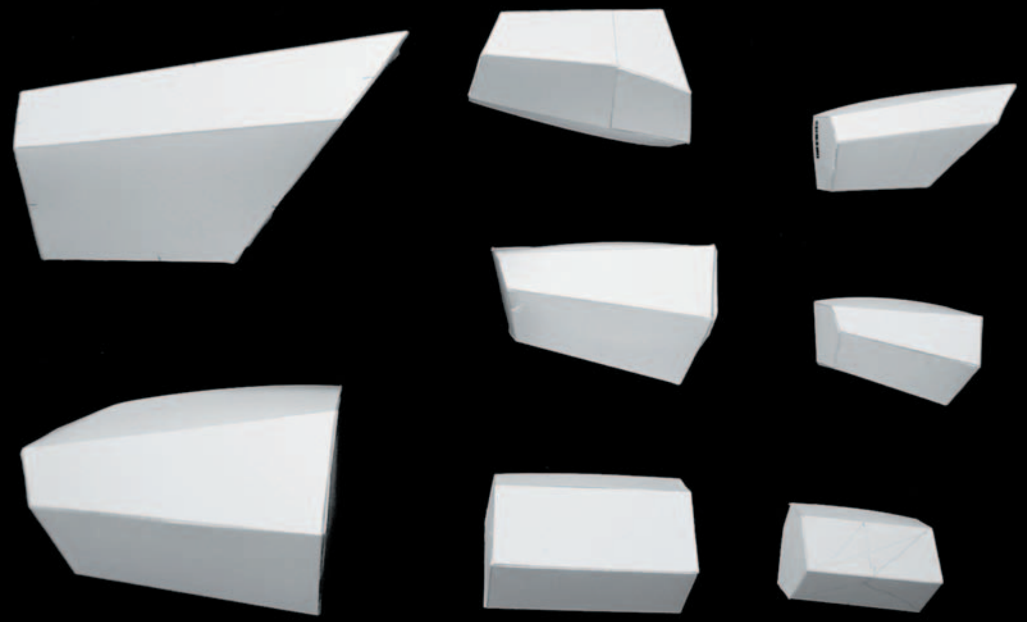
http://www.caad.arch.ethz.ch

CHRISTIAN DÜRR

MORPHOGENESIS

ETHZ | CAAD | NDS2004

MORPHOGENESIS
Evolution of Shape



IMAGINATION AMPLIFIER 1.0
Interactive Housemachine

The advent of computer technologies in the design-processes has already taken place, is meanwhile ordinary. New design perspectives are opened, and an almost inexhaustable form repertoire is available, even buildable - 'Nothing is impossible'. This thesis work deals only with a small clip from there. Essentially it consists of two parts:
» MORPHOGENESIS – Evolution of Shape « describes the present situation of generating shape with the help of computers. Some of the technologies that are used for, as comupter-morphing or evolutionary programming, are examined more closely here.
» IMAGINATION AMPLIFIER Version 1.0 « is an interactive FormGenerator for houses - a HouseMachine. The program deals with the possibilities of interpolation and morphing between two, 3 dimensional, states (Start- and TargetHouse) configured by the user. As an output–result, the generator depicts the put in number of steps in between, with characteristic values like cubical contents (V), surface quadrature (A) and the relation between V/A.
All the results are stored in a database, where it is possible to select from for new morphing operations, to get in the end closer to a more optimized shape.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Chair of CAAD
Institute for Building Technology
Prof. Dr. Ludger Hovestadt
ETH Hönggerberg
HIL E 15.1
8093 Zurich

www. caad.arch.ethz.ch

Christian Dürr | Thesis NDS2004
Subject: MORPHOGENESIS - The Evolution of Shape
IMAGINATION AMPLIFIER V1.0 - A Interactive Housemachine

Zurich, october 2004

This work I dedicate to all people,
being prepared to pass on their knowledge to people,
willed to learn from them.

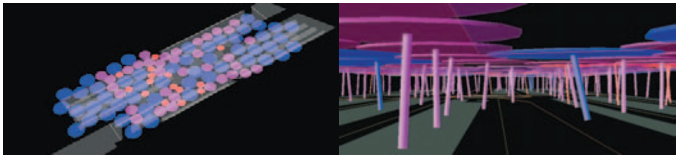
CONTENTS

INTRODUCTION

Philosophy of the professorship

»The professorship for CAAD at the Swiss Federal Institute of Technology Zurich, conducted by Ludger Hovestadt, is researching for the integration of current information technologies into architectural design- and production processes. Aim of the research is not the development of new architecture under formal aspects, rather than the structural development of buildings from its core: architecture is described not by drawings, but by code, fixed in a data record – comparable to a 'gentic code' – containing all relevant building informations. This data record is configurable, depictable and produceable by an output device: in the internet, on the plotter or directly as component. «[1]

As an example » the 'groningen twister' is an application of generative design methods to real life architecture: the new „stadsbalkon" in front of the groningen train station in the netherlands, designed by kees christianse architects and planers (kcap) in rotterdam.
the architectural design required the placement of over 100 columns in the basement which bear the load of a pedestrian area on top. the slab has various holes, incisions and ramps. there are three different types of columns with differing bearing capacities, they are allowed to be tilted to a maximum angle of ten degrees and they can only be placed in areas where they do not obstruct the pathways.
a software, completely programmed in java (and java3d) allows the interactive placement of the columns. the columns grow and arrange themselves within the basement according to simple rules of thumb that were defined by the engineering partner, the office of arup in amsterdam. each column acts as an independent "agent", trying to find its best location while being pushed by other columns and drawn towards various attractors. the designer can interact with the system by picking and moving columns and by adjusting various parameters.«[2]



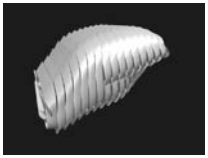
[1] »Bauen mit Maschinen« , Oliver Fritz, »archithese« 2.03 P.46-51;
[2] »Groningen Twister« , Fabian Scheurer, Chair of CAAD - ETH Zurich;

Chapter A| MORPHOGENESIS

A1 SHAPE made by computer

» . . . SHAPE is invented, produced in computers, generated, evolved, transformed. Design as an emergent process, the computer as BlackBox of creativity. « [3]

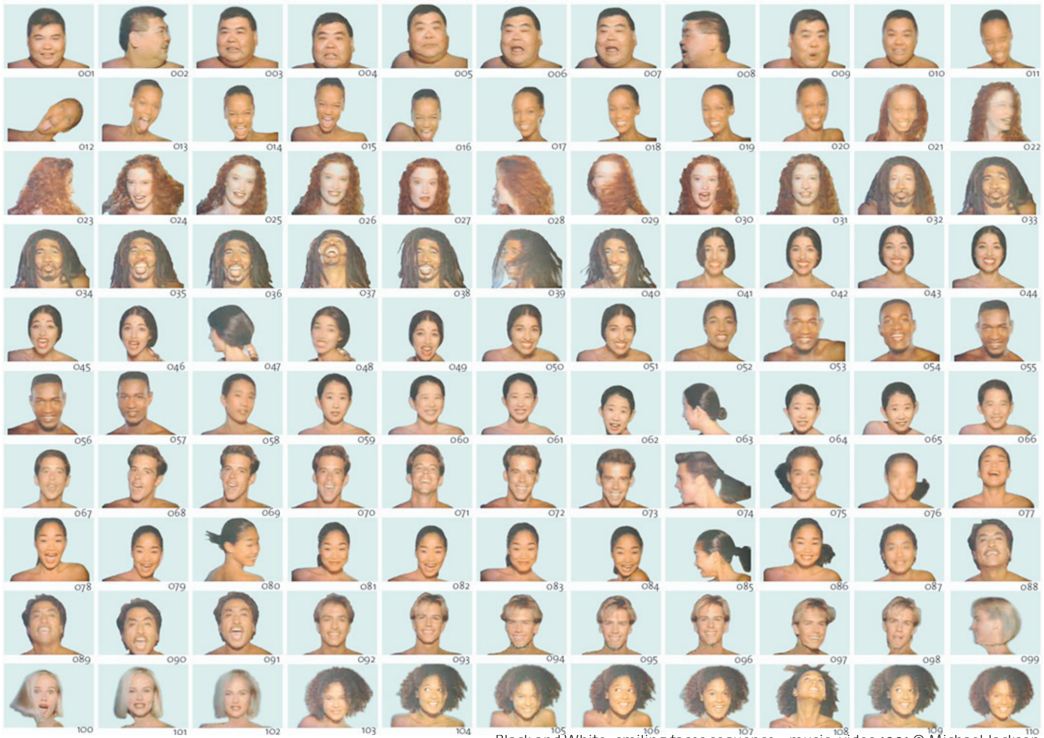
Today Design and SHAPE are generated mainly with computers. The current discussion about 'Biomorphism' is a indication for the assertion. Undoubtedly, the geometrical SHAPE repertoire is being enriched by computers. With this new tool almost every dented, twisted,stretched or compressed SHAPE beyond the euclidian geometry is modelable, even realizable and buildable with the finite elemente calculation.



A2 »Dynamic SHAPE « - ComputerDesign Strategies

ComputerSupported - or ComputerGenerated SHAPE ?
There are two different methods to design SHAPE with computers: The ComputerGenerated SHAPE - Process is characterized by an automatic process, without the intervention of the user, who predefines the parameters and algorithms of the program. In contrast, the ComputerSupported SHAPE -Process intends the possibilities of feedbacks and intersections between man and machine. Intuition and knowledge of users are fixed parts of the generative process.

Picture Credits »OPEL« car design and development | Greg Lynn Architect, Los Angeles | NOX Architects, Holland
[3] »Formfindungen« , Florian Böhm, »Arch+ 159/160« P.128 ;


»Black and White -smiling faces sequence«, music- video 1991 © Michael Jackson

Morphology and Morphing | What is morphing and what can it do to you? First developed in 1988 for the film »Willow«, which shows animals changing into other animals, morphing is an evolving technology, involving computer animation. Basically the programmers place grids over two dissimilar objects, connect the intersections of the grid, and stretch or compress the sections, in order to create a continuum of intermediary SHAPE and colors, similar as D'Arcy Thompson or Albrecht Dürer did it.

Michale Jackson's music video »Black or White«, published in 1991 was the first video ever to use the 'Morphing Special Effects', for the smilling faces sequence. It starts with the face of a mellow sumo wrestler, he shakes his head and activates the morph that changes him into a thin black woman, who continues to smile, before she twists her head , bleaches, freckles and sprouts red hair. The series continues: rasta man, indian woman, black man, asian man, asian woman, white man, asian woman, latino man, white man, white woman, black woman all singing 'yeah, yeah, yeah'.

029

030

031

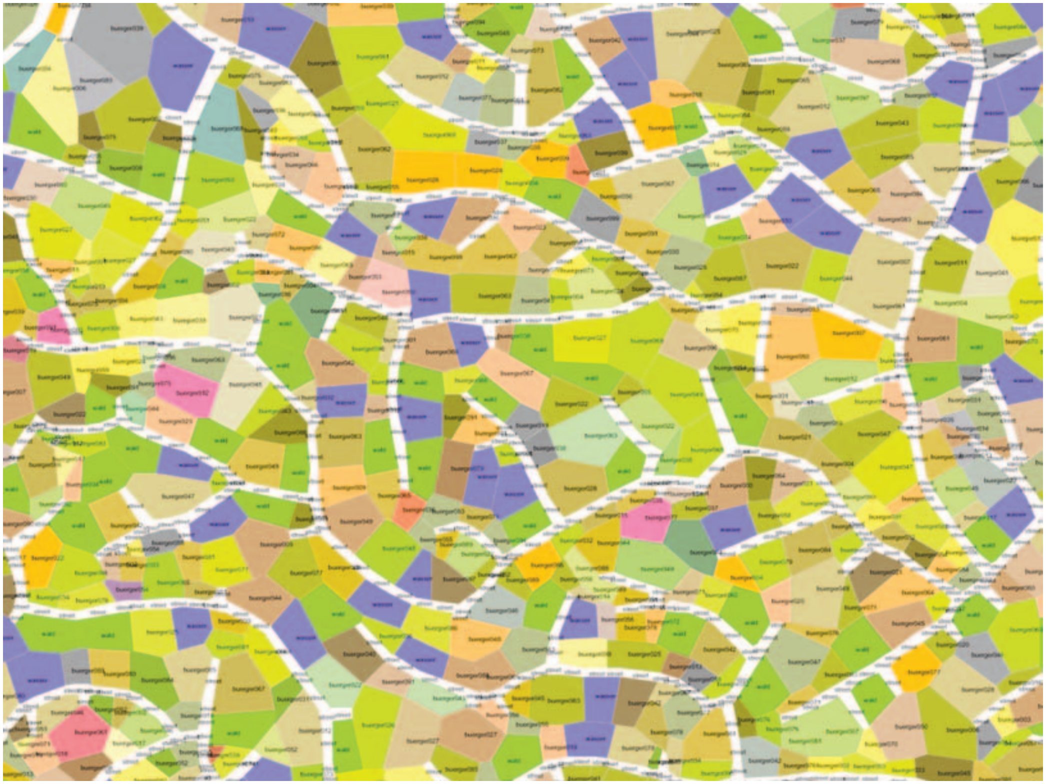
032

033

Big advantage of the morphing technique is the smoothness of coordinate transformation before our eyes, affecting us viscerally with the power of photography and the fancy of animation. Also Morphing Programms enables users to choose the images, to morph between.

Within design-process, morphing is implemented, to achieve the almost optimal design. In analogy to morpology, morphing describes the determination and alteration of SHAPE by the influence of outer forces, choosen at will. The transformation between at least two or infinite states, is described in a number coordinates. The decsription of the coordinate transformation is done in an individual adjustable algorithm. Some of already existing animation-software includes morphing tools like 3dmax or maya.

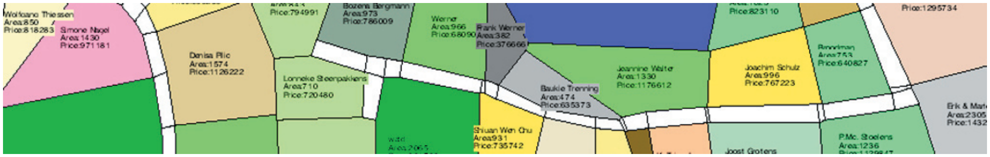
The idea behind the use of morhing techniques in the evolution of SHAPE, is to generate a huge number of variants, to select from and to get out some unexpected results – the computer as »Imagination Amplifier«.


The occupants individual interests are negotiated democratically by the »KaisersRot« - Software, Chair of CAAD - ETH Zurich.

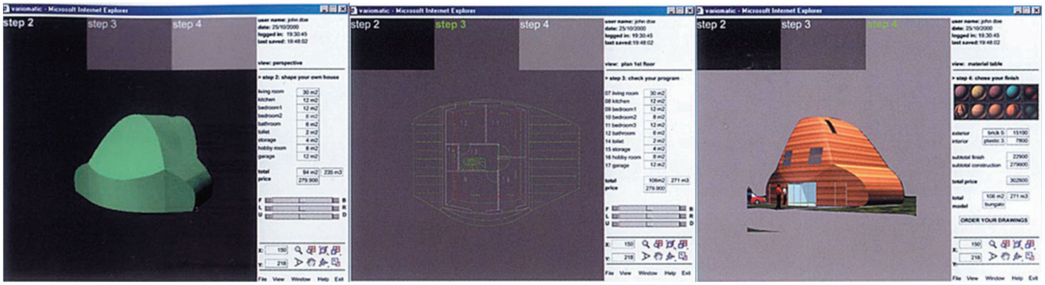
Biomorphism and Evolutionary Design | Biomorphism deals with the transformation of evolutionary principles, like processes taking place in nature e.g. behavoiur of foraging ants, to simulate self-organizing principles as e.g. volume or density of traffic. To amplify the asked properties, the mechanisms of selection and recombination over many generations are installed. The basic concept behind these Artificial Life Technologies are programs, reacting adjustable and flexible to changing conditions. Frequently agent technology is used in this context, that enables to manage processes parallel, and every agent acts simultaneously. Evolutionary Design is based on genetic algorithms , also used in cellular automata, where numbers of values are represented by geometric objects, like rectangles or circles.

