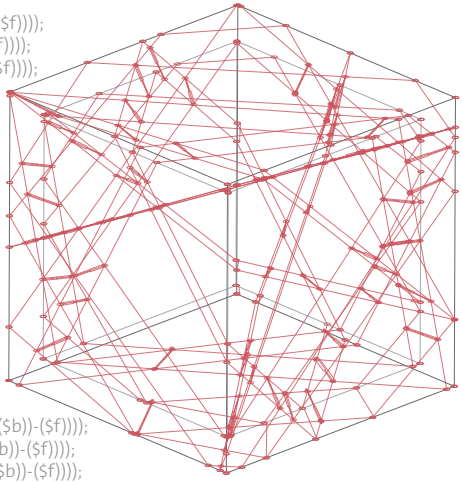
```
float $XBunit=$Borderunit.x;
float $YBunit=$Borderunit.y;
float $ZBunit=$Borderunit.z;

//Vertex Second side (aus last_linevertex_global_v2.mel)

global float $xedge1;
global float $yedge1;
global float $zedge1;
global float $xedge2;
global float $yedge2;
global float $zedge2;

LineVertexOut $secPlane 2;

$PointOneLineOut2={$xedge1, $yedge1, $zedge1};
$PointTwoLineOut2={$xedge2, $yedge2, $zedge2};

float $XOneTwoValue2=($PointOneLineOut2[0] - $PointTwoLineOut2[0]);
float $YOneTwoValue2=($PointOneLineOut2[1] - $PointTwoLineOut2[1]);
float $ZOneTwoValue2=($PointOneLineOut2[2] - $PointTwoLineOut2[2]);

vector $Border2=<< $XOneTwoValue2, $YOneTwoValue2, $ZOneTwoValue2 >>;
vector $Borderunit2=unit ($Border2);

float $XBunit2=$Borderunit2.x;
float $YBunit2=$Borderunit2.y;
float $ZBunit2=$Borderunit2.z;

//Angle value
angle_between_planes $firstPlane $secPlane;
global float $angp_m_n;
print „angle“;
print $angp_m_n;

//firstdirection
float $XFPD1=$InsOutunit.x;
float $YFPD1=$InsOutunit.y;
float $ZFPD1=$InsOutunit.z;

//finding point coordinates
```



```
//1point
global float $FirstPointDir1[];
$FirstPointDir1[0]=$midPt[0]+($XFPD1*$f);
$FirstPointDir1[1]=$midPt[1]+($YFPD1*$f);
$FirstPointDir1[2]=$midPt[2]+($YFPD1*$f);

//2point
float $b=(($d/sind ($angp_m_n))+($d*atan ($angp_m_n)));
print „b“;
print $b;

global float $SecPointDir1[];
$SecPointDir1[0]=($FirstPointDir1[0]-(($Xcp1)*((0.5*($b))-($f))));
$SecPointDir1[1]=($FirstPointDir1[1]-(($Ycp1)*((0.5*($b))-($f))));
$SecPointDir1[2]=($FirstPointDir1[2]-(($Zcp1)*((0.5*($b))-($f))));

//3point
global float $ThirdPointDir1[];
$ThirdPointDir1[0]=$SecPointDir1[0]-($XFPD1*0.2);
$ThirdPointDir1[1]=$SecPointDir1[1]-($YFPD1*0.2);
$ThirdPointDir1[2]=$SecPointDir1[2]-($ZFPD1*0.2);

//4.point
global float $fourtPointDir1[];
$fourtPointDir1[0]=($ThirdPointDir1[0]-(($XBunit)*((0.5*($b))-($f))));
$fourtPointDir1[1]=($ThirdPointDir1[1]-(($YBunit)*((0.5*($b))-($f))));
$fourtPointDir1[2]=($ThirdPointDir1[2]-(($ZBunit)*((0.5*($b))-($f))));


//5.point
global float $fiftPointDir1[];
$fiftPointDir1[0]=($fourtPointDir1[0]+(($XFPD1)*(0.2)));
$fiftPointDir1[1]=($fourtPointDir1[1]+(($YFPD1)*(0.2)));
$fiftPointDir1[2]=($fourtPointDir1[2]+(($ZFPD1)*(0.2)));

//calculating Points For second Plane
//2point
global float $SecPointDir1S2[];
$SecPointDir1S2[0]=($FirstPointDir1[0]-(($Xcp2)*((0.5*($b))-($f))));
$SecPointDir1S2[1]=($FirstPointDir1[1]-(($Ycp2)*((0.5*($b))-($f))));
$SecPointDir1S2[2]=($FirstPointDir1[2]-(($Zcp2)*((0.5*($b))-($f))));
```



```
//2point!!!
global float $SecPointDir2S2[];

$SecPointDir2S2[0]=($FirstPointDir1[0]+(($Xcp2)*((0.5*($b))-($f))));
$SecPointDir2S2[1]=($FirstPointDir1[1]+(($Ycp2)*((0.5*($b))-($f))));
$SecPointDir2S2[2]=($FirstPointDir1[2]+(($Zcp2)*((0.5*($b))-($f))));

//3point
global float $ThirdPointDir1S2[];
$ThirdPointDir1S2[0]=$SecPointDir1S2[0]-($XFPD1*0.2);
$ThirdPointDir1S2[1]=$SecPointDir1S2[1]-($YFPD1*0.2);
$ThirdPointDir1S2[2]=$SecPointDir1S2[2]-($ZFPD1*0.2);

//3point!!!
global float $ThirdPointDir2S2[];
$ThirdPointDir2S2[0]=$SecPointDir2S2[0]-($XFPD1*0.2);
$ThirdPointDir2S2[1]=$SecPointDir2S2[1]-($YFPD1*0.2);
$ThirdPointDir2S2[2]=$SecPointDir2S2[2]-($ZFPD1*0.2);

global float $FourtPointDir1S2[];

$FourtPointDir2S2[0]=($ThirdPointDir2S2[0]+(($Xcp2)*((0.5*($b))-($f))));
$FourtPointDir2S2[1]=($ThirdPointDir2S2[1]+(($Ycp2)*((0.5*($b))-($f))));
$FourtPointDir2S2[2]=($ThirdPointDir2S2[2]+(($Zcp2)*((0.5*($b))-($f))));

//drawing curves
curve -d 1 -p $FirstPointDir1[0] $FirstPointDir1[1] $FirstPointDir1[2] -p $SecPointDir1[0] $SecPointDir1[1] $SecPointDir1[2]
-k 0 -k 1 -n pr2ima;
curve -d 1 -p $SecPointDir1[0] $SecPointDir1[1] $SecPointDir1[2] -p $ThirdPointDir1[0] $ThirdPointDir1[1]
$ThirdPointDir1[2] -k 0 -k 1 -n se2conda;
//curve -d 1 -p $ThirdPointDir1[0] $ThirdPointDir1[1] $ThirdPointDir1[2] -p $fourtPointDir1[0] $fourtPointDir1[1]
$fourtPointDir1[2] -k 0 -k 1 -n terza;
curve -d 1 -p $fourtPointDir1[0] $fourtPointDir1[1] $fourtPointDir1[2] -p $fiftPointDir1[0] $fiftPointDir1[1] $fiftPointDir1[2]
-k 0 -k 1 -n mitt2ellinie;

curve -d 1 -p $FirstPointDir1[0] $FirstPointDir1[1] $FirstPointDir1[2] -p $SecPointDir2[0] $SecPointDir2[1] $SecPointDir2[2]
-k 0 -k 1 -n pri2adir2;
curve -d 1 -p $SecPointDir2[0] $SecPointDir2[1] $SecPointDir2[2] -p $ThirdPointDir2[0] $ThirdPointDir2[1]
$ThirdPointDir2[2] -k 0 -k 1 -n seco2ndadir2;
curve -d 1 -p $ThirdPointDir2[0] $ThirdPointDir2[1] $ThirdPointDir2[2] -p $ThirdPointDir1[0] $ThirdPointDir1[1]
$ThirdPointDir1[2] -k 0 -k 1 -n te2rzabo;
```



```
//For second Plane for second Plane
curve -d 1 -p $FirstPointDir1[0] $FirstPointDir1[1] $FirstPointDir1[2] -p $SecPointDir1S2[0] $SecPointDir1S2[1]
$SecPointDir1S2[2] -k 0 -k 1 -n pri2maS2;
curve -d 1 -p $SecPointDir1S2[0] $SecPointDir1S2[1] $SecPointDir1S2[2] -p $ThirdPointDir1S2[0] $ThirdPointDir1S2[1]
$ThirdPointDir1S2[2] -k 0 -k 1 -n sec2ondaS2;
curve -d 1 -p $FirstPointDir1[0] $FirstPointDir1[1] $FirstPointDir1[2] -p $SecPointDir2S2[0] $SecPointDir2S2[1]
$SecPointDir2S2[2] -k 0 -k 1 -n terz2aS2;
curve -d 1 -p $SecPointDir2S2[0] $SecPointDir2S2[1] $SecPointDir2S2[2] -p $ThirdPointDir2S2[0] $ThirdPointDir2S2[1]
$ThirdPointDir2S2[2] -k 0 -k 1 -n quar2taS2;
curve -d 1 -p $ThirdPointDir2S2[0] $ThirdPointDir2S2[1] $ThirdPointDir2S2[2] -p $ThirdPointDir1S2[0] $ThirdPointDir1S2[1]
$ThirdPointDir1S2[2] -k 0 -k 1 -n qu2artaS2;

            }
```