The very interesting thing in this system is the increasing interactivity within the SHAPE generating process, between man and machine, by getting variants to a given task by the computer, as well as the user has the possibility to generate his own variants. The next step of communication between user and machine within the process of SHAPE, is the participation of several people with individual needs, as in 'mass customization' intended.

As an example, the »KaisersRot«-Software is a program for generating selforganizing urban structures. »Each plot in the mental settlement has specific properties, e.g. size, proportion, type and position of building, as well as external attractors like, water, nature, infrastructure or public traffic. In a phased simulation all of the plots striving for optimization to fullfil the needs put in. The software reacts regenerative, that means if one plot will be deleted, at this position the structure grows together. Are all parameters fullfilled best, a balance stabilizes within the system. Instead of a Top-Down-Method, where design is going from bigger to smaller scale, here the individual needs of the future occupants are negotiated in a Bottom-Up-Principle.« 4



4 »KaisersRot« , Oliver Fritz/Markus Braach, »werk, bauen + wohnen« 04, 2002 P.4;


variomatic - Microsoft Internet Explorer
step 2
step 3
step 4
File View Window Help Exit

user name: john doe
date: 25/10/2000
logged in: 19:30:45
last saved: 19:48:02

view: perspective

> step 2: shape your own house

living room	30 m2
kitchen	12 m2
bedroom1	12 m2
bedroom2	8 m2
bathroom	6 m2
toilet	2 m2
storage	4 m2
hobby room	8 m2
garage	12 m2
total	94 m2	235 m3
price	279.900

F B
L R
U D
X: 150
Y: 218

user name: john doe
date: 25/10/2000
logged in: 19:30:45
last saved: 19:48:02

view: plan 1st floor

> step 3: check your program

07 living room	30 m2	
08 kitchen	12 m2	
09 bedroom1	12 m2	
10 bedroom2	8 m2	
11 bedroom3	12 m2	
12 bathroom	6 m2	
14 toilet	2 m2	
15 storage	4 m2	
16 hobby room	8 m2	
17 garage	12 m2	
total	106m2	271 m3
price	279.900	

F B
L R
U D
X: 150
Y: 218

user name: john doe
date: 25/10/2000
logged in: 19:30:45
last saved: 19:48:02

view: material table

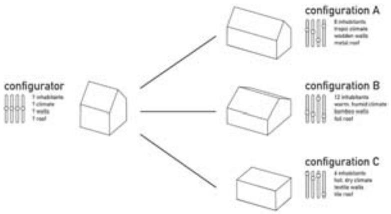
> step 4: chose your finish

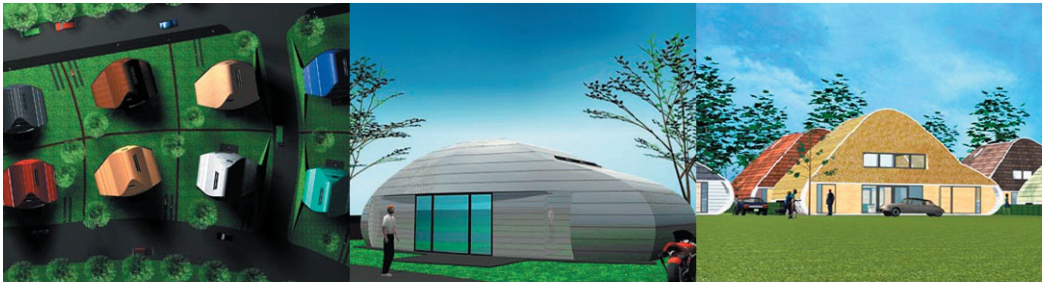
exterior	brick 5	15100
interior	plastic 3	7800
subtotal finish		22900
subtotal construction		279900
total price		302800
total	106 m2	271 m3
model	bungalo	

ORDER YOUR DRAWINGS

X: 150
Y: 218

Variomatic House - configurator, individual settings like height, width, depth and materiality can be taken here.




Variomatic House - the house will be configured within a already planed settlement.

A3 Parametric Design and Mass Customization

The concept for individualized SHAPE is based on the model of »parametric design«, that offers the perspective, to consider designer and client as co-designers. Design with others, and not only for others leads to a different point of view in the SHAPE generating-process, as well as a changing market-politics – connecting production directly with demand, to react in dependency to needs of the market and to avoid wasting of resources. Individualized mass-production, mass customization, is the mantle for the realisation of individual parametric design. Mass customization itself is the answer to growing numbers of individualization- and globalization tendencies in the society, having an effect on LifeStyle and also on an individualized SHAPE-vocabulary. People asking for products to emphasize their uniqueness.

Parametric Design | is founded on a predefined set of rules and regulations, aligned to specific functions and questions, for each product to fulfill. By the change of parameters within a predefined range of variations, a individual product is created, representing at the same time one of infinite possible variations of this specific regulation. The product-model is no more an exact stored geometry, rather a combination of parameters of the geometric figure and numeric variables of these parameters to describe the size of these object variant.

Mass customization | is based on the idea, to enable clients to contribute their needs and wishes in the design process, without being overtaxed by the possible options. The aim is, to offer and produce goods with the same price of mass-produced articles. Through the feedback between clients and producer, informations introduced by the clients can be utilized for the development of further products.

Configurators and Generators | are bundling parametric regulations in the form of a program, visible for the user through the Graphical User Interface (GUI), where to take individual settings. Worlwide Interfaces enabled by the internet, makes it possible to reach and communicate with clients all over the world. Configurators are mediators between client and product. Data collected, are the basis for the manufacturing of individualized objects.

» **Variomatic – Kas Oosterhuis, Netherlands** | Interactive aspects of parametric design This new concept for a catalogue house is elastic in all directions · In height, depth and width · Hence its name Variomatic · The unicity of the project is that the clients are co-designers of their own house · Not only do they define the final form of the curves (of the façades or the roofs) but also, as they wish, the overall dimensions of the house, the place where the kitchen should be or the installation of a solar water-heater They can moreover choose among many materials and colours to finish the volumes: reed, wood, metal, tiles, pvc. « [5]

[5] Kas Oosterhuis, Variomatic, http://www.oosterhuis.nl/variomatic/variomatic.html

Chapter B | INTENTION OF WORK

B1 Episode »Snapshot from life of Family Glück«

At last! After twenty years of unsuccessful lottery, Hans Glück cracked the jackpot. 10.000.000 SFr – now for himself and his family a new era begins without turning around each centime : a new car of course, a trip on the Maldive Islands and moving out of the too small 3 1/2 rooms apartement, as soon as possible. A own house. Within the family a passionated discussion breaks out, about how the new home has to look like. Hans and his wife Hilde have more something traditional in mind, chaste and moderate – a small house with ridge or mansard roof, but in any case no flat roof, especially Hans is suspicious against it. Martin and Vicky the two children of the Glücks, are not very happy with their parents ideas, especially this word »traditional« they don't like. After a while the discussion focuses on two questions:

How does a house look like ?
How could a house look like?

Martin, the older of the two children, spend in his free time a lot with writing little computer programs. Later on he intends to study computer sciences. Vicky is more interested in industrial design also architecture she likes. The day after, Martin and Vicky talking about yesterdays discussion. While sitting together and talking, on TV runs an old video clip of Michael Jacksons »Black and White«. In the last scenes of the clip human faces changing their figure and skin color smoothly, as in a flip-book, but much more perfect. Fascinated by the morphs , Martin decided to write a program, that could be helpful for the FORM - discussion of the house with his parents. For the house- program he also wants to use the computer-morphing, describing the way between two or infinite states, smoothly and without interruptions. For the states from one form to another, he intends to use a coordinate model, where during the morph coordinates are transformed from one position to the next and stored in a database, to build up a visual model and printing it out, also to select one of the morph results to recombine it. Vicky likes the idea very much. Especially the computer-morping technique, because of the possibilities it offers. It's like an »Imagination Amplifier«, to get inspired by unexpected results – design as an emergent process. She would have like to build a series of models from the morphs, to help their parents to imagine.
Later on they talk about their idea to Hans and Hilde. Both are not really clear about what their chrildren are talking about, »Computer-Morphing? Database? Never heard before.«, but they like the commitment of Martin and Vicky, and finally they all agree, that their house should be something special, different from others.

- to be continued -

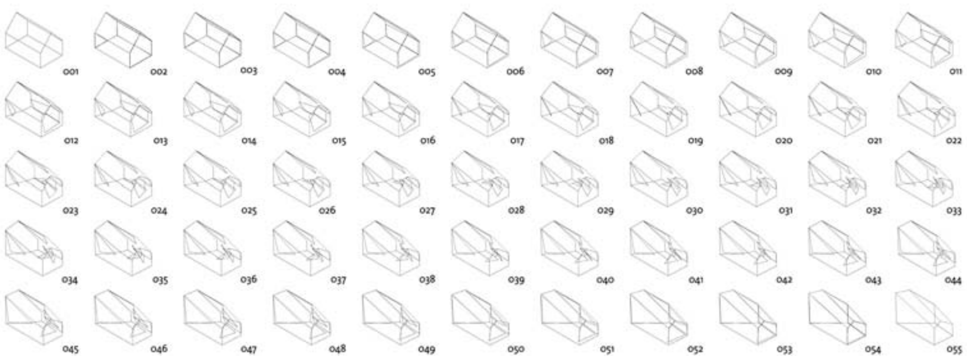
Chapter C|

IMAGINATION AMPLIFIER 1.0 FUNDAMENTALS

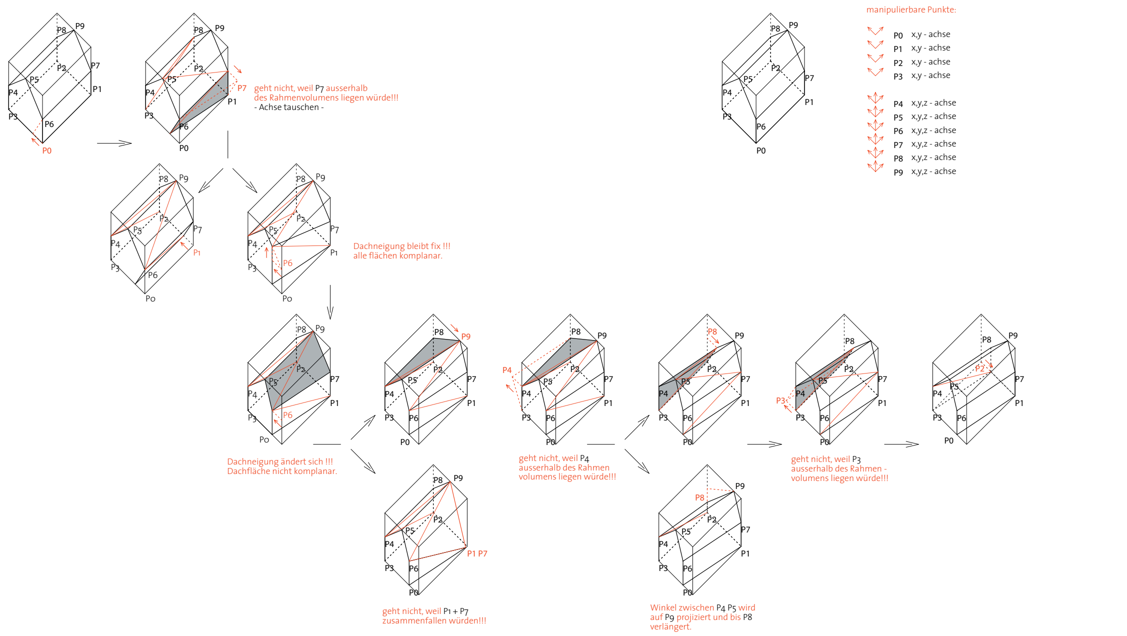
C1 Program Structure - » IMAGINATION AMPLIFIER 1.0 - HOUSEMACHINE «

Essentially the »HouseMachine« consits of two parts:

1. GENERATOR	• Definition of morphing-algorithm between to 3d- shapes (StartHouse - GoalHouse); • Graphical User Interface (GUI) for program control; • Export of results as DXF, IGES etc.
2. SQL-Database	• Store and retrieve of coordinate transformations (Morphs); • administration of program



Test of an available morphing algorithm in FlashMX | ShapeTweening, but result is not satisfying, less control!

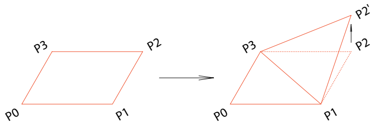

geht nicht, weil P7 ausserhalb des Rahmenvolumens liegen würde!!! - Achse tauschen -
Dachneigung bleibt fix !!! alle flächen komplanar.
Dachneigung ändert sich !!! Dachfläche nicht komplanar.
geht nicht, weil P1 + P7 zusammenfallen würden!!!
geht nicht, weil P4 ausserhalb des Rahmen volumens liegen würde!!!
Winkel zwischen P4 P5 wird auf P9 projiziert und bis P8 verlängert.
geht nicht, weil P3 ausserhalb des Rahmen - volumens liegen würde!!!
manipulierbare Punkte:
P0 x,y - achse
P1 x,y - achse
P2 x,y - achse
P3 x,y - achse
P4 x,y,z - achse
P5 x,y,z - achse
P6 x,y,z - achse
P7 x,y,z - achse
P8 x,y,z - achse
P9 x,y,z - achse

DragPoints-Model - making sure that planes will be coplanar is getting very complicated.

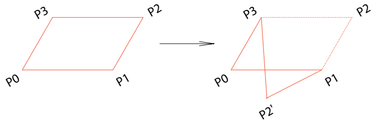
C2 **Strategy** - DragPoints or Planes ?

Which is the best strategy? DragPoints Model has the idea to grab one point by mouse an move it along x, y or z- axis, to form out the model you like to have. Problems are firstly in making sure , that the four or five-cornered planes will stay coplanar, after drag and drop of one point. Secondly to avoid that the face is getting crossed by drag and drop of any point. To manage these to problems mathematically becomes in the end very complicated.

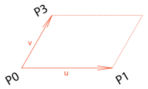
Therefore the Planes Model, because the two problems, mentioned before, are not existing. A plane is from the Mathematical point of view always coplanar, and crossing of the face is not possible, too.



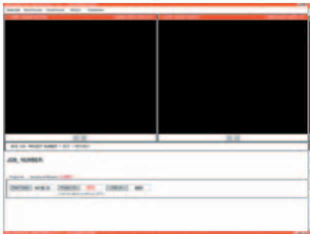
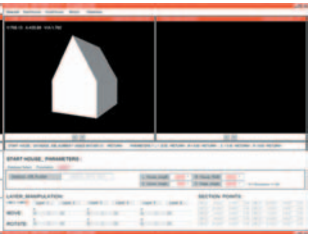
By moving P2 to P2' ,rectangle is not any longer coplanar.

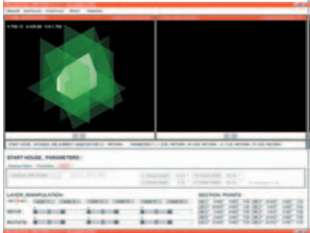
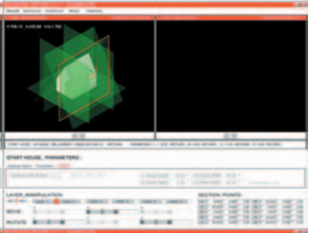


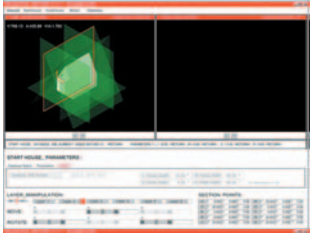
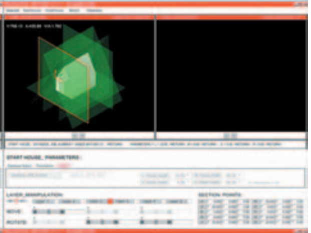
By Moving P2 to P2', face is crossed.

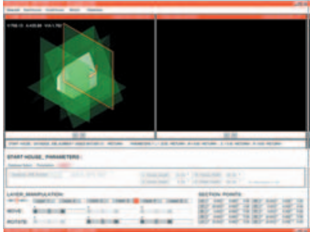
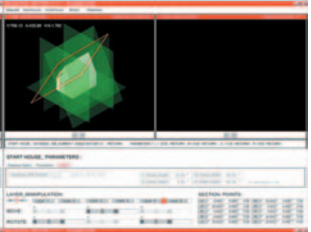


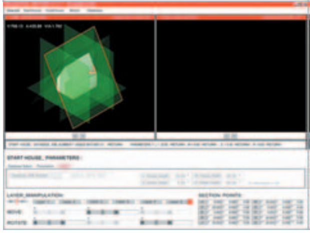
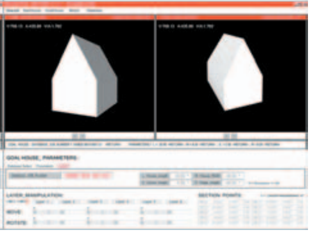
Plane defined by 3 points is always coplanar.



1) | 2)



2.1) | 2.2)



2.3) | 2.4)



2.5) | 2.6)



2.7) | 3)

C3 **Preliminary Draft** - »Housemachine« ProgramRun

1) NewJOB-Number, consists of four segments:

- DateToday (year, month, day): e.g. 04 09 27 ;
- ProjectNumber: e.g. 0815;
- JobNumber: e.g. 0001 ;

→ < SUBMIT > for confirming the Input-Data;

2) StartHouse-Definition, Two possibilities:

- Database - Select: Input of JOB-Number from Database 30 Coordinate-Values are read in;
- House-Parameters: Input Values of HouseLength (L), HouseWidth (W), EavesHeight (E), RidgeHeight (R) ; The Coordinate Values of the symmetrical house are calculated in dependency of the four Input Values.

→ Status of Planes -MANIPULATION: < OFF >, Planes are not depicted and manipulable, and vice versa.

2.1) StartHouse, Status of Planes-Manipulation: < ON >; Planes 1 to 7 are depicted;
• StartHouse_Parameters, < Database Select > and < Parameters > are deactivated;

2.2) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 1 > activated;
• Plane # 1 is moveable in Y – direction, as well as turnable in X- and Z- direction;

2.3) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 2 > activated;
• Plane # 2 is moveable in Y – direction, as well as turnable in X- and Z- direction;

2.4) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 3 > activated;
• Plane # 3 is moveable in X – direction, as well as turnable in Y- and Z- direction;

2.5) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 4 > activated;
• Plane # 4 is moveable in X – direction, as well as turnable in Y- and Z- direction;

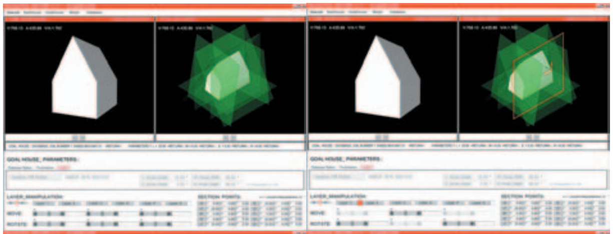
2.6) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 5 > activated;
• Plane # 5 is moveable in Y – direction, as well as turnable in X- and Z- direction;

2.7) StartHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 6 > activated;
•Plane # 6 is moveable in Y – direction, as well as turnable in X- and Z- direction;

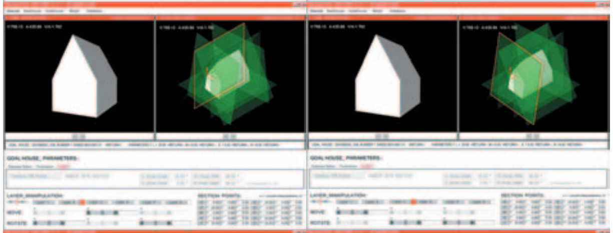
3) GoalHouse-Definition, Two possibilities:

- Database –Select: Input of JOB-Number from Database 30 Coordinate-Values are read in;
- House-Parameters: Input Values of HouseLength (L), HouseWidth (W), EavesHeight (E), RidgeHeight (R) ; The Coordinate Values of the symmetrical house are calculated in dependency of the four input-values.

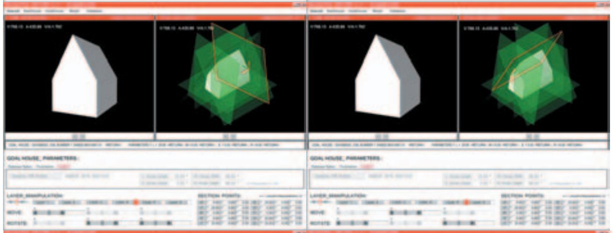
→ Status of Planes -MANIPULATION: < OFF >, Planes are not depicted and manipulable, and vice versa.



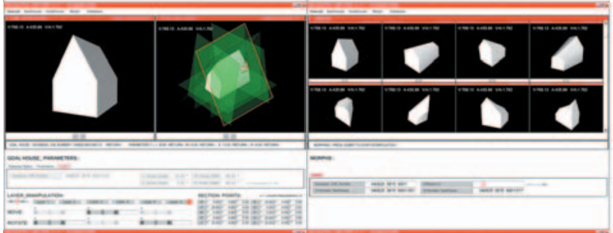
3.1) | 3.2)



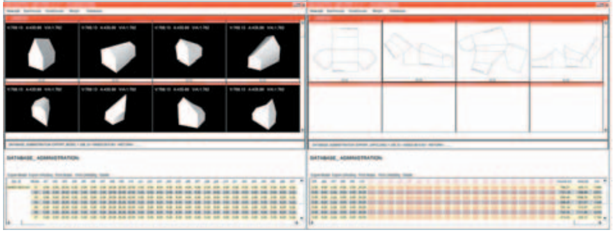
3.3) | 3.4)



3.5) | 3.6)



3.7) | 4)



5) | 5.1)

3.1) GoalHouse, Status of Planes-Manipulation: < ON >; Planes 1 to 7 are depicted;
- GoalHouse_Parameters, < Database Select > and < Parameters > are deactivated;

3.2) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 1 > activated;
- Plane # 1 is moveable in Y – direction, as well as turnable in X- and Z- direction;

3.3) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 2 > activated;
- Plane # 2 is moveable in Y – direction, as well as turnable in X- and Z- direction;

3.4) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 3 > activated;
- Plane # 3 is moveable in X – direction, as well as turnable in Y- and Z- direction;

3.5) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 4 > activated;
- Plane # 4 is moveable in X – direction, as well as turnable in Y- and Z- direction;

3.6) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 5 > activated;
- Plane # 5 is moveable in Y – direction, as well as turnable in X- and Z- direction;

3.7) GoalHouse, Status of Planes-Manipulation: < ON >; → < PLANE # 6 > activated;
- Plane # 6 is moveable in Y – direction, as well as turnable in X- and Z- direction;

4) Morphs:

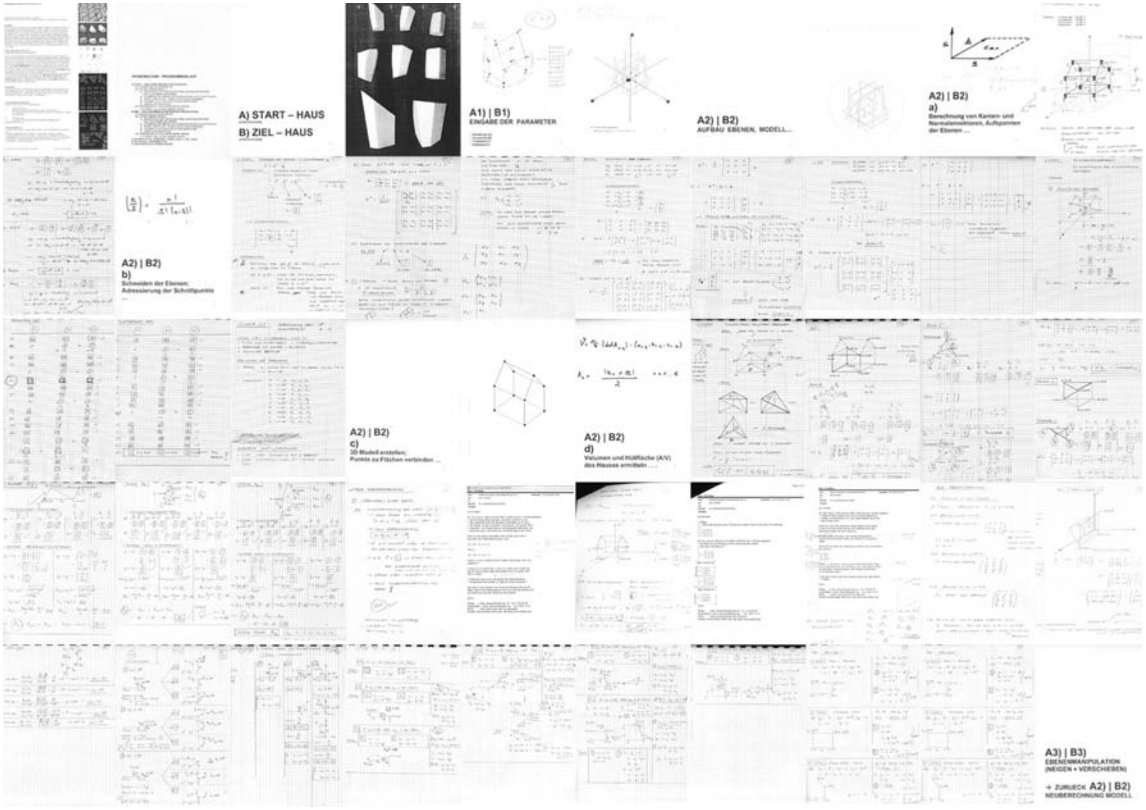
- Number of Morphs (i): coordinates interpolation between StartHouse and GoalHouse in 'i' steps;

→ < SUBMIT > for confirming the Input-Data; → start of morphing – algorithm;
→ ID – numbers of Start- and GoalHouse are depicted on screen;

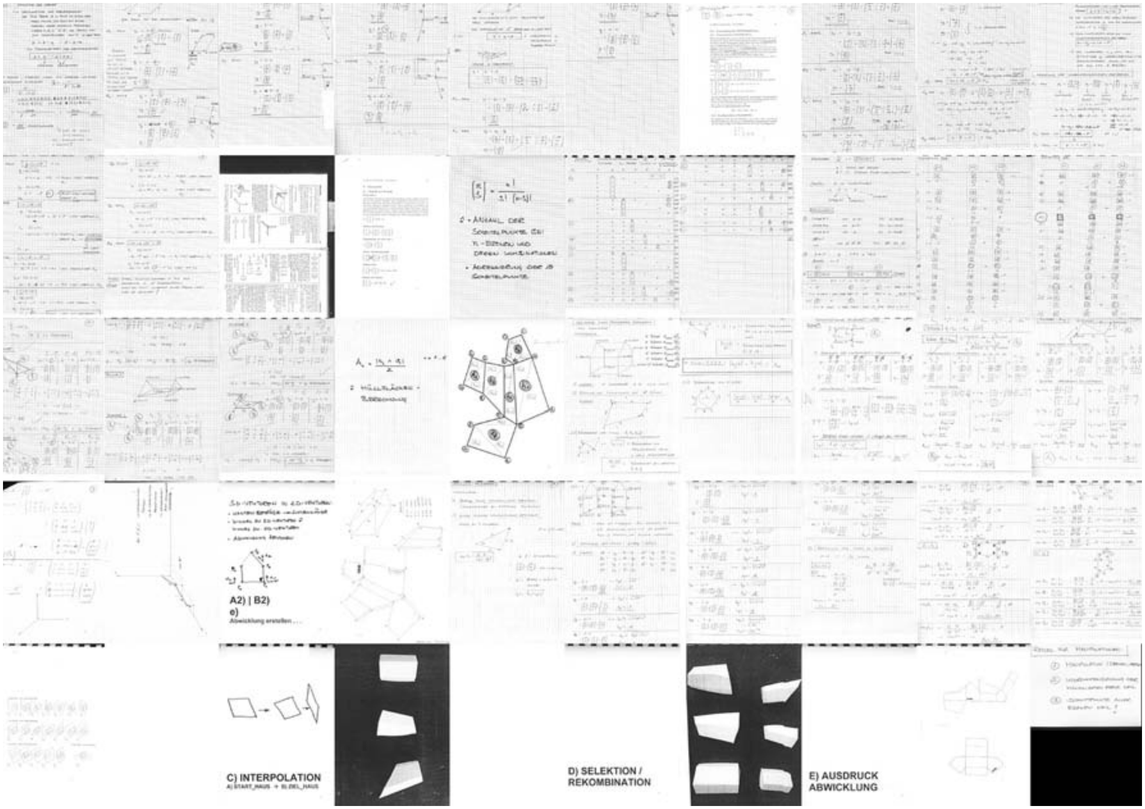
5) Database:

- Store and Retrieve of data, selection and recombination of stored morph-results – 'iterative optimization';
- < Export Model >: DXF, IGES;

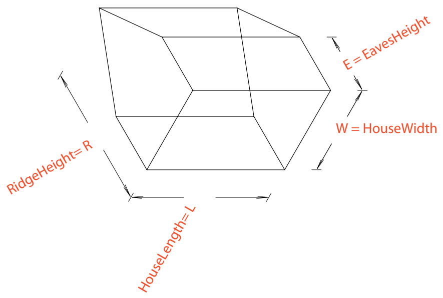
5.1) Database → < Print Unfolding >; in Version 1.0 not implemented;


A) START – HAUS
B) ZIEL – HAUS
A1) | B1)
EINGABE DER PARAMETER
A2) | B2)
AUFBAU EBENEN, MODELL...
A2) | B2)
a)
Berechnung von Kanten- und Normalenvektoren, Aufspannen der Ebenen ...
A2) | B2)
b)
Schneiden der Ebenen; Adressierung der Schnittpunkte
A2) | B2)
c)
3D Modell erstellen; Punkte zu Flächen verbinden ...
A2) | B2)
d)
Volumen und Hüllfläche (A/V) des Hauses ermitteln ...
A3) | B3)
EBENENMANIPULATION (NEIGEN + VERSCHIEBEN)
→ ZURUECK A2) | B2)
NEUBERECHNUNG MODELL

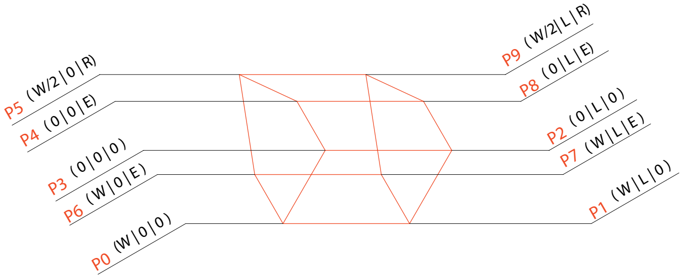
C4 Maths Fundamentals



C4.1 Input Parameters



Definition of a symetrical ridge roof house by four parameters;

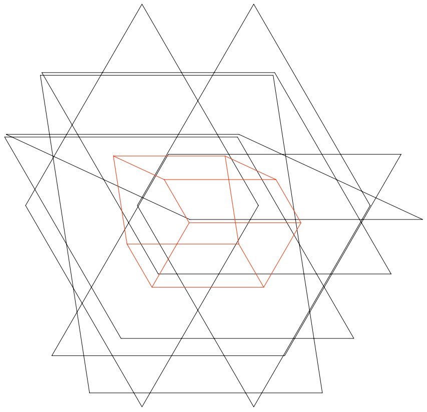


Each point is calculated in dependency of the input parameters;

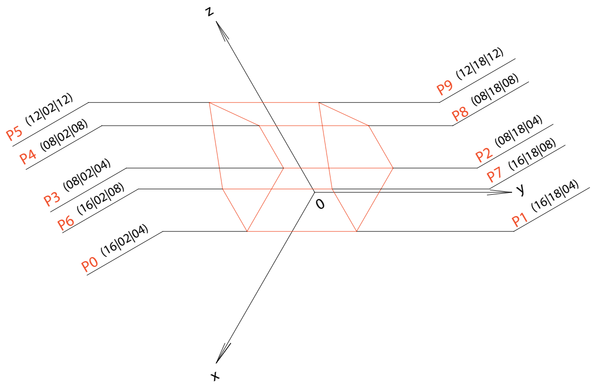
C4.2 Defintion of Planes

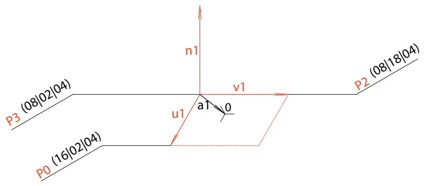
1. RULES:

• CoordinateFormula:	$ax + by + cz + d = 0;$
• Definition of Plane:	A Plane 'E' in space is defined by three points, not lying on a straight line. By the Points P0,P1,P2, two Vectors are calculable: $u = P1 - P0; \quad v = P2 - P0;$ With the VectorProduct of 'u $\otimes$ v', the perpendicular Vector 'n' to 'u' and 'v' is determined. Inserting 'n' in the CoordinateFormula, to get the CoordinateEquation of the Plane.