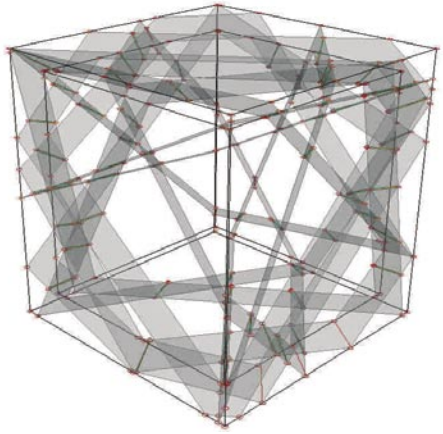


Bild 5.2-1
X-Ray Ansicht mit Schlitze

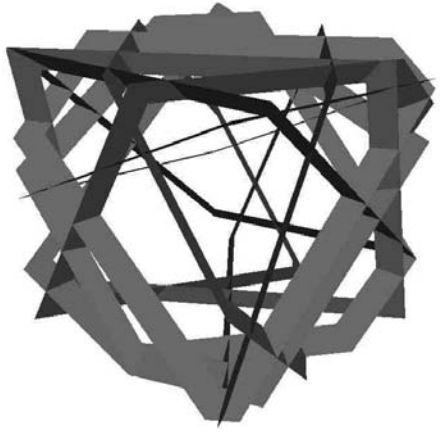
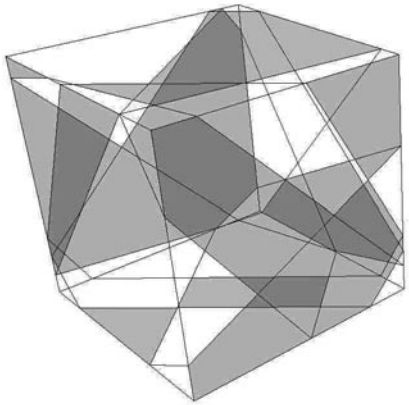