2. EXAMPLE: Coordinate Equation of Planes





E 1: *PLANE Po=(16,2,4), P1=(16,18,4), P2=(8,18,4), P3=(8,2,4);*

A: Calculation of Vectors :

u1 = Po – P3; v1 = P2 – P3;

$$u1 = \begin{vmatrix}16\\2\\4\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}8\\0\\0\end{vmatrix}; \qquad v1 = \begin{vmatrix}8\\18\\4\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix};$$

B: Product of Vectors:

$$n1: \Rightarrow u1 \otimes v1 = \begin{vmatrix}u1\\u2\\u3\end{vmatrix} \otimes \begin{vmatrix}v1\\v2\\v3\end{vmatrix} = \begin{vmatrix}u2v3 - & u3v2\\u3v1 - & u1v3\\u1v2 - & u2v1\end{vmatrix};$$

$$n1: \Rightarrow u1 \otimes v1 = \begin{vmatrix}8\\0\\0\end{vmatrix} \otimes \begin{vmatrix}0\\16\\0\end{vmatrix} = \begin{vmatrix}0 - & 0\\0 - & 0\\8*16 - & 0\end{vmatrix} = \begin{vmatrix}0\\0\\128\end{vmatrix};$$

C: CoordinateEquation of Plane E1:

$$n1 = \begin{vmatrix}0\\0\\128\end{vmatrix}; \qquad P3 = \begin{vmatrix}8\\2\\4\end{vmatrix};$$

⇒ n1 in ax + by + cz + d = 0;

⇒ 0x + 0y + 128z + d = 0;

⇒ P3 in 0x + 0y + 128z + d = 0;

⇒ (0*8) +(0*2) + (128*4) + d = 0;

⇒ d = -512; in 0x+ 0y + 128z + d = 0;

⇒ **0x + 0y + 128z –512 = 0;** E1:

E 2: *PLANE Po=(16,2,4), P1=(16,18,4), P7=(16,18,8), P6=(16,2,8);*

A: Calculation of Vectors :

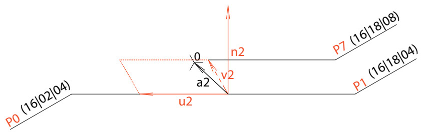
u2 = Po – P1; v2 = P7 – P1;

$u2 = \begin{vmatrix}16\\2\\4\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}0\\-16\\0\end{vmatrix};\quad v2 = \begin{vmatrix}16\\18\\8\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}0\\0\\4\end{vmatrix};$

B: Product of Vectors:

$n2: \Rightarrow u2 \otimes v2 = \begin{vmatrix}u1\\u2\\u3\end{vmatrix} \otimes \begin{vmatrix}v1\\v2\\v3\end{vmatrix} = \begin{vmatrix}u2v3 - & u3v2\\u3v1 - & u1v3\\u1v2 - & u2v1\end{vmatrix};$

$n2: \Rightarrow u2 \otimes v2 = \begin{vmatrix}0\\-16\\0\end{vmatrix} \otimes \begin{vmatrix}0\\0\\4\end{vmatrix} = \begin{vmatrix}(-16*4) - & 0\\0 - & 0\\0 - & 0\end{vmatrix} = \begin{vmatrix}-64\\0\\0\end{vmatrix};$



C: CoordinateEquation of Plane E2:

$n2 = \begin{vmatrix}-64\\0\\0\end{vmatrix};\qquad P1 = \begin{vmatrix}16\\18\\4\end{vmatrix};$

$\Rightarrow$ n2 in ax + by + cz + d = o;

$\Rightarrow$ -64x + oy + oz + d = o;

$\Rightarrow$ P1 in -64x + oy + oz + d = o;

$\Rightarrow$ (-64*16) + (o*18) + (o*4) + d = o;

$\Rightarrow$ d = 1024; in -64x + oy + oz + d = o;

$\Rightarrow$ **-64x + oy + oz + 1024 = o;** E2:

E 3: *PLANE P2=(8,18,4), P3=(8,2,4), P4=(8,2,8), P8=(8,18,8);*

A: Calculation of Vectors :

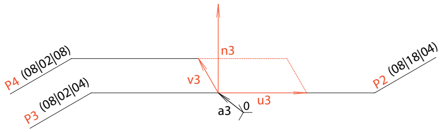
u3 = P2 – P3; v3 = P4 – P3;

$u3 = \begin{vmatrix}8\\18\\4\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix};\quad v3 = \begin{vmatrix}8\\2\\8\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}0\\0\\4\end{vmatrix};$

B: Product of Vectors:

$n3: \Rightarrow u3 \otimes v3 = \begin{vmatrix}u1\\u2\\u3\end{vmatrix} \otimes \begin{vmatrix}v1\\v2\\v3\end{vmatrix} = \begin{vmatrix}u2v3 - & u3v2\\u3v1 - & u1v3\\u1v2 - & u2v1\end{vmatrix};$

$n3: \Rightarrow u3 \otimes v3 = \begin{vmatrix}0\\16\\0\end{vmatrix} \otimes \begin{vmatrix}0\\0\\4\end{vmatrix} = \begin{vmatrix}(16*4) - & 0\\0 - & 0\\0 - & 0\end{vmatrix} = \begin{vmatrix}64\\0\\0\end{vmatrix};$



C: CoordinateEquation of Plane E3:

$n3 = \begin{vmatrix}64\\0\\0\end{vmatrix};\qquad P3 = \begin{vmatrix}8\\2\\4\end{vmatrix};$

$\Rightarrow$ n3 in ax + by + cz + d = o;

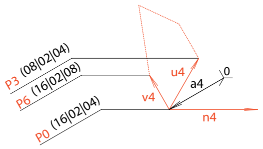
$\Rightarrow$ 64x + oy + oz + d = o;

$\Rightarrow$ P3 in 64x + oy + oz + d = o;

$\Rightarrow$ (64*8) + (o*18) + (o*4) + d = o;

$\Rightarrow$ d = -512; in 64x + oy + oz + d = o;

$\Rightarrow$ **64x + oy + oz - 512 = o;** E3:



E 4: *PLANE P3=(8,2,4), P0=(16,2,4), P6=(16,2,8), P5=(12,2,12); P4=(8,2,8);*

A: Calculation of Vectors :

u4 = P3 – P0; v4 = P6 – P0;

$$u4 = \begin{vmatrix} 8 \\ 2 \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ 2 \\ 4 \end{vmatrix} = \begin{vmatrix} -8 \\ 0 \\ 0 \end{vmatrix}; \quad v3 = \begin{vmatrix} 16 \\ 2 \\ 8 \end{vmatrix} - \begin{vmatrix} 16 \\ 2 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix};$$

B: Product of Vectors:

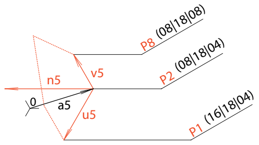
$$n4: \Rightarrow u4 \otimes v4 = \begin{vmatrix} u1 \\ u2 \\ u3 \end{vmatrix} \otimes \begin{vmatrix} v1 \\ v2 \\ v3 \end{vmatrix} = \begin{vmatrix} u2v3 - & u3v2 \\ u3v1 - & u1v3 \\ u1v2 - & u2v1 \end{vmatrix};$$

$$n4: \Rightarrow u4 \otimes v4 = \begin{vmatrix} -8 \\ 0 \\ 0 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 - & 0 \\ 0 - & (-8*4) \\ 0 - & 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 32 \\ 0 \end{vmatrix};$$

C: CoordinateEquation of Plane E4:

$$n4 = \begin{vmatrix} 0 \\ 32 \\ 0 \end{vmatrix}; \qquad P0 = \begin{vmatrix} 16 \\ 2 \\ 4 \end{vmatrix};$$

⇒ n4 in ax + by + cz + d = 0;
⇒ 0x + 32y + 0z + d = 0;
⇒ P0 in 0x + 32y + 0z + d = 0;
⇒ (0*16) + (32*2) + (0*4) + d = 0;
⇒ d = -64; in 0x + 32y + 0z + d = 0;
⇒ **0x + 32y + 0z - 64 = 0;** E4:



E 5: *PLANE P1=(16,18,4), P2=(8,18,4), P8=(8,18,8), P9=(12,18,12); P7=(16,18,8);*

A: Calculation of Vectors :

u5 = P1 – P2; v5 = P8 – P2;

$$u5 = \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix} = \begin{vmatrix} 8 \\ 0 \\ 0 \end{vmatrix}; \quad v5 = \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix};$$

B: Product of Vectors:

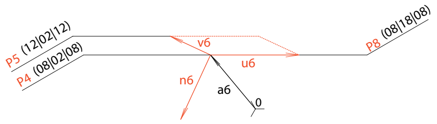
$$n5: \Rightarrow u5 \otimes v5 = \begin{vmatrix} u1 \\ u2 \\ u3 \end{vmatrix} \otimes \begin{vmatrix} v1 \\ v2 \\ v3 \end{vmatrix} = \begin{vmatrix} u2v3 - & u3v2 \\ u3v1 - & u1v3 \\ u1v2 - & u2v1 \end{vmatrix};$$

$$n5: \Rightarrow u5 \otimes v5 = \begin{vmatrix} 8 \\ 0 \\ 0 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 - & 0 \\ 0 - & (8*4) \\ 0 - & 0 \end{vmatrix} = \begin{vmatrix} 0 \\ -32 \\ 0 \end{vmatrix};$$

C: CoordinateEquation of Plane E5:

$$n5 = \begin{vmatrix} 0 \\ -32 \\ 0 \end{vmatrix}; \qquad P2 = \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix};$$

⇒ n5 in ax + by + cz + d = 0;
⇒ 0x - 32y + 0z + d = 0;
⇒ P2 in 0x - 32y + 0z + d = 0;
⇒ (0*8) - (32*18) + (0*4) + d = 0;
⇒ d = 576; in 0x - 32y + 0z + d = 0;
⇒ **0x - 32y + 0z + 576 = 0;** E5:



E 6: *PLANE P8=(8,18,8), P4=(8,2,8), P5=(12,2,12), P9=(12,18,12);*

A: Calculation of Vectors :

u6 = P8 – P4; v6 = P5 – P4;

$$u6 = \begin{vmatrix}8\\18\\8\end{vmatrix} - \begin{vmatrix}8\\2\\8\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix}; \quad v6 = \begin{vmatrix}12\\2\\12\end{vmatrix} - \begin{vmatrix}8\\2\\8\end{vmatrix} = \begin{vmatrix}4\\0\\4\end{vmatrix};$$

B: Product of Vectors:

$$n6: \Rightarrow u6 \otimes v6 = \begin{vmatrix}u1\\u2\\u3\end{vmatrix} \otimes \begin{vmatrix}v1\\v2\\v3\end{vmatrix} = \begin{vmatrix}u2v3- & u3v2\\u3v1- & u1v3\\u1v2- & u2v1\end{vmatrix};$$

$$n6: \Rightarrow u6 \otimes v6 = \begin{vmatrix}0\\16\\0\end{vmatrix} \otimes \begin{vmatrix}4\\0\\4\end{vmatrix} = \begin{vmatrix}(16*4)- & 0\\0- & 0\\0- & (16*4)\end{vmatrix} = \begin{vmatrix}64\\0\\-64\end{vmatrix};$$

C: CoordinateEquation of Plane E6:

$$n6 = \begin{vmatrix}64\\0\\-64\end{vmatrix}; \qquad P4 = \begin{vmatrix}8\\2\\8\end{vmatrix};$$

⇒ n6 in ax + by + cz + d = 0;

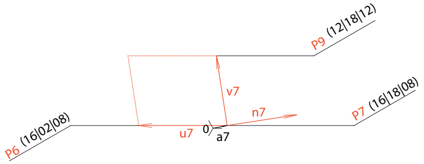
⇒ 64x + 0y - 64z + d = 0;

⇒ P4 in 64x + 0y - 64z + d = 0;

⇒ (64*8) + (0*2) - (64*8) + d = 0;

⇒ d = 0; in 64x + 0y - 64z + d = 0;

⇒ **64x + 0y - 64z + 0 = 0;** E6:



E 7: *PLANE P6=(16,2,8), P7=(16,18,8), P9=(12,18,12), P5=(12,2,12);*

A: Calculation of Vectors :

u7 = P6 – P7; v7 = P9 – P7;

$$u7 = \begin{vmatrix}16\\2\\8\end{vmatrix} - \begin{vmatrix}16\\18\\8\end{vmatrix} = \begin{vmatrix}0\\-16\\0\end{vmatrix}; \quad v7 = \begin{vmatrix}12\\18\\12\end{vmatrix} - \begin{vmatrix}16\\18\\8\end{vmatrix} = \begin{vmatrix}-4\\0\\4\end{vmatrix};$$

B: Product of Vectors:

$$n7: \Rightarrow u7 \otimes v7 = \begin{vmatrix}u1\\u2\\u3\end{vmatrix} \otimes \begin{vmatrix}v1\\v2\\v3\end{vmatrix} = \begin{vmatrix}u2v3- & u3v2\\u3v1- & u1v3\\u1v2- & u2v1\end{vmatrix};$$

$$n7: \Rightarrow u7 \otimes v7 = \begin{vmatrix}0\\-16\\0\end{vmatrix} \otimes \begin{vmatrix}-4\\0\\4\end{vmatrix} = \begin{vmatrix}(-16*4)- & 0\\0- & 0\\0- & (64)\end{vmatrix} = \begin{vmatrix}-64\\0\\-64\end{vmatrix};$$

C: CoordinateEquation of Plane E7:

$$n7 = \begin{vmatrix}-64\\0\\-64\end{vmatrix}; \qquad P7 = \begin{vmatrix}16\\18\\8\end{vmatrix};$$

⇒ n7 in ax + by + cz + d = 0;

⇒ -64x + 0y - 64z + d = 0;

⇒ P7 in -64x + 0y - 64z + d = 0;

⇒ (-64*16) + (0*18) - (64*8) + d = 0;

⇒ d = 1536; in -64x + 0y - 64z + d = 0;

⇒ **-64x + 0y - 64z + 1536 = 0;** E7:

C4.3 Calculation of Section Points

1. RULES:

- Definition SectionPoint: A SectionPoint is the blending-result of three planes

  E1: $a1x + b1y + c1z + d1 = 0;$
  E2: $a2x + b2y + c2z + d2 = 0;$
  E3: $a3x + b3y + c3z + d3 = 0;$

  A House defined by 7 Planes has 10 SectionPoints:

(1) P0: E1, E2, E4;	(2) P1: E1, E2, E5;
(3) P2: E1, E3, E5;	(4) P3: E1, E3, E4;
(5) P4: E3, E4, E6;	(6) P5: E4, E6, E7;
(7) P6: E2, E4, E7;	(8) P7: E2, E5, E7;
(9) P8: E3, E5, E6	(10) P9: E5, E6, E7;

- Calculation SectionPoint: = **A * x = b**

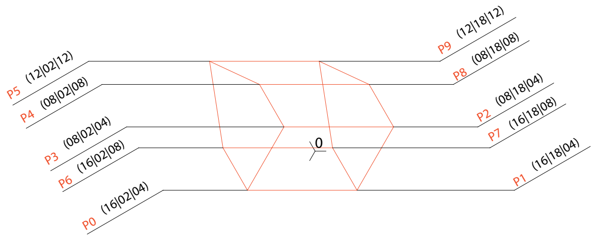
$$= \begin{vmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{vmatrix} * \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix} = \begin{vmatrix} d1 \\ d2 \\ d3 \end{vmatrix}$$

A = CoordinateMatrix; x = SectionPoint; b = SectionPoint Vector

- Equation Inverse Matrix: = **A^{-1} * b = x**

$$= 1/\,Det\,A \; * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} * \begin{vmatrix} d1 \\ d2 \\ d3 \end{vmatrix} = \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix}$$

2. EXAMPLE:



(1) Po: *E1:* $\Rightarrow$ 0x + 0y + 128z – 512 = 0;

E2: $\Rightarrow$ -64x + 0y + 0z + 1024 = 0;

E4: $\Rightarrow$ 0x + 32y + 0z – 64 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$Ao = \begin{vmatrix} 0 & 0 & 128 \\ -64 & 0 & 0 \\ 0 & 32 & 0 \end{vmatrix}; \qquad bo = \begin{vmatrix} 512 \\ -1024 \\ 64 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det Ao} = \begin{vmatrix} 0 & 0 & 128 \\ -64 & 0 & 0 \\ 0 & 32 & 0 \end{vmatrix} = 0 * \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} - (-64) * \begin{vmatrix} 0 & 128 \\ 32 & 0 \end{vmatrix} + 0 * \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix};$$

$$= 64 * (-32 * 128);$$

Det Ao = -262144 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det Ao } * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & 32 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & -64 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & -64 \end{vmatrix} \\ \begin{vmatrix} -64 & 0 \\ 0 & 32 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ +32 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & (128*32) & 0 \\ 0 & 0 & (128*-64) \\ (-64*32) & 0 & 0 \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & 4096 & 0 \\ 0 & 0 & -8192 \\ -2048 & 0 & 0 \end{vmatrix};$$

D: Calculation SectionPoint 'Po':

$$Po = 1/(-262144) * \begin{vmatrix} 0 & 4096 & 0 \\ 0 & 0 & -8192 \\ -2048 & 0 & 0 \end{vmatrix} * \begin{vmatrix} 512 \\ -1024 \\ 64 \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0+ & (-4194304)+ & 0 \\ 0+ & 0+ & (-524288) \\ -1048576 & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} -4194304 \\ -524288 \\ -1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 16 \\ 2 \\ 4 \end{vmatrix};$$

Po = (16,2,4)

(2) P1: *E1:* $\Rightarrow$ 0x + 0y + 128z – 512 = 0;

E2: $\Rightarrow$ -64x + 0y + 0z + 1024 = 0;

E5: $\Rightarrow$ 0x - 32y + 0z + 576 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$\text{A1} = \begin{vmatrix} 0 & 0 & 128 \\ -64 & 0 & 0 \\ 0 & -32 & 0 \end{vmatrix}; \qquad \text{b1} = \begin{vmatrix} 512 \\ -1024 \\ -576 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det A1} = \begin{vmatrix} 0 & 0 & 128 \\ -64 & 0 & 0 \\ 0 & -32 & 0 \end{vmatrix} = 0 * \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} - (-64) * \begin{vmatrix} 0 & 128 \\ -32 & 0 \end{vmatrix} + 0 * \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix};$$

$$= 64^{*} (-(-32 * 128)) ;$$

Det A1 = 262144 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det A1} * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & -32 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & -64 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & -64 \end{vmatrix} \\ \begin{vmatrix} -64 & 0 \\ 0 & -32 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(262144) * \begin{vmatrix} 0 & (128*-32) & 0 \\ 0 & 0 & (128*-64) \\ (-64*-32) & 0 & 0 \end{vmatrix};$$

$$\Rightarrow 1/(262144) * \begin{vmatrix} 0 & -4096 & 0 \\ 0 & 0 & -8192 \\ 2048 & 0 & 0 \end{vmatrix};$$

D: Calculation SectionPoint 'P1':

$$\text{P1} = 1/(262144) * \begin{vmatrix} 0 & -4096 & 0 \\ 0 & 0 & -8192 \\ 2048 & 0 & 0 \end{vmatrix} * \begin{vmatrix} 512 \\ -1024 \\ -576 \end{vmatrix};$$

$$\Rightarrow 1/(262144) * \begin{vmatrix} 0+ & (4194304)+ & 0 \\ 0+ & 0+ & (4718592) \\ (1048576)+ & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow 1/(262144) * \begin{vmatrix} 4194304 \\ 4718592 \\ 1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix};$$

P1 = (16,18,4)

(3) P2: *E1:* $\Rightarrow$ 0x + 0y + 128z – 512 = 0;

E3: $\Rightarrow$ 64x + 0y + 0z - 512 = 0;

E5: $\Rightarrow$ 0x - 32y + 0z + 576 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$\text{A2} = \begin{vmatrix} 0 & 0 & 128 \\ 64 & 0 & 0 \\ 0 & -32 & 0 \end{vmatrix}; \qquad \text{b2} = \begin{vmatrix} 512 \\ 512 \\ -576 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det A2} = \begin{vmatrix} 0 & 0 & 128 \\ 64 & 0 & 0 \\ 0 & -32 & 0 \end{vmatrix} = 0 * \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} - (64) * \begin{vmatrix} 0 & 128 \\ -32 & 0 \end{vmatrix} + 0 * \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix};$$

$$= -\ 64^{*} (-(-32 * 128)) ;$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & (128*-32) & 0 \\ 0 & 0 & (128*64) \\ (64*-32) & 0 & 0 \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & -4096 & 0 \\ 0 & 0 & 8192 \\ -2048 & 0 & 0 \end{vmatrix};$$

$\Rightarrow$ D: Calculation SectionPoint 'P2':

$$\Rightarrow \text{P2} = 1/(-262144) * \begin{vmatrix} 0 & -4096 & 0 \\ 0 & 0 & 8192 \\ -2048 & 0 & 0 \end{vmatrix} * \begin{vmatrix} 512 \\ 512 \\ -576 \end{vmatrix};$$

Det A2= - 262144 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\,\text{Det A2} \;*\; \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & -32 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 64 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & 64 \end{vmatrix} \\ \begin{vmatrix} 64 & 0 \\ 0 & -32 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 64 & 0 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow\; 1/(-262144) \;*\; \begin{vmatrix} 0+ & (-2097152)+ & 0 \\ 0+ & 0+ & (4718592) \\ (-1048576)+ & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow\; 1/(-262144) \;*\; \begin{vmatrix} -2097152 \\ 4718592 \\ -1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix};$$

P2 = (8,18,4)

(4) P3: *E1:* $\Rightarrow$ 0x + 0y + 128z - 512 = 0;

E3: $\Rightarrow$ 64x + 0y + 0z - 512 = 0;

E4: $\Rightarrow$ 0x + 32y + 0z - 64 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$\text{A3} = \begin{vmatrix} 0 & 0 & 128 \\ 64 & 0 & 0 \\ 0 & 32 & 0 \end{vmatrix}; \qquad \text{b3} = \begin{vmatrix} 512 \\ 512 \\ 64 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det A3} = \begin{vmatrix} 0 & 0 & 128 \\ 64 & 0 & 0 \\ 0 & 32 & 0 \end{vmatrix} = 0 * \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} - (64) * \begin{vmatrix} 0 & 128 \\ 32 & 0 \end{vmatrix} + 0 * \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix};$$

$$= -64 * (-(32 * 128));$$

Det A3= 262144 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\,\text{Det A3} \;*\; \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & 32 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 64 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 128 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 128 & 0 \\ 0 & 64 \end{vmatrix} \\ \begin{vmatrix} 64 & 0 \\ 0 & 32 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 64 & 0 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(262144) \;*\; \begin{vmatrix} 0 & (128 * 32) & 0 \\ 0 & 0 & (128 * 64) \\ (64 * 32) & 0 & 0 \end{vmatrix};$$

$$\Rightarrow\; 1/(262144) \;*\; \begin{vmatrix} 0 & 4096 & 0 \\ 0 & 0 & 8192 \\ 2048 & 0 & 0 \end{vmatrix};$$

$\Rightarrow$ D: Calculation SectionPoint 'P3':

$$\Rightarrow \text{P3} = 1/(262144) \;*\; \begin{vmatrix} 0 & 4096 & 0 \\ 0 & 0 & 8192 \\ 2048 & 0 & 0 \end{vmatrix} * \begin{vmatrix} 512 \\ 512 \\ 64 \end{vmatrix};$$

$$\Rightarrow\; 1/(262144) \;*\; \begin{vmatrix} 0+ & (2097152)+ & 0 \\ 0+ & 0+ & (524288) \\ (1048576)+ & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow\; 1/(262144) \;*\; \begin{vmatrix} 2097152 \\ 524288 \\ 1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 8 \\ 2 \\ 4 \end{vmatrix};$$

P3 = (8,2,4)

(5) P4: *E3:* $\Rightarrow$ 64x + 0y + 0z - 512 = 0;

E4: $\Rightarrow$ 0x + 32y + 0z - 64 = 0;

E6: $\Rightarrow$ 64x + 0y - 64z + 0 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$A4 = \begin{vmatrix} 64 & 0 & 0 \\ 0 & 32 & 0 \\ 64 & 0 & -64 \end{vmatrix}; \qquad b4 = \begin{vmatrix} 512 \\ 64 \\ 0 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det } A4 = \begin{vmatrix} 64 & 0 & 0 \\ 0 & 32 & 0 \\ 64 & 0 & -64 \end{vmatrix} = 64 * \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} - 0 * \begin{vmatrix} 0 & 0 \\ 0 & -64 \end{vmatrix} + 64 * \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix};$$

$$= 64 * (32 * -64) ;$$

Det A4= -131072 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det } A4 * \left\| \begin{matrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{matrix} \right\| = \left\| \begin{matrix} \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 0 \\ -64 & 64 \end{vmatrix} & \begin{vmatrix} 64 & 0 \\ 64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 64 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 32 \\ 64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 64 \\ 0 & 64 \end{vmatrix} & \begin{vmatrix} 64 & 0 \\ 0 & 32 \end{vmatrix} \end{matrix} \right\|$$

$$\Rightarrow \quad 1/(-131072\ ) * \begin{vmatrix} (32*-64) & 0 & 0 \\ 0 & (64*-64) & 0 \\ -(64*32) & 0 & (64*32) \end{vmatrix};$$

$$\Rightarrow 1/(-131072\ ) * \begin{vmatrix} -2048 & 0 & 0 \\ 0 & -4096 & 0 \\ -2048 & 0 & 2048 \end{vmatrix};$$

D: Calculation SectionPoint 'P4':

$$P4 = 1/(-131072) * \begin{vmatrix} -2048 & 0 & 0 \\ 0 & -4096 & 0 \\ -2048 & 0 & 2048 \end{vmatrix} * \begin{vmatrix} 512 \\ 64 \\ 0 \end{vmatrix};$$

$$\Rightarrow \quad 1/(-131072) * \begin{vmatrix} (-2048*512)+ & 0+ & 0 \\ 0+ & (-4096*64)+ & 0 \\ (-2048*512)+ & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow \quad 1/(-131072) * \begin{vmatrix} -1048576 \\ -262144 \\ -1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 8 \\ 2 \\ 8 \end{vmatrix};$$

P4 = (8,2,8)

(6) P5: *E4:* $\Rightarrow$ 0x + 32y + 0z - 64 = 0;

E6: $\Rightarrow$ 64x + 0y - 64z + 0 = 0;

E7: $\Rightarrow$ - 64x + 0y - 64z + 1536 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$A5 = \begin{vmatrix} 0 & 32 & 0 \\ 64 & 0 & -64 \\ -64 & 0 & -64 \end{vmatrix}; \qquad b5 = \begin{vmatrix} 64 \\ 0 \\ -1536 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det } A5 = \begin{vmatrix} 0 & 32 & 0 \\ 64 & 0 & -64 \\ -64 & 0 & -64 \end{vmatrix} = 0 * \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} - 64 * \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} + (-64) * \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix};$$

$$= -64 * (32 * -64) \ + (-64)*(32*-64) ;$$

$$\Rightarrow \quad 1/(\ 262144) * \begin{vmatrix} 0 & -(-64*32) & (32*-64) \\ (4096)-(-4096) & 0 & 0 \\ 0 & (32*-64) & -(64*32) \end{vmatrix};$$

$$\Rightarrow \quad 1/(262144) * \begin{vmatrix} 0 & 2048 & -2048 \\ 8192 & 0 & 0 \\ 0 & -2048 & -2048 \end{vmatrix};$$

D: Calculation SectionPoint 'P5':

$$P5 = 1/(262144) * \begin{vmatrix} 0 & 2048 & -2048 \\ 8192 & 0 & 0 \\ 0 & -2048 & -2048 \end{vmatrix} * \begin{vmatrix} 64 \\ 0 \\ -1536 \end{vmatrix};$$

Det A5= 262144 $\neq 0$ $\Rightarrow$ SectionPoint exists;

<u>C: InverseMatrix :</u>

$$1/\text{ Det A5 } * \left\| \begin{matrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{matrix} \right\| = \left\| \begin{matrix} \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 32 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} \\ \begin{vmatrix} -64 & 64 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 64 \end{vmatrix} \\ \begin{vmatrix} 64 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 32 \\ 64 & 0 \end{vmatrix} \end{matrix} \right\|$$

$$\Rightarrow \quad 1/(262144) * \begin{vmatrix} 0+ & 0+ & (-2048*-1536) \\ (8192*64)+ & 0+ & 0 \\ 0 & 0+ & (-2048*-1536) \end{vmatrix};$$

$$\Rightarrow \quad 1/(262144) * \begin{vmatrix} 3145728 \\ 524288 \\ 3145728 \end{vmatrix} \Rightarrow \begin{vmatrix} 12 \\ 2 \\ 12 \end{vmatrix};$$

P5 = (12,2,12)

(7) P6: *E2:* $\Rightarrow$ -64x + 0y + 0z + 1024 = 0;

E4: $\Rightarrow$ 0x + 32y + 0z - 64 = 0;

E7: $\Rightarrow$ - 64x + 0y - 64z + 1536 = 0;

<u>A: CoordinateMatrix (A) ; SectionVector (b) :</u>

$$\text{A6} = \begin{vmatrix} -64 & 0 & 0 \\ 0 & 32 & 0 \\ -64 & 0 & -64 \end{vmatrix}; \qquad \text{b6} = \begin{vmatrix} -1024 \\ 64 \\ -1536 \end{vmatrix};$$

<u>B: Determinat Coordinate Matrix (A) :</u>

$$\text{Det A6} = \begin{vmatrix} -64 & 0 & 0 \\ 0 & 32 & 0 \\ -64 & 0 & -64 \end{vmatrix} = -64 * \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} - 0 * \begin{vmatrix} 0 & 0 \\ 0 & -64 \end{vmatrix} + (-64) * \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix};$$

$$= -64 * (32 * -64);$$

Det A6= 131072 $\neq 0$ $\Rightarrow$ SectionPoint exists;