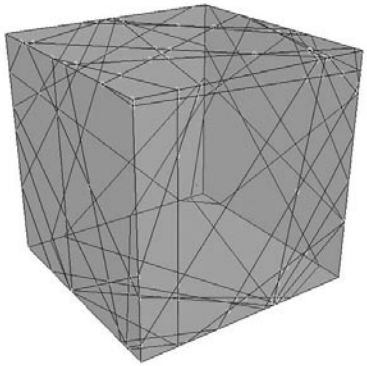
Bild 5.2-1
X-Ray Ansicht mit Schlitze

5.3 Panel_drawer

```
global proc PanelDrawer1 (){

global float $XintPoint1[];
global float $YintPoint1[];
global float $ZintPoint1[];
global float $Aline1[];

global int $panelCounter;
$panelCounter = 1;

global int $tempPanelCounter;
$tempPanelCounter = 0;
global float $XintPoint1[];

int $numer = size ($XintPoint1);
global int $panel1side1[];
clear ($panel1side1);
```



```
for ($i=0; $i<$numer; $i=$i+1){
if(($XintPoint1[$i]==0)&&($YintPoint1[$i]==0)&&($ZintPoint1[$i]==0)){$panel1side1[0]=$i;}
if(($XintPoint1[$i]==1)&&($YintPoint1[$i]==0)&&($ZintPoint1[$i]==0)){$panel1side1[1]=$i;}
if(($XintPoint1[$i]==1)&&($YintPoint1[$i]==0)&&($ZintPoint1[$i]==1)){$panel1side1[2]=$i;}
if(($XintPoint1[$i]==0)&&($YintPoint1[$i]==0)&&($ZintPoint1[$i]==1)){$panel1side1[3]=$i;}
}
//print $panel1side1;
global int $amoutExistingOfPanels;
$amoutExistingOfPanels = 1;
global int $amountOfLines;
$amountOfLines = size ($Aline1);
for ($k=4; $k<$amountOfLines; $k=$k+1){

for ($i=1; $i<=$panelCounter; $i=$i+1){

string $st = „global int $panel“+$i+“side1[]“;
eval $st;
global int $panelAg[];
clear ($panelAg);
string $st = „$panelAg = $panel“+$i+“side1“;
eval $st;
```



```
makingTwoPanelsWithLine1 $panelAg $k $i $amoutExistingOfPanels;

$amoutExistingOfPanels = $tempPanelCounter;
}
$panelCounter=$amoutExistingOfPanels;

}

//print „panelCounter“;
//print $panelCounter;

}
```

5.4 Stylesheet-Studien

Die Programmierphase lief Hand in Hand mit der Entwicklung der Konstruktion, es wurden somit mehre Stylesheets erstellt. Im folgenden sind einige Beispiel-Studien abgebildet.

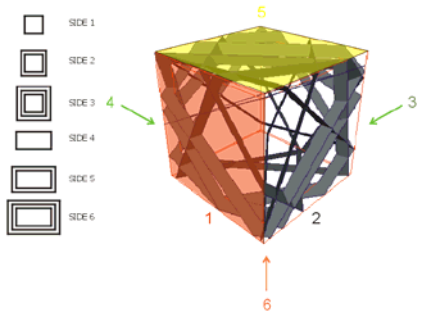


Bild 5.4-1
Elementbezeichnung mit symbolen

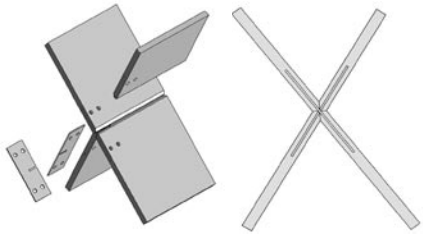
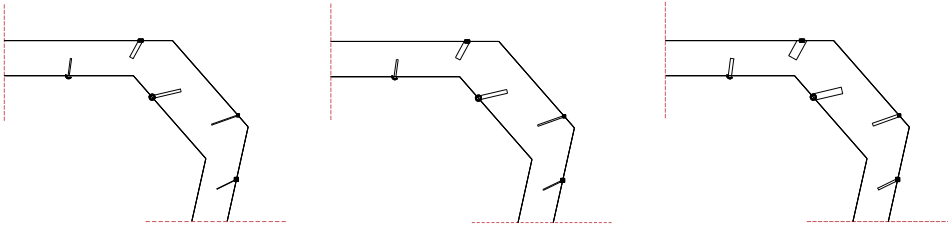


Bild 5.4-1
Laschen-Steckverbindung



8mm milling

10mm lasering

16mm milling

Bild 5.4-2
Stylesheets für Produktion mit Laser und Fräse

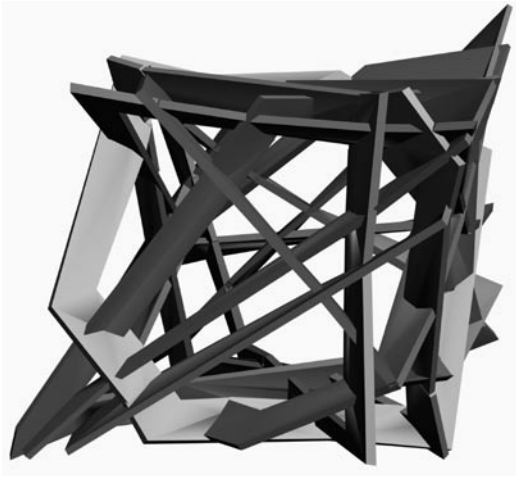


Bild 5.4-1
Deformations-Studie "parametric lattice-deformer"

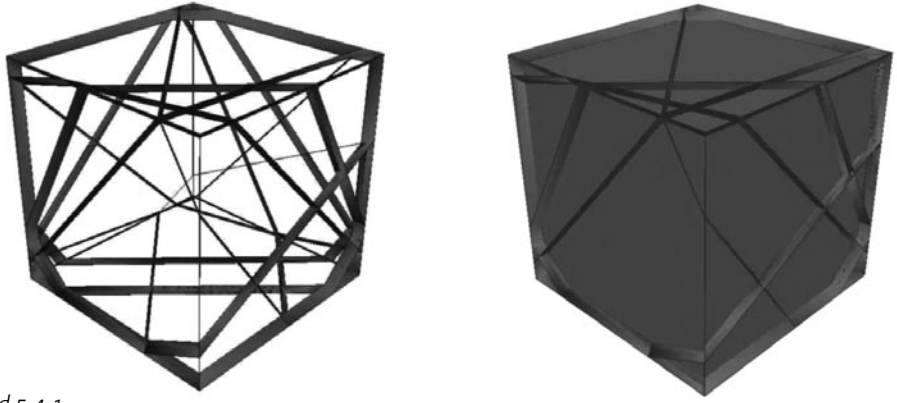


Bild 5.4-1
Variante des Panel-Stylesheets

6.0 Beispiele aus der Baupraxis

6.1 Beijing Water Cube the IT challenge

Stuart Bull (Senior 3D modeler) und Steve Downing (IT Spezialist), Arup Australia, erklären

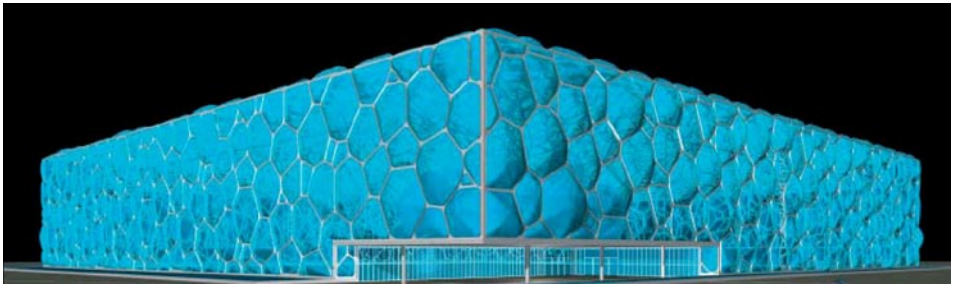


Bild 6.1-1 <<The structural Engineer>> 6 July 2004

die Vorgehensweise von der Erstellung des 3D Modells bis zu den fertigen Werkpläne.
«AD .» Architectural Design 6 July 2004: 23-26.

Der *Water Cube* in dem die Schwimm- und Tauchwettkämpfe während den olympischen Spielen 2008 stattfinden werden, ist ein Project der China State Construction & Engineering Corporation (CSCEC), in Kollaboration mit PTW Architects (Sydney) and Arup (Sydney).
Die Struktur des 'Water Cube' besteht aus einem Räumlichen Stabtragwerk dessen Abmessungen 176m x 176m x 29m betragen. Die Wabenstruktur die dieses Tragwerk charakterisiert besteht aus Pentagone und Hexagone die dreidimensional wiederholbar sind ohne Lücken zu hinterlassen.

Die Modellierung und die Dokumentation der 24 000 Stäbe, 12 000 Knoten und die Erstellung der 112 repräsentativen Gebäudeschnitten war eine Große Herausforderung für die Planer. Insbesondere weil die definitiven Abmessungen der Tragwerkskomponenten, die in ihren Dimensionen variieren erst in der Endphase des Projektes hätten vorliegen können. Die komplette Dokumentation des Projektes hätte somit innert kürzester Zeit erstellt werden müssen.

Microstation VBA script- Wettbewerbsphase

Die Planer haben sich somit entschieden folgenden Weg zu gehen:
Da während der Wettbewerbsphase noch keine endgültige Geometrie vorlag, lag der Fokus beim schreiben eines Programmskripts der, ein *wireframe* Modell in kürzester Zeit zu einem repräsentativen 3D *solid* Modell hätte umwandeln können.
Das 3D Modell wurde mit der Microstation VBA Skriptsprache erstellt und konnte in seiner Anfangsphase nur Stäbe und Knoten gleicher Dimensionen verwalten.
Erst später konnte das Programm mit variierenden Dimensionen der Tragwerksteile umgehen und auch ob das Tragwerk als *solid* oder als *surface* Element hätte erstellt werden sollen, ist in einer späteren Phase implementiert worden. Diese Differenzierung ist wichtig, da je nach Ausgabeformat z.B. STL oder DWG die Geometrien respektive als *surface* oder *solid* vorliegen müssen. Das mit diesem Programm generierte Tragwerk konnte als DXF in Strand 7 zur Strukturellen Analyse exportiert werden.