<u>C: InverseMatrix :</u>

$$1/\text{ Det A6 } * \left\| \begin{matrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{matrix} \right\| \quad \left\| \begin{matrix} \begin{vmatrix} 32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 32 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 32 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ 0 & 32 \end{vmatrix} \end{matrix} \right\|$$

$$\Rightarrow \quad 1/(131072) * \begin{vmatrix} (32*-64) & 0 & 0 \\ 0 & (-64)*(-64) & 0 \\ -(-64*32) & 0 & -(64*32) \end{vmatrix};$$

$$\Rightarrow \quad 1/(131072) * \begin{vmatrix} -2048 & 0 & 0 \\ 0 & 4096 & 0 \\ 2048 & 0 & -2048 \end{vmatrix};$$

<u>D: Calculation SectionPoint ’P6’:</u>

$$\text{P6} = 1/(131072) * \begin{vmatrix} -2048 & 0 & 0 \\ 0 & 4096 & 0 \\ 2048 & 0 & -2048 \end{vmatrix} * \begin{vmatrix} -1024 \\ 64 \\ -1536 \end{vmatrix};$$

$$\Rightarrow \quad 1/(131072) * \begin{vmatrix} (2097152)+ & 0+ & 0 \\ 0+ & (262144)+ & 0 \\ (-2097152)+ & 0+ & (3145728) \end{vmatrix};$$

$$\Rightarrow \quad 1/(131072) * \begin{vmatrix} 2097152 \\ 262144 \\ 1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 16 \\ 2 \\ 8 \end{vmatrix};$$

P6 = (16,2,8)

(8) P7: *E2:* $\Rightarrow$ -64x + 0y + 0z + 1024 = 0;

E5: $\Rightarrow$ 0x - 32y + 0z + 576 = 0;

E7: $\Rightarrow$ - 64x + 0y - 64z + 1536 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$A7 = \begin{vmatrix} -64 & 0 & 0 \\ 0 & -32 & 0 \\ -64 & 0 & -64 \end{vmatrix}; \qquad b7 = \begin{vmatrix} -1024 \\ -576 \\ -1536 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det } A7 = \begin{vmatrix} -64 & 0 & 0 \\ 0 & -32 & 0 \\ -64 & 0 & -64 \end{vmatrix} = -64 * \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} - 0 * \begin{vmatrix} 0 & 0 \\ 0 & -64 \end{vmatrix} + (-64) * \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix};$$

$$= -64 * (-32 * -64);$$

Det A7= -131072 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det } A7 * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & -32 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ 0 & -32 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(-131072) * \begin{vmatrix} (-32 * -64) & 0 & 0 \\ 0 & (-64) * (-64) & 0 \\ -(-64 * -32) & 0 & -(64 * -32) \end{vmatrix};$$

$$\Rightarrow 1/(-131072) * \begin{vmatrix} 2048 & 0 & 0 \\ 0 & 4096 & 0 \\ -2048 & 0 & 2048 \end{vmatrix};$$

D: Calculation SectionPoint 'P7':

$$P7 = 1/(-131072) * \begin{vmatrix} 2048 & 0 & 0 \\ 0 & 4096 & 0 \\ -2048 & 0 & 2048 \end{vmatrix} * \begin{vmatrix} -1024 \\ -576 \\ -1536 \end{vmatrix};$$

$$\Rightarrow 1/(-131072) * \begin{vmatrix} (-2097152) + & 0 + & 0 \\ 0 + & (-2359296) + & 0 \\ (2097152) + & 0 + & (-3145728) \end{vmatrix};$$

$$\Rightarrow 1/(-131072) * \begin{vmatrix} -2097152 \\ -2359296 \\ -1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 16 \\ 18 \\ 8 \end{vmatrix};$$

P7 = (16,18,8)

(9) P8: *E3:* $\Rightarrow$ 64x + 0y + 0z -512 = 0;

E5: $\Rightarrow$ 0x - 32y + 0z + 576 = 0;

E6: $\Rightarrow$ 64x + 0y - 64z + 0 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$A8 = \begin{vmatrix} 64 & 0 & 0 \\ 0 & -32 & 0 \\ 64 & 0 & -64 \end{vmatrix}; \qquad b8 = \begin{vmatrix} 512 \\ -576 \\ 0 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det } A8 = \begin{vmatrix} 64 & 0 & 0 \\ 0 & -32 & 0 \\ 64 & 0 & -64 \end{vmatrix} = 64 * \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} - 0 * \begin{vmatrix} 0 & 0 \\ 0 & -64 \end{vmatrix} + (64) * \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix};$$

$$= 64 * (-32 * -64);$$

$$\Rightarrow 1/(131072) * \begin{vmatrix} (-32 * -64) & 0 & 0 \\ 0 & (-64) * (64) & 0 \\ -(64 * -32) & 0 & (64 * -32) \end{vmatrix};$$

$$\Rightarrow 1/(131072) * \begin{vmatrix} 2048 & 0 & 0 \\ 0 & -4096 & 0 \\ 2048 & 0 & -2048 \end{vmatrix};$$

D: Calculation SectionPoint 'P8':

$$P8 = 1/(131072) * \begin{vmatrix} 2048 & 0 & 0 \\ 0 & -4096 & 0 \\ 2048 & 0 & -2048 \end{vmatrix} * \begin{vmatrix} 512 \\ -576 \\ 0 \end{vmatrix};$$

Det A8= 131072 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det A8 } * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -32 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 0 \\ -64 & 64 \end{vmatrix} & \begin{vmatrix} 64 & 0 \\ 64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 64 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & -32 \\ 64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 64 \\ 0 & 64 \end{vmatrix} & \begin{vmatrix} 64 & 0 \\ 0 & -32 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(131072) * \begin{vmatrix} (2048*512)+ & 0+ & 0 \\ 0+ & (-4096*-576)+ & 0 \\ (2048*512)+ & 0+ & 0 \end{vmatrix};$$

$$\Rightarrow 1/(131072) * \begin{vmatrix} 1048576 \\ 2359296 \\ 1048576 \end{vmatrix} \Rightarrow \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix};$$

P8 = (8,18,8)

(10)P9: *E5:* $\Rightarrow$ 0x - 32y + 0z + 576 = 0;

E6: $\Rightarrow$ 64x + 0y - 64z + 0 = 0;

E7: $\Rightarrow$ -64x + 0y + -64z + 1536 = 0;

A: CoordinateMatrix (A) ; SectionVector (b) :

$$A9 = \begin{vmatrix} 0 & -32 & 0 \\ 64 & 0 & -64 \\ -64 & 0 & -64 \end{vmatrix}; \qquad b9 = \begin{vmatrix} -576 \\ 0 \\ -1536 \end{vmatrix};$$

B: Determinat Coordinate Matrix (A) :

$$\text{Det A9} = \begin{vmatrix} 0 & -32 & 0 \\ 64 & 0 & -64 \\ -64 & 0 & -64 \end{vmatrix} = 0*\begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} -64*\begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} +(-64)*\begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix};$$

= - 64* (-32 * -64) + (-64)*(-32*-64);

Det A9= -262144 $\neq$ 0 $\Rightarrow$ SectionPoint exists;

C: InverseMatrix :

$$1/\text{ Det A9 } * \begin{Vmatrix} \begin{vmatrix} b2 & c2 \\ b3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & b1 \\ c3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & c1 \\ b2 & c2 \end{vmatrix} \\ \begin{vmatrix} c2 & a2 \\ c3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & c1 \\ a3 & c3 \end{vmatrix} & \begin{vmatrix} c1 & a1 \\ c2 & a2 \end{vmatrix} \\ \begin{vmatrix} a2 & b2 \\ a3 & b3 \end{vmatrix} & \begin{vmatrix} b1 & a1 \\ b3 & a3 \end{vmatrix} & \begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix} \end{Vmatrix} = \begin{Vmatrix} \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & -32 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} \\ \begin{vmatrix} -64 & 64 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 64 \end{vmatrix} \\ \begin{vmatrix} 64 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} -32 & 0 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & -32 \\ 64 & 0 \end{vmatrix} \end{Vmatrix}$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & -(-64*-32) & (-32*-64) \\ 8192 & 0 & 0 \\ 0 & (-32*-64) & -(64*-32) \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0 & -2048 & 2048 \\ 8192 & 0 & 0 \\ 0 & 2048 & 2048 \end{vmatrix};$$

D: Calculation SectionPoint 'P9':

$$P9 = 1/(-262144) * \begin{vmatrix} 0 & -2048 & 2048 \\ 8192 & 0 & 0 \\ 0 & 2048 & 2048 \end{vmatrix} * \begin{vmatrix} -576 \\ 0 \\ -1536 \end{vmatrix};$$

$$\Rightarrow 1/(-262144) * \begin{vmatrix} 0+ & 0+ & (2048*-1536) \\ (8192*-576)+ & 0+ & 0 \\ 0+ & 0+ & (2048*-1536) \end{vmatrix};$$

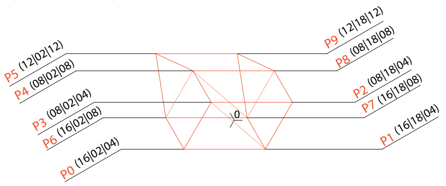
$$\Rightarrow 1/(-262144) * \begin{vmatrix} -3145728 \\ -4718592 \\ -3145728 \end{vmatrix} \Rightarrow \begin{vmatrix} 12 \\ 18 \\ 12 \end{vmatrix};$$

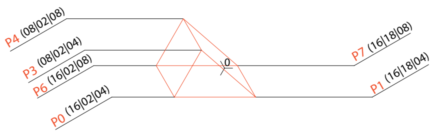
P9 = (12,18,12)

1. RULES:

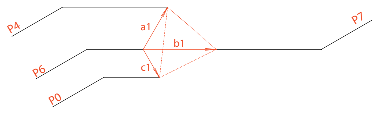
- Every house, with 10 sectionpoints, consits of 3 prisms.
- 1 Prism consists of 3 Tetrahedrons.
- VOLUME Tetrahedron: **V= 1/6 x det (a,b,c)**

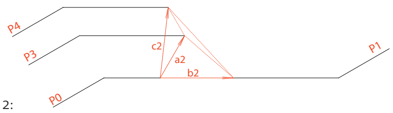
2. EXAMPLE:





PRISM I :





TETRAHEDRON 1:

A: Calculation Vectors:

a1 = P4 – P6 b1 = P7 - P6 c1 = P0 - P6

$$a1 = \begin{vmatrix} \mathbf{8} \\ \mathbf{2} \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} -\mathbf{8} \\ 0 \\ 0 \end{vmatrix}; \quad b1 = \begin{vmatrix} 16 \\ 18 \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix}; \quad c1 = \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ -4 \end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A1} = \begin{vmatrix} e0 & e1 & e2 \\ e3 & e4 & e5 \\ e6 & e7 & e8 \end{vmatrix} = \begin{vmatrix} -8 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 4 \end{vmatrix} = -8 * \begin{vmatrix} 16 & 0 \\ 0 & 4 \end{vmatrix} - 0 * \begin{vmatrix} 0 & 0 \\ 0 & 4 \end{vmatrix} + 0 * \begin{vmatrix} 0 & 0 \\ 16 & 0 \end{vmatrix}$$

Det A1 = - 8 * | 16*4| = $\underline{\mathbf{|512|;}}$

V1 = 1/6 * Det A1 = 1/6 * 512 = **85 1/3**

TETRAHEDRON 2:

A: Calculation Vectors:

a2 = P3 – P0 b2 = P 1 - P0 c2 = P4 - P0

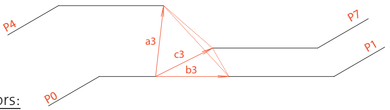
$$a2 = \begin{vmatrix} \mathbf{8} \\ \mathbf{2} \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} -\mathbf{8} \\ 0 \\ 0 \end{vmatrix}; \quad b2 = \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix}; \quad c2 = \begin{vmatrix} 8 \\ \mathbf{2} \\ 8 \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} -8 \\ 0 \\ 4 \end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A2} = \begin{vmatrix} e0 & e1 & e2 \\ e3 & e4 & e5 \\ e6 & e7 & e8 \end{vmatrix} = \begin{vmatrix} -8 & 0 & -8 \\ 0 & 16 & 0 \\ 0 & 0 & 4 \end{vmatrix} = -8 * \begin{vmatrix} 16 & 0 \\ 0 & 4 \end{vmatrix} - 0 * \begin{vmatrix} 0 & -8 \\ 0 & 4 \end{vmatrix} + 0 * \begin{vmatrix} 0 & -8 \\ 16 & 0 \end{vmatrix}$$

Det A2 = - 8 * | 16*4| = $\underline{\mathbf{|512|}}$

V2 = 1/6 * Det A2 = 1/6 * 512 = **85 1/3**



TETRAHEDRON 3:

A: Calculation Vectors:

a3 = P4 – P0 b3 = P 1 - P0 c3 = P7 - P0

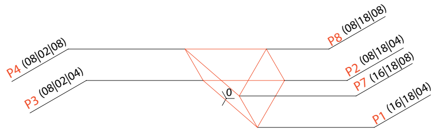
$$a3 = \begin{vmatrix}\mathbf{8}\\\mathbf{2}\\8\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\4\end{vmatrix} = \begin{vmatrix}\mathbf{-8}\\0\\4\end{vmatrix};\quad b3 = \begin{vmatrix}16\\18\\4\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\4\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix};\quad c3 = \begin{vmatrix}16\\18\\8\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\4\end{vmatrix} = \begin{vmatrix}0\\16\\4\end{vmatrix};$$

B: Calculation Determinant of Matrix:

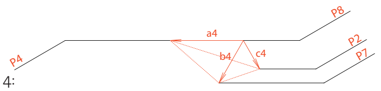
$$\text{Det A3} = \begin{vmatrix}e0 & e1 & e2\\e3 & e4 & e5\\e6 & e7 & e8\end{vmatrix} = \begin{vmatrix}-8 & 0 & 0\\0 & 16 & 16\\4 & 0 & 4\end{vmatrix} = -8^*\begin{vmatrix}16 & 16\\0 & 4\end{vmatrix} - 0^*\begin{vmatrix}0 & 0\\0 & 4\end{vmatrix} + 0^*\begin{vmatrix}0 & 0\\16 & 16\end{vmatrix}$$

Det A3 = - 8 * | 16*4| = |**512**|

V3 = 1/6 * Det A3 = 1/6 * 512 = **85 1/3**



PRISM II :



TETRAHEDRON 4:

A: Calculation Vectors:

a4 = P4 – P8 b4 = P7 - P8 c4 = P2 - P8

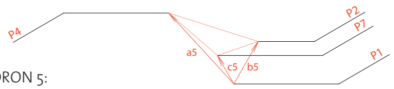
$$a4 = \begin{vmatrix}\mathbf{8}\\\mathbf{2}\\\mathbf{8}\end{vmatrix} - \begin{vmatrix}8\\18\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}0\\-16\\0\end{vmatrix};\quad b4 = \begin{vmatrix}16\\18\\\mathbf{8}\end{vmatrix} - \begin{vmatrix}8\\18\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}8\\0\\0\end{vmatrix};\quad c4 = \begin{vmatrix}8\\18\\4\end{vmatrix} - \begin{vmatrix}8\\18\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}0\\0\\-4\end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A4} = \begin{vmatrix}0 & 8 & 0\\-16 & 0 & 0\\0 & 0 & -4\end{vmatrix} = o^*\begin{vmatrix}0 & 0\\0 & -4\end{vmatrix} - |-16|^*\begin{vmatrix}8 & 0\\0 & -4\end{vmatrix} + o^*\begin{vmatrix}8 & 0\\0 & 0\end{vmatrix}$$

Det A4 = 16* | 8 * - 4| = |**512**|;

V4 = 1/6 * Det A4 = 1/6 * 512 = **85 1/3**



TETRAHEDRON 5:

A: Calculation Vectors:

a5 = P4 – P1 b5 = P2 - P1 c5 = P7 - P1

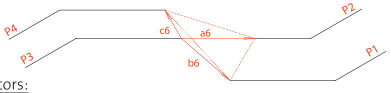
$$a5 = \begin{vmatrix}\mathbf{8}\\\mathbf{2}\\\mathbf{8}\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}-8\\-16\\4\end{vmatrix};\quad b5 = \begin{vmatrix}8\\18\\4\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}-8\\0\\0\end{vmatrix};\ c5 = \begin{vmatrix}16\\18\\8\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}0\\0\\4\end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A5} = \begin{vmatrix}-8 & -8 & 0\\-16 & 0 & 0\\4 & 0 & 4\end{vmatrix} = -8^*\begin{vmatrix}0 & 0\\0 & 4\end{vmatrix} - |-16|^*\begin{vmatrix}-8 & 0\\0 & 4\end{vmatrix} + 4^*\begin{vmatrix}-8 & 0\\0 & 0\end{vmatrix}$$

Det A5 = 16* | -8 * -4| = |**512**|;

V5 = 1/6 * Det A5 = 1/6 * 512 = **85 1/3**



TETRAHEDRON 6:

A: Calculation Vectors:

a6 = P2 – P3 b6 = P1 - P3 c6 = P4 - P3

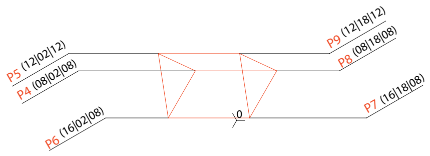
$$a6 = \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix}; \quad b6 = \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ 4 \end{vmatrix} = \begin{vmatrix} 8 \\ 16 \\ 0 \end{vmatrix}; \quad c6 = \begin{vmatrix} 8 \\ 2 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix};$$

B: Calculation Determinant of Matrix:

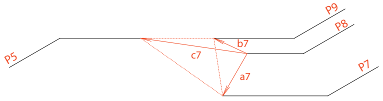
$$\text{Det A6} = \begin{vmatrix} 0 & 8 & 0 \\ 16 & 16 & 0 \\ 0 & 0 & 4 \end{vmatrix} = 0 * \begin{vmatrix} 0 & 0 \\ 0 & 4 \end{vmatrix} - 16 * \begin{vmatrix} 8 & 0 \\ 0 & 4 \end{vmatrix} + 4 * \begin{vmatrix} 8 & 0 \\ 16 & 0 \end{vmatrix}$$

Det A6 = - 16 * | 8 * 4| = **|512|;**

V6 = 1/6 * Det A6 = 1/6 * 512 = **85 1/3**



PRISM III:



TETRAHEDRON 7:

A: Calculation Vectors:

a7 = P7 – P8 b7 = P9 - P8 c7 = P5 - P8

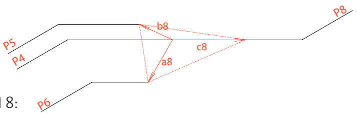
$$a7 = \begin{vmatrix} 16 \\ 18 \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 8 \\ 0 \\ 0 \end{vmatrix}; \quad b7 = \begin{vmatrix} 12 \\ 18 \\ 12 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 4 \\ 0 \\ 4 \end{vmatrix}; \quad c7 = \begin{vmatrix} 12 \\ 2 \\ 12 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 4 \\ -16 \\ 4 \end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A7} = \begin{vmatrix} 8 & 4 & 4 \\ 0 & 0 & -16 \\ 0 & 4 & 4 \end{vmatrix} = 8 * \begin{vmatrix} 0 & -16 \\ 4 & 4 \end{vmatrix} - 0 * \begin{vmatrix} 4 & 4 \\ 4 & 4 \end{vmatrix} + 0 * \begin{vmatrix} 4 & 4 \\ 0 & -16 \end{vmatrix}$$

Det A7 = 8 * | 4 * - 16| = **|512|;**

V7 = 1/6 * Det A7 = 1/6 * 512 = **85 1/3**



TETRAHEDRON 8:

A: Calculation Vectors:

a8 = P6 – P4 b8 = P5 - P4 c8 = P8 - P4

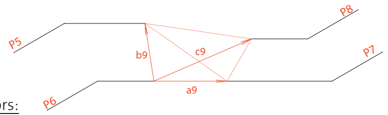
$$a8 = \begin{vmatrix} 16 \\ 2 \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 8 \\ 0 \\ 0 \end{vmatrix}; \quad b8 = \begin{vmatrix} 12 \\ 2 \\ 12 \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 4 \\ 0 \\ 4 \end{vmatrix}; \quad c8 = \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ 2 \\ \mathbf{8} \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det A8} = \begin{vmatrix} 8 & 4 & 0 \\ 0 & 0 & 16 \\ 0 & 4 & 0 \end{vmatrix} = 8 * \begin{vmatrix} 0 & 16 \\ 4 & 0 \end{vmatrix} - 0 * \begin{vmatrix} 4 & 0 \\ 4 & 0 \end{vmatrix} + 0 * \begin{vmatrix} 4 & 0 \\ 0 & 16 \end{vmatrix}$$

Det A8 = 8 * | 4 * 16| = **|512|;**

V8 = 1/6 * Det A8 = 1/6 * 512 = **85 1/3**



TETRAHEDRON 9:

A: Calculation Vectors:

a9 = P7 – P6 b9 = P5 - P6 c9 = P8 - P6

$$a9 = \begin{vmatrix}16\\18\\\mathbf{8}\end{vmatrix} - \begin{vmatrix}16\\2\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix}; \quad b9 = \begin{vmatrix}12\\2\\12\end{vmatrix} - \begin{vmatrix}16\\2\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}-4\\0\\4\end{vmatrix}; \quad c9 = \begin{vmatrix}8\\18\\8\end{vmatrix} - \begin{vmatrix}16\\2\\\mathbf{8}\end{vmatrix} = \begin{vmatrix}-8\\16\\0\end{vmatrix};$$

B: Calculation Determinant of Matrix:

$$\text{Det } A9 = \begin{vmatrix}0 & -4 & -8\\16 & 0 & 16\\0 & 4 & 0\end{vmatrix} = 0 * \begin{vmatrix}0 & 16\\4 & 0\end{vmatrix} - 16 * \begin{vmatrix}4 & -8\\4 & 0\end{vmatrix} + 0 * \begin{vmatrix}4 & -8\\0 & 16\end{vmatrix}$$

Det A9 = -16 * | 4 * -8| = |**512|;**

V9 = 1/6 * Det A9 = 1/6 * 512 = **85 1/3**

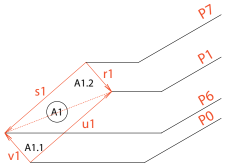
TOTAL VOLUME : =V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9

85 1/3+ 85 1/3+85 1/3+85 1/3+85 1/3+85 1/3+85 1/3+85 1/3+85 1/3 = **768**

C4.5 Calculation of Surface Quadrature

1. RULES:

- Every House with 10 Scectionpoints has 6 Surfaces .
- 1 Surface consists of 2 –3 Triangles.
- Area TRIANGLE: A= (u ⊗ v) / 2
- VectorProduct: $u \otimes v = \begin{vmatrix} u1 \\ u2 \\ u3 \end{vmatrix} \otimes \begin{vmatrix} v1 \\ v2 \\ v3 \end{vmatrix} = \begin{vmatrix} u2v3 \\ u3v1 \\ u1v2 \end{vmatrix} - \begin{vmatrix} u3v2 \\ u1v3 \\ u2v1 \end{vmatrix}$;
- Sum of VectorProduct: $|u| = \sqrt{(u1)^2+(u2)^2+(u3)^2}$



2. EXAMPLE:

A 1:

TRIANGLE A1.1 + TRIANGLE A1.2

A: Calculation Vectors:

u1 = P1 – P0 v1 = P6 - P0 r1 = P1 - P7 s1 = P6 - P7

$u1 = \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix}$; $v1 = \begin{vmatrix} 16 \\ 2 \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 16 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix}$; $r1 = \begin{vmatrix} 16 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 16 \\ 18 \\ 8 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ -4 \end{vmatrix}$; $s1 = \begin{vmatrix} 16 \\ \mathbf{2} \\ 8 \end{vmatrix} - \begin{vmatrix} 16 \\ 18 \\ 8 \end{vmatrix} = \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix}$;

B: Product of Vectors:

$u1 \otimes v1 = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix} = \begin{vmatrix} 16*4 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 64 \\ 0 \\ 0 \end{vmatrix}$;

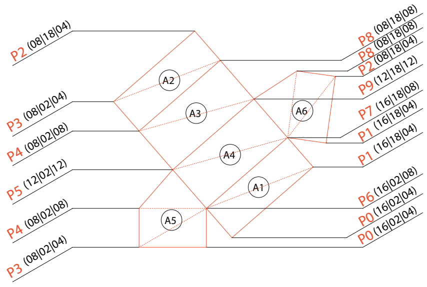
$r1 \otimes s1 = \begin{vmatrix} 0 \\ 0 \\ -4 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} -4*-16 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} -64 \\ 0 \\ 0 \end{vmatrix}$;

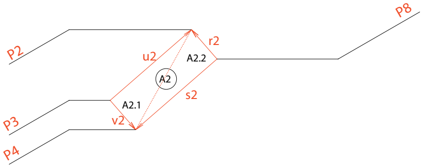
C: Sum of VectorProduct:

A1.1 : $|u1 \otimes v1| = \sqrt{(64)^2+(0)^2+(0)^2}$ = 64/2 = 32 ;

A1.2 : $|r1 \otimes s1| = \sqrt{(-64)^2+(0)^2+(0)^2}$ = 64/2 = 32 ;

A1.1 + A1.2 = 32 + 32 = **64**





A 2:

TRIANGLE A2.1 + TRIANGLE A2.2

A: Calculation Vectors:

u2 = P2 – P3 v2 = P4 - P3 r2 = P2 - P8 s2 = P4 - P8

$$u2 = \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 4 \end{vmatrix} ; v2 = \begin{vmatrix} 8 \\ 2 \\ \mathbf{8} \end{vmatrix} - \begin{vmatrix} 8 \\ \mathbf{2} \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix} ; r2 = \begin{vmatrix} 8 \\ 18 \\ 4 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ -4 \end{vmatrix} ; s2 = \begin{vmatrix} 8 \\ 2 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} = \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix} ;$$

B: Product of Vectors:

$$u2 \otimes v2 = \begin{vmatrix} 0 \\ 16 \\ 4 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ 0 \\ 4 \end{vmatrix} = \begin{vmatrix} 16*4 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 64 \\ 0 \\ 0 \end{vmatrix} ;$$

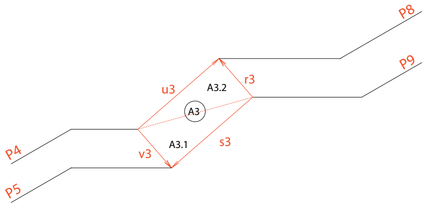
$$r2 \otimes s2 = \begin{vmatrix} 0 \\ 0 \\ -4 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} -4*-16 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} -64 \\ 0 \\ 0 \end{vmatrix} ;$$

C: Sum of VectorProduct:

$A2.1 : |u2 \otimes v2| = \sqrt{(64)\hat{}2+(0)\hat{}2+(0)\hat{}2} = 64/2 = 32 ;$

$A2.2 : |r2 \otimes s2| = \sqrt{(-64)\hat{}2+(0)\hat{}2+(0)\hat{}2} = 64/2 = 32 ;$

A2.1 + A2.2 = 32 + 32 = **64**



A 3:

TRIANGLE A3.1 + TRIANGLE A3.2

A: Calculation Vectors:

u3 = P8 – P4 v3 = P5 - P4 r3 = P8 - P9 s3 = P5 - P9

$$u3 = \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} - \begin{vmatrix} 8 \\ \mathbf{2} \\ 8 \end{vmatrix} = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix} ; v3 = \begin{vmatrix} 12 \\ 2 \\ 12 \end{vmatrix} - \begin{vmatrix} 8 \\ \mathbf{2} \\ 8 \end{vmatrix} = \begin{vmatrix} 4 \\ 0 \\ 4 \end{vmatrix} ; r3 = \begin{vmatrix} 8 \\ 18 \\ 8 \end{vmatrix} - \begin{vmatrix} 12 \\ 18 \\ 12 \end{vmatrix} = \begin{vmatrix} -4 \\ 0 \\ -4 \end{vmatrix} ; s3 = \begin{vmatrix} 12 \\ 2 \\ 12 \end{vmatrix} - \begin{vmatrix} 12 \\ 18 \\ 12 \end{vmatrix} = \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix} ;$$

B: Product of Vectors:

$$u3 \otimes v3 = \begin{vmatrix} 0 \\ 16 \\ 0 \end{vmatrix} \otimes \begin{vmatrix} 4 \\ 0 \\ 4 \end{vmatrix} = \begin{vmatrix} 16*4 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 0 \\ 0 \\ 16*4 \end{vmatrix} = \begin{vmatrix} 64 \\ 0 \\ -64 \end{vmatrix} ;$$

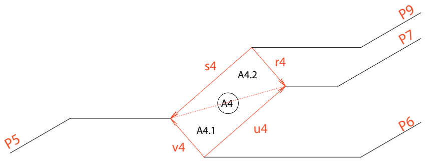
$$r3 \otimes s3 = \begin{vmatrix} -4 \\ 0 \\ -4 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ -16 \\ 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ -4*-16 \end{vmatrix} - \begin{vmatrix} -4*-16 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} -64 \\ 0 \\ 64 \end{vmatrix} ;$$

C: Sum of VectorProduct:

$A3.1 :| u3 \otimes v3| = \sqrt{(64)\hat{}2+(0)\hat{}2+(-64)\hat{}2} = 90.5/2 = 45.25 ;$

$A3.2 :| r3 \otimes s3| = \sqrt{(-64)\hat{}2+(0)\hat{}2+(64)\hat{}2} = 90.5/2 = 45.25 ;$

A3.1 + A3.2 = 45.25 + 45.25 = **90.5**



A 4:

TRIANGLE A4.1 + TRIANGLE A4.2

<u>A: Calculation Vectors:</u>

u4 = P7 – P6 v4 = P5 - P6 r4 = P7 - P9 s4 = P5 - P9

$$u4 = \begin{vmatrix}16\\18\\8\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\8\end{vmatrix} = \begin{vmatrix}0\\16\\0\end{vmatrix};\ v4 = \begin{vmatrix}12\\2\\12\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\8\end{vmatrix} = \begin{vmatrix}-4\\0\\4\end{vmatrix};\ r4 = \begin{vmatrix}16\\18\\8\end{vmatrix} - \begin{vmatrix}12\\18\\12\end{vmatrix} = \begin{vmatrix}4\\0\\-4\end{vmatrix};\ s4 = \begin{vmatrix}12\\2\\12\end{vmatrix} - \begin{vmatrix}12\\18\\12\end{vmatrix} = \begin{vmatrix}0\\-16\\0\end{vmatrix};$$

<u>B: Product of Vectors:</u>

$$u4 \otimes v4 = \begin{vmatrix}0\\16\\0\end{vmatrix} \otimes \begin{vmatrix}-4\\0\\4\end{vmatrix} = \begin{vmatrix}16*4\\0\\0\end{vmatrix} - \begin{vmatrix}0\\0\\16*-4\end{vmatrix} = \begin{vmatrix}64\\0\\64\end{vmatrix};$$

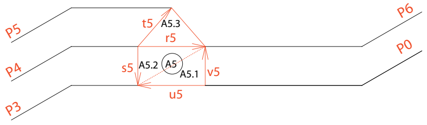
$$r4 \otimes s4 = \begin{vmatrix}4\\0\\-4\end{vmatrix} \otimes \begin{vmatrix}0\\-16\\0\end{vmatrix} = \begin{vmatrix}0\\0\\4*-16\end{vmatrix} - \begin{vmatrix}-4*-16\\0\\0\end{vmatrix} = \begin{vmatrix}-64\\0\\-64\end{vmatrix};$$