Bild 6.1-2
<<The structural Engineer>> 6 July 2004

Microstation VBA script- Planungsphase

Um die Analysedaten in Microstation zu transferieren, wurde ein weiteres Skript geschrieben, welches diese Daten in ein für Microstation verständliches Format exportierte. Als dann das *wireframe* Modell komplett vorlag und schon die Bemessung der Stäbe und Knoten beendet war, sind *point* Elemente hinzugefügt worden mit den Bezeichnungen der Elemente und dessen Dimensionierungen. Für jede Komponente sind zusätzlich weitere Textelemente hinzugefügt worden. Diese wurden dann für die Erstellung der 2D-Schnitte in TriForma benutzt. Das dreidimensionale Modell und die Textelemente sind in das von AutoCAD benutzte Dateiformat DWG als Blockelemente exportiert worden damit sie dessen Attribute beibehielten.
Alle Elemente sind in ein lokales Koordinatensystems generiert worden und wurden erst zum Schluss in das Globale Koordinatensystems übertragen. Dabei musste bei der Orientierung der einzelnen Teile Vorsicht geboten werden da sie keine Information darüber enthielten. Besonders war darauf zu achten, dass Deckenelemente und Fussbodenelemente nicht vertauscht wurden. Einige Koordinaten hat man deswegen als 'hard coded ' fest in das Programm eingebettet, damit diese Elemente an die richtige Stelle generiert werden konnten.
Das Programm hat als End-Output detaillierte und strukturierte Daten in Form einer Excel-Datei ausgegeben. Die Daten, die die Excel-Datei beinhaltete, konnten in einem Optionen Menu selektiert werden. (*Bild 6.1-4)*
Das fertige VBA-Skript war in seinem Endstadium über 3000 Zeilen lang, beinhaltete 330 Variablen (zwei von denen waren *arrays* mit mehr als 20 000 Einträgen) und hat weniger als 25 Minuten gebraucht, um das 3D-Modell zu erstellen und die Daten in die Excel-Datei zu schreiben.

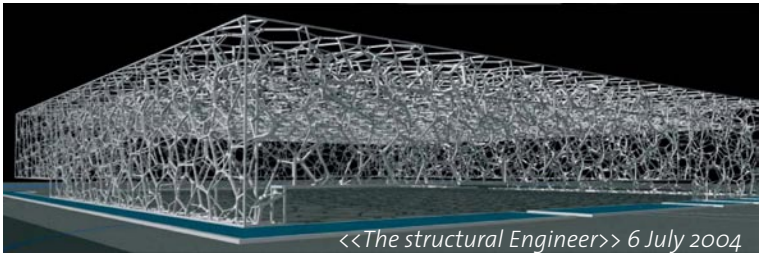


Bild 6.1.-3

Die Gebäude Dokumentation

Nachdem das Modell in Strand analysiert und via VBA Skript wieder in Microstation importiert worden war, wurde es mit einem von Microstation eigenem Modul TriForma für die Erstellung der Werkpläne bearbeitet. Mit diesem Modul sollten die 112 Gebäudeschnitte erstellt werden. Um diese Pläne mit *intelligenten Objekten* von TriForma zu erstellen, mussten die Elemente des 'wireframe' Modells durch diese Objekte ersetzt werden.

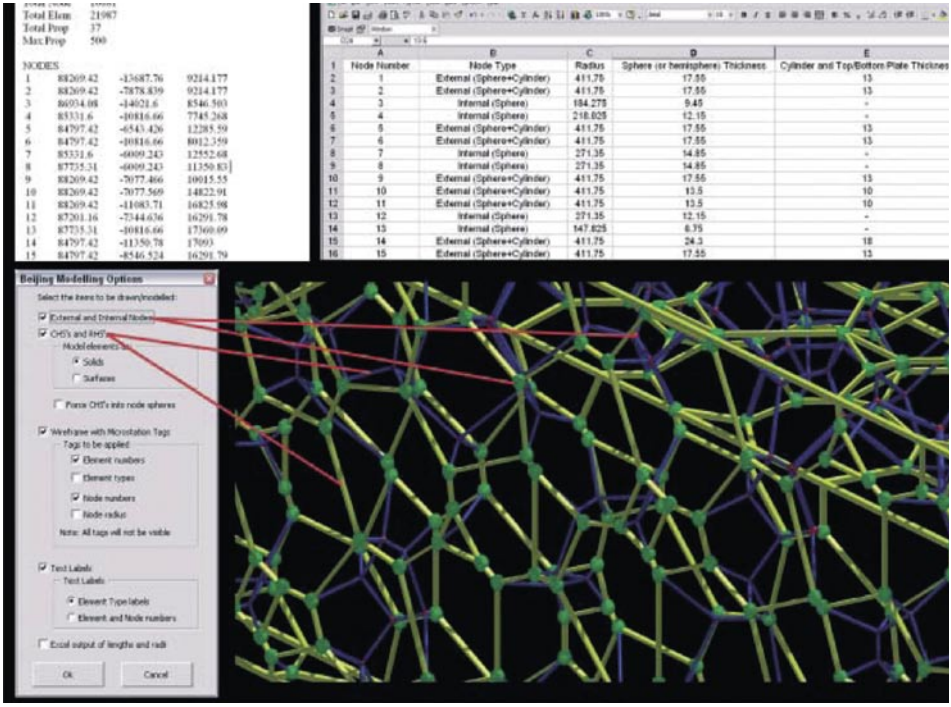


Bild 6.1-4 VBA Skript

<<The structural Engineer>> 6 July 2004

Dazu mussten die einzelnen Tragwerkselemente in Microstation selektiert werden. Jedes selektierte Element enthielt die Information über dessen 37 spezifischen Eigenschaften in Form von versteckten Attributen.
Alle 37 Eigenschaften konnten als Selektions-Kriterium über die Funktion *Select by Attribute* fungieren, so dass in TriForma nur noch das passende Element von Hand auszuwählen war um es in den Plan einzufügen. **(***Bild 6.1-4***)** Am Ende dieses Arbeitsschrittes (*skinning*) war das *wireframe* Modell in ein vollständiges hoch-detailliertes 3D-Solid Modell umgewandelt worden. *(Bild 6.1-5)*

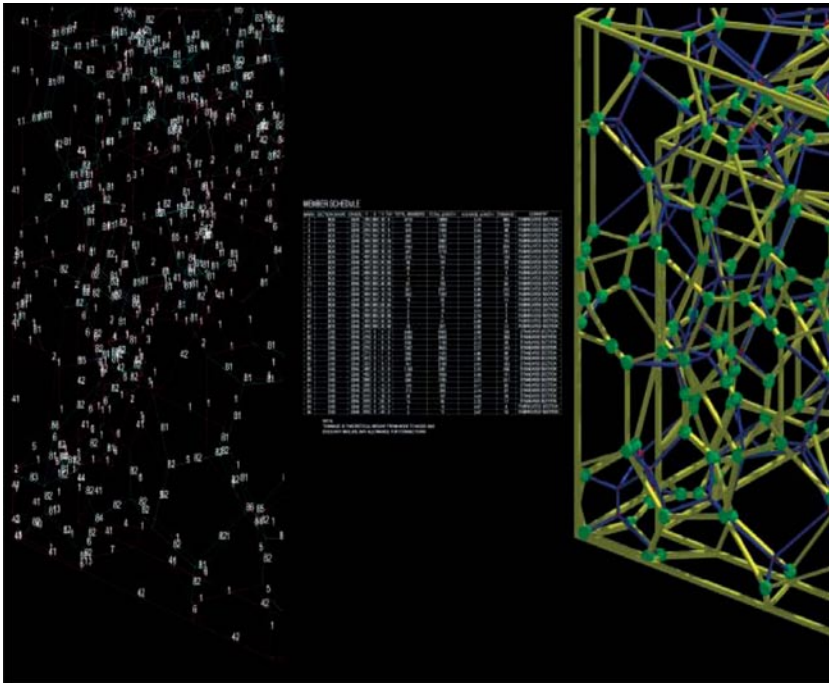


Bild 6.1-5 Wireframe/Skinning *<<The structural Engineer>> 6 July 2004*

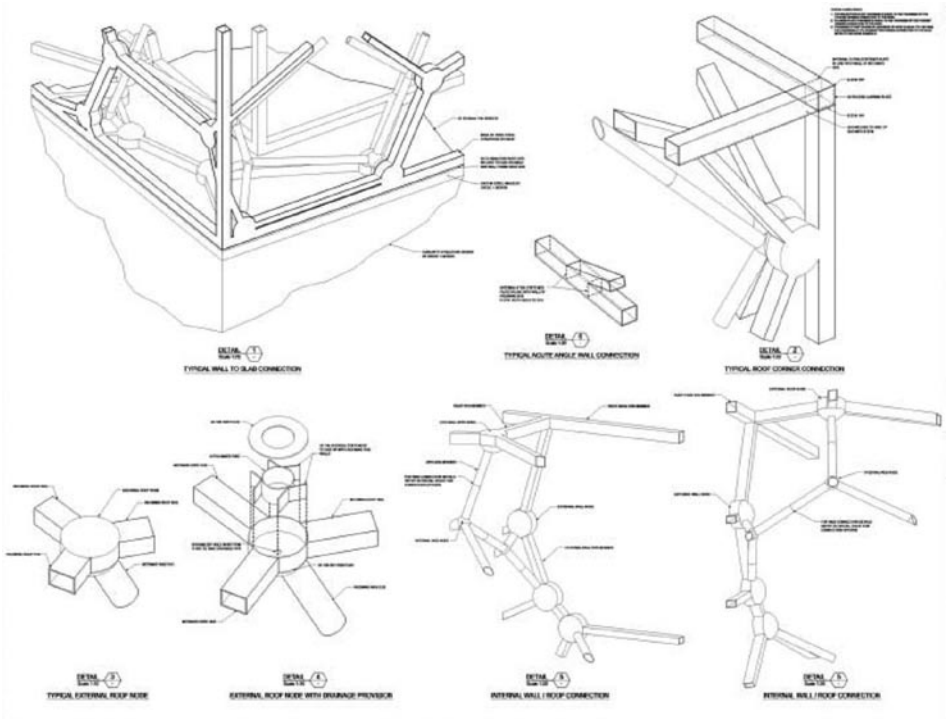


Bild 6.1-6
Deatail Zeichnungen *<<The structural Engineer>> 6 July 2004*

Dieses so genannte *skinning* Prozess hätte auch von einem VBA-Skript übernommen werden können. Die zeitlichen Einschränkungen ermöglichten jedoch den Planern keine weiteren Erkundungen in den VBA-Skriptfunktionalitäten von TriForma.
An den 45 Werkplänen musste dann nur noch mit wenig „Handarbeit" Änderungen und Ergänzungen vorgenommen werden um sie zu vervollständigen. *(Bild 6.1-6)*

Der in diesem Projekt beschrittene *Workflow* gewährleistet eine hohe Flexibilität, da es bei eventuellen Änderungen an Tragwerksteilen oder an der ganzen Struktur, das 3D Modell und die komplette Dokumentation innert kürzester Zeit wieder auf den aktuellen Stand versetzt werden kann. Bei einem von den Planern durchgeführten Update-Test sind die 45 Zeichnungen erfolgreich in nur einem Wochenende auf neustem Stand gebracht worden. Nur wenige kleine Fehler waren noch von „Hand" auszubessern.

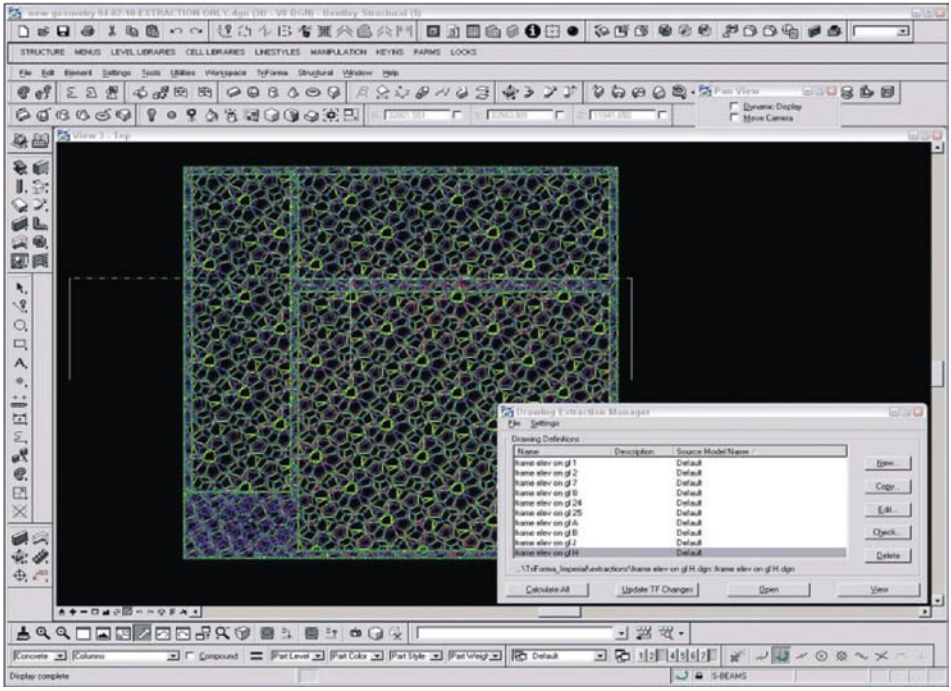


Bild 6.1-7
Gebäude Schnitte mit Triforma

<<The structural Engineer>> 6 July 2004

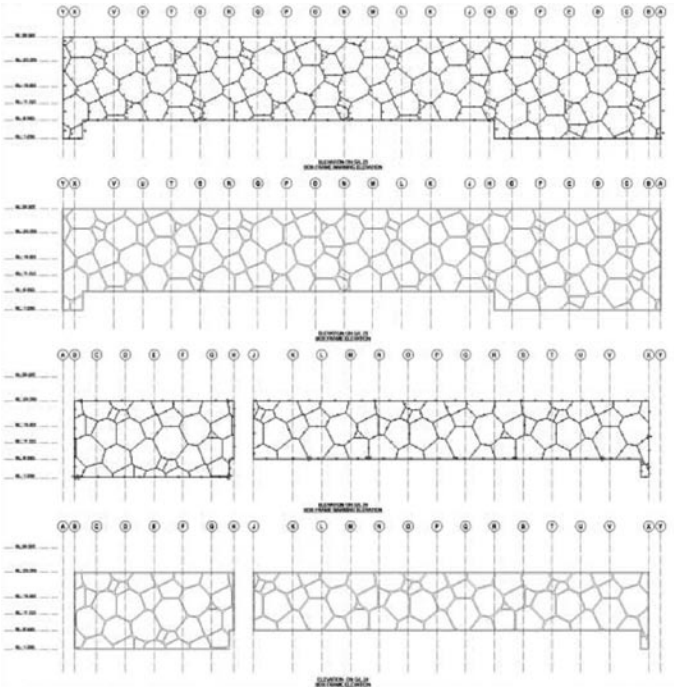


Bild 6.1-8
Gebäude Schnitte *<<The structural Engineer>> 6 July 2004*

Der in diesem Abschnitt Beschriebene Workflow für die Werkplanung des Beijing Water Cube, präsentiert starke Analogien mit dem der für das NDS 2004 Prototyp beschritten wurde.
Die angewendeten Techniken verschaffen dem Planer extreme Flexibilität, diese könnte in diesem Masse mit konvetionellen CAAD-Techniken nicht erreicht werden.

6.2 The Digital and the Material

Für dieses Projekt wurde die Geometrie in Einzelteile unterteilt und für die CNC-Fertigung und den Transport optimiert.

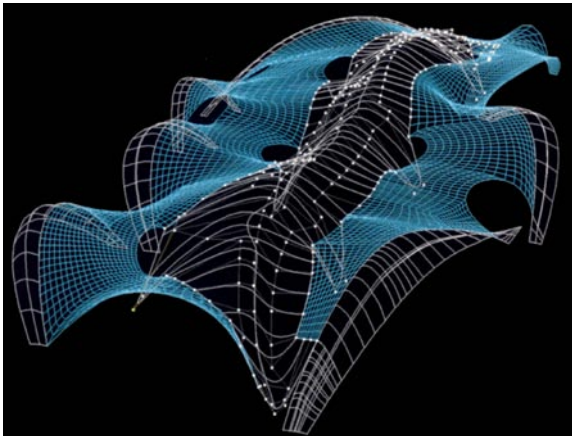
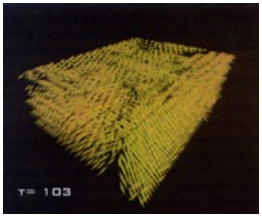


Bild 6.2-1
Parameterisierte Maschenweit <<Architectural Design>>



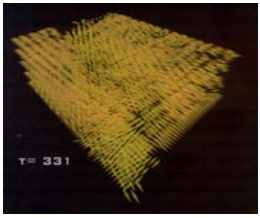


Bild 6.2-2 *<<A.D>>*
Unterteilung der Geometrien für die CNC-Fertigung

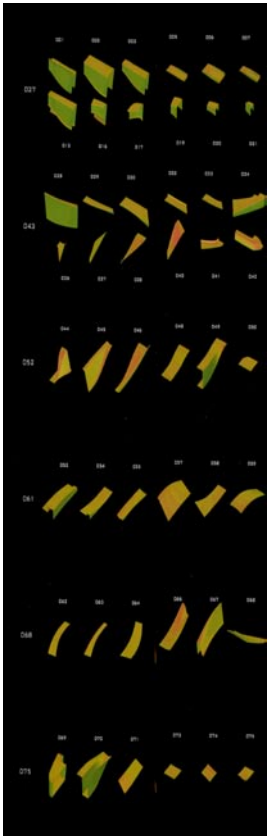


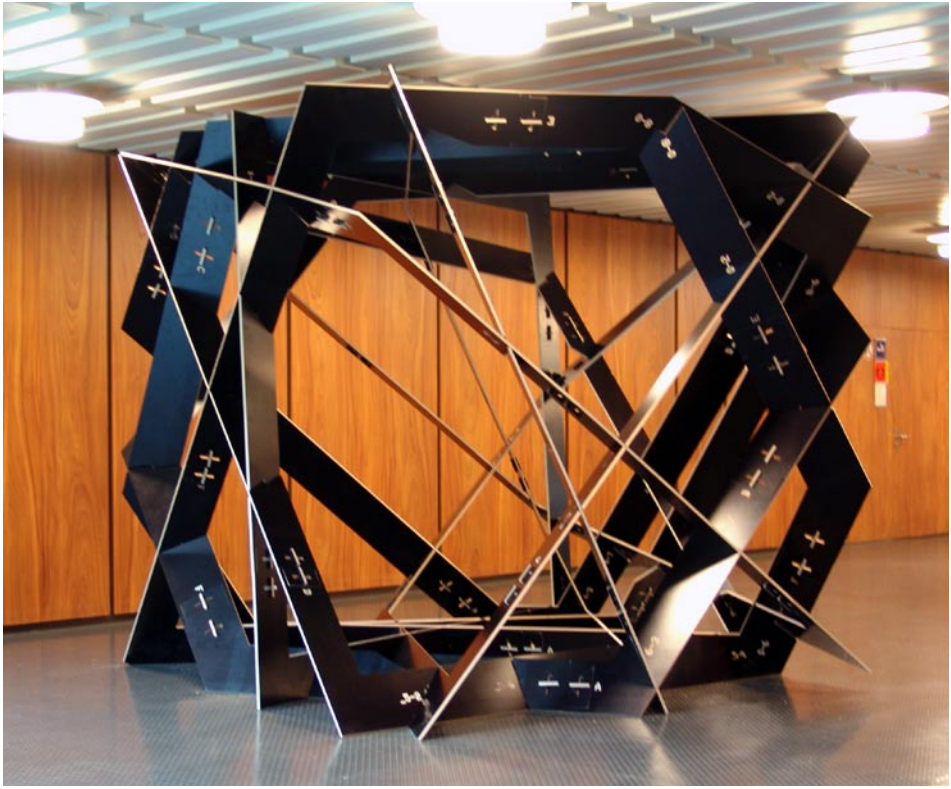
Bild 6.2-3 *<<A.D>>*
Stückliste

7.0 Schlussbetrachtungen und Folgerungen

Nach zweimonatiger intensiver Auseinandersetzung mit der Thematik der automatisierten Werkplanzeichnung und der Programmierung in MEL am Beispiel des NDS2004-Abschlussprojektes, kommt der Autor dieser These zu folgenden zusammenfassenden Schlussbetrachtungen und Folgerungen:

Das Wissen aus den Experimenten, die in der Anfangsphase des Projekts mit Mayas reichhaltigen Werkzeugpaletten der Module "Modeling", "Animation", " Dynamics" und „Cloth" durchgeführt wurden, konnte für das Projekt nicht vorteilhaft angewandt werden. Dies weil die Verwendung der MEL-Befehle der Oben genannten Module eine auf Mayas Programmlogik abgestimmte Beschreibung der Geometrien und Datenstrukturierung bedingt. Da aber am Anfang angenommen wurde, dass für den Generierungscode der Struktur ein mathematisches Skript-Modell mehr Flexibilität geboten hätte, musste man im Verlauf des Projekts auf die daraus resultierenden Umständlichkeiten reagieren. Die Prozeduren des Konstruktions-Stylesheets und der automatisierten Werkplanzeichnung wurden deswegen in den Generierungscode der Struktur integriert. Nur auf diese Art und Weise konnte eine vollautomatische Erzeugung der Werkpläne sichergestellt werden, an denen nur geringe Handarbeit geleistet werden musste. Andernfalls wäre es nur durch hardcodierte Einträge in einigen Prozeduren oder auf semi-automatischen Weg möglich gewesen. Der beschrittene Lösungsweg erwies sich als geeignet, um die erwarteten Anforderungen und Ziele zu erfüllen. Der Autor dieser These würde jedoch mit dem Wissen, das durch dieses Projekt gewonnenen wurde eine ähnliche Aufgabenstellung auf anderem Wege angehen. Man würde zum Beispiel versuchen eine engere Verzahnung zwischen den beschrittenen mathematischen Lösungsweg und Mayas Programmlogik zu erzielen oder

einen Lösungsansatz aufzustellen, der rein auf den 3D-Vektor basiert.
Das NDS2004 Ausstellungs-Prototyp ist nicht als architektonisches Statement zu verstehen, dennoch möchte es bis ins Detail den beschrittenen computergenerierten Entwurfsansatz und die CNC-gesteuerte Produktion hervorheben.



7.1 Bildergallerie

In diesem Kapitel wird eine Auswahl der auf der Beigelegten CD-ROM enthaltenen Bild-Dokumentation vorgestellt.