<u>C: Sum of VectorProduct:</u>

$$A4.1 :|\ u4 \otimes v4| = \sqrt{(64)^\wedge 2 + (0)^\wedge 2 + (64)^\wedge 2} = 90.5/2 = 45.25;$$

$$A4.2 :|\ r4 \otimes s4| = \sqrt{(-64)^\wedge 2 + (0)^\wedge 2 + (-64)^\wedge 2} = 90.5/2 = 45.25;$$

A4.1 + A4.2 = 45.25 + 45.25 = **90.5**



A 5:

TRIANGLE A5.1 + TRIANGLE A5.2 + TRIANGLE A5.3

<u>A: Calculation Vectors:</u>

u5 = P3 – Po v5 = P6 - Po r5 = P3 - P4 s5 = P6 - P4 t5 = P5 - P4

$$u5 = \begin{vmatrix}8\\2\\4\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\4\end{vmatrix} = \begin{vmatrix}-8\\0\\0\end{vmatrix};v5 = \begin{vmatrix}16\\2\\8\end{vmatrix} - \begin{vmatrix}16\\\mathbf{2}\\4\end{vmatrix} = \begin{vmatrix}0\\0\\4\end{vmatrix};r5 = \begin{vmatrix}8\\2\\4\end{vmatrix} - \begin{vmatrix}8\\2\\8\end{vmatrix} = \begin{vmatrix}0\\0\\-4\end{vmatrix};s5 = \begin{vmatrix}16\\2\\8\end{vmatrix} - \begin{vmatrix}8\\2\\8\end{vmatrix} = \begin{vmatrix}8\\0\\0\end{vmatrix};t5 = \begin{vmatrix}12\\2\\12\end{vmatrix} - \begin{vmatrix}8\\2\\8\end{vmatrix} = \begin{vmatrix}4\\0\\4\end{vmatrix};$$

<u>B: Product of Vectors:</u>

$$u5 \otimes v5 = \begin{vmatrix}-8\\0\\0\end{vmatrix} \otimes \begin{vmatrix}0\\0\\4\end{vmatrix} = \begin{vmatrix}0\\0\\0\end{vmatrix} - \begin{vmatrix}0\\-8*4\\0\end{vmatrix} = \begin{vmatrix}0\\32\\0\end{vmatrix};$$

$$r5 \otimes s5 = \begin{vmatrix}0\\0\\-4\end{vmatrix} \otimes \begin{vmatrix}8\\0\\0\end{vmatrix} = \begin{vmatrix}0\\-4*8\\0\end{vmatrix} - \begin{vmatrix}0\\0\\0\end{vmatrix} = \begin{vmatrix}0\\-32\\0\end{vmatrix};$$

$$t5 \otimes s5 = \begin{vmatrix}4\\0\\4\end{vmatrix} \otimes \begin{vmatrix}8\\0\\0\end{vmatrix} = \begin{vmatrix}0\\4*8\\0\end{vmatrix} - \begin{vmatrix}0\\0\\0\end{vmatrix} = \begin{vmatrix}0\\32\\0\end{vmatrix};$$

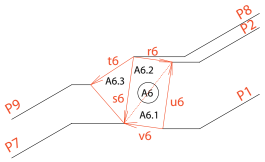
<u>C: Sum of VectorProduct:</u>

$$A5.1 :|\ u5 \otimes v5| = \sqrt{(0)^\wedge 2 + (32)^\wedge 2 + (0)^\wedge 2} = 32/2 = 16.0;$$

$$A5.2 :|\ r5 \otimes s5| = \sqrt{(0)^\wedge 2 + (-32)^\wedge 2 + (0)^\wedge 2} = 32/2 = 16.0;$$

$$A5.2 :|\ t5 \otimes s5| = \sqrt{(0)^\wedge 2 + (32)^\wedge 2 + (0)^\wedge 2} = 32/2 = 16.0;$$

A5.1 + A5.2+A5.3 = 16.0 + 16.0 + 16.0 = **48.0**



A 6:

TRIANGLE A6.1 + TRIANGLE A6.2 + TRIANGLE A6.3

A: Calculation Vectors:

u6 = P2 – P1   v6 = P7 - P1   r6 = P2 - P8   s6 = P7 - P8   t6 = P9 - P8

$$u6=\begin{vmatrix}8\\18\\4\end{vmatrix}-\begin{vmatrix}16\\18\\4\end{vmatrix}=\begin{vmatrix}-8\\0\\0\end{vmatrix};\ v6=\begin{vmatrix}16\\18\\8\end{vmatrix}-\begin{vmatrix}16\\18\\4\end{vmatrix}=\begin{vmatrix}0\\0\\4\end{vmatrix};\ r6=\begin{vmatrix}8\\18\\4\end{vmatrix}-\begin{vmatrix}8\\18\\8\end{vmatrix}=\begin{vmatrix}0\\0\\-4\end{vmatrix};\ s6=\begin{vmatrix}16\\18\\8\end{vmatrix}-\begin{vmatrix}8\\18\\8\end{vmatrix}=\begin{vmatrix}8\\0\\0\end{vmatrix};\ t6=\begin{vmatrix}12\\18\\12\end{vmatrix}-\begin{vmatrix}8\\18\\8\end{vmatrix}=\begin{vmatrix}4\\0\\4\end{vmatrix};$$

B: Product of Vectors:

$$u6\otimes v6=\begin{vmatrix}-8\\0\\0\end{vmatrix}\otimes\begin{vmatrix}0\\0\\4\end{vmatrix}=\begin{vmatrix}0\\0\\0\end{vmatrix}-\begin{vmatrix}0\\-8*4\\0\end{vmatrix}=\begin{vmatrix}0\\32\\0\end{vmatrix};$$

$$r6\otimes s6=\begin{vmatrix}0\\0\\-4\end{vmatrix}\otimes\begin{vmatrix}8\\0\\0\end{vmatrix}=\begin{vmatrix}0\\-4*8\\0\end{vmatrix}-\begin{vmatrix}0\\0\\0\end{vmatrix}=\begin{vmatrix}0\\-32\\0\end{vmatrix};$$

$$t6\otimes s6=\begin{vmatrix}4\\0\\4\end{vmatrix}\otimes\begin{vmatrix}8\\0\\0\end{vmatrix}=\begin{vmatrix}0\\4*8\\0\end{vmatrix}-\begin{vmatrix}0\\0\\0\end{vmatrix}=\begin{vmatrix}0\\32\\0\end{vmatrix};$$

C: Sum of VectorProduct:

A6.1 :$|u6\otimes v6|=\sqrt{(0)^\wedge 2+(32)^\wedge 2+(0)^\wedge 2}$ =32/2= 16.0 ;

A6.2:$|r6\otimes s6|=\sqrt{(0)^\wedge 2+(-32)^\wedge 2+(0)^\wedge 2}$ =32/2= 16.0 ;

A6.2:$|t6\otimes s6|=\sqrt{(0)^\wedge 2+(32)^\wedge 2+(0)^\wedge 2}$ =32/2= 16.0 ;

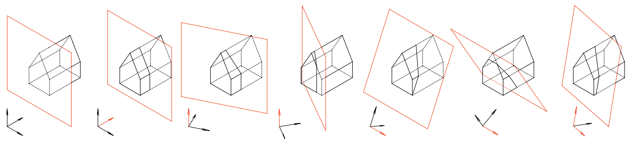
A6.1 + A6.2+A6.3 = 16.0 + 16.0 + 16.0 = **48.0**

TOTAL SURFACE : =A1 + A2 + A3 + A4 + A5 + A6

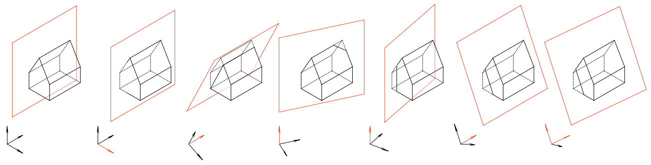
64.0+64.0+90.5+90.5+48.0+48.0 = **405.0**

C4.6 Plane Manipulation MOVING and ROTATING

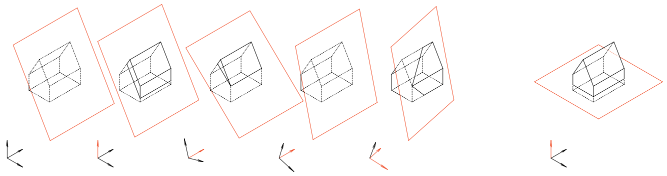
gable wall - rotation-/moving axis



eaves wall - rotation-/moving axis



roof - rotation-/moving axis



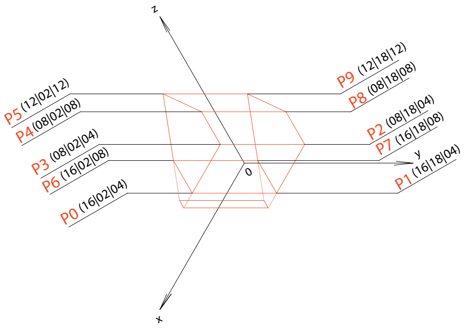
1. RULES | Moving:

- CoordinateEquation of Plane: $ax+by+cz+d=0$;
- by moving plane along axis, only 'd' changes;
- d' is calculated by insertion of point from absolute value of moving into coordinate equation :
- each plane has reference point to move:

e.g. movement along x-axis by 50 → $P = \begin{vmatrix} 50 \\ 0 \\ 0 \end{vmatrix}$

E 1 (P0,P1,P2,P3) : _ ;	E 2 (P0,P1,P7,P6) : P7;
E 3 (P2,P3,P4,P8): P4;	E 4 (P0,P3,P4,P5,P6): P6;
E 5 (P1,P2,P8,P9,P7): P8;	E 6 (P4,P5,P8,P9): P5;
E 7 (P6,P7,P5,P9): P9;	

2. EXAMPLE | Moving:



E 2: P7; PLANE P0,P1,P6,P7 → MOVE ALONG X-AXIS TO V=20;
CoordinateEquation E 2: $-64x + 0y + 0z + d = 0$; P7=(16,18,8)

A: Plane Point P7 After Movement:

(1) : $P = \begin{vmatrix} 20 \\ 18 \\ 8 \end{vmatrix}$

B: New Coordinate Equation :

- in $-64x + 0y + 0z + d = 0$;

$\Rightarrow -64 * (20) + 0* (18) + 0* (8) + d' = 0$

$\Rightarrow -1280 + d' = 0$

$\Rightarrow 1280 = d'$ in $-64x + 0y + 0z + d = 0$;

$\Rightarrow$ **$-64x + 0y + 0z + 1280 = 0$** : E 2

1. RULES | Rotating:

- Rotation Matrix: 3 cases

$$\text{x\_Axis}: \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos w^* & \sin w \\ 0 & -\sin w & \cos w \end{vmatrix};$$

$$\text{y\_Axis}: \begin{vmatrix} \cos w & 0 & \sin w \\ 0 & 1 & 0 \\ -\sin w & 0 & \cos w \end{vmatrix};$$

$$\text{z\_Axis}: \begin{vmatrix} \cos w & \sin w & 0 \\ -\sin w & \cos w & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

*w = Angle of Rotation

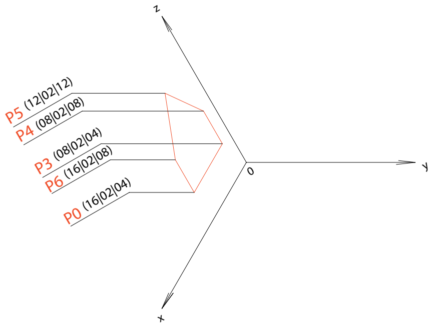
- Calculation of Plane Points: RotationMatrix * Vector:
  Vector is one of three Points from the Plane before Rotation.

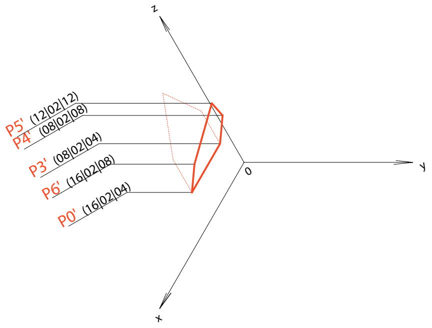
$$\begin{vmatrix} a11 & a12 & a13 \\ a21 & a22 & 23 \\ a31 & a32 & a33 \end{vmatrix} * \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix} = \begin{vmatrix} a11*x1+ & a12*x2+ & a13*x3 \\ a21*x1+ & a22*x2+ & a23*x3 \\ a31*x1+ & a32*x2+ & a33*x3 \end{vmatrix}$$

- Plane Points To Calculate after Manipulation:

E1:	- Plane not manipulabel -	
E2:	P0', P1', P7'	$\Rightarrow$ u2', v2', n2'
E3:	P3', P2', P4'	$\Rightarrow$ u3', v3', n3'
E4:	P0', P3', P6'	$\Rightarrow$ u4', v4', n4'
E5:	P2', P1', P8'	$\Rightarrow$ u5', v5', n5'
E6:	P4', P8', P5'	$\Rightarrow$ u6', v6', n6'
E7:	P7', P6', P9'	$\Rightarrow$ u7', v7', n7'

2. EXAMPLE | Rotating:





E 4:

PLANE Po,P3,P4,P5,P6 → ROTATION X - AXIS; w=+30°;

CoordinateEquation E 4: ox + 32y + oz + d =o;

Plane Points: Po=(16,2,4); P3=(8,2,4); P6=(16,2,8);

<u>A: Product Matrix and VectorPoint :</u>

$$(1){:}Po' = \begin{vmatrix}1&0&0\\0&\cos30&\sin30\\0&-\sin30&\cos30\end{vmatrix} * \begin{vmatrix}16\\2\\4\end{vmatrix} = \begin{vmatrix}16+&0+&0\\0+&\cos30*2+&\sin30*4\\0+&(-\sin30*2)+&\cos30*4\end{vmatrix} = \begin{vmatrix}16\\3.732\\2.464\end{vmatrix}$$

$$(2){:}P3' = \begin{vmatrix}1&0&0\\0&\cos30&\sin30\\0&-\sin30&\cos30\end{vmatrix} * \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}8+&0+&0\\0+&\cos30*2+&\sin30*4\\0+&(-\sin30*2)+&\cos30*4\end{vmatrix} = \begin{vmatrix}8\\3.732\\2.464\end{vmatrix}$$

$$(3){:}P6' = \begin{vmatrix}1&0&0\\0&\cos30&\sin30\\0&-\sin30&\cos30\end{vmatrix} * \begin{vmatrix}16\\2\\8\end{vmatrix} = \begin{vmatrix}16+&0+&0\\0+&\cos30*2+&\sin30*8\\0+&(-\sin30*2)+&\cos30*8\end{vmatrix} = \begin{vmatrix}16\\5.73\\5.928\end{vmatrix}$$

<u>B: VectorCalculations :</u>

u4' = P3' − Po'   v4' = P6' - Po'

$$u4' = \begin{vmatrix}8\\3.732\\2.464\end{vmatrix} - \begin{vmatrix}16\\3.732\\2.464\end{vmatrix} = \begin{vmatrix}-\mathbf{8}\\0\\0\end{vmatrix}; \quad v4' = \begin{vmatrix}16\\5.73\\5.928\end{vmatrix} - \begin{vmatrix}16\\3.732\\2.464\end{vmatrix} = \begin{vmatrix}0\\2\\3.464\end{vmatrix};$$

<u>C: Product of Vectors:</u>

$$n4'{:}\ u4' \otimes v4' = \begin{vmatrix}-\mathbf{8}\\0\\0\end{vmatrix} \otimes \begin{vmatrix}0\\2\\3.464\end{vmatrix} = \begin{vmatrix}0\\0\\-8*2\end{vmatrix} - \begin{vmatrix}0\\-8*3.464\\0\end{vmatrix} = \begin{vmatrix}0\\27{,}712\\-16\end{vmatrix};$$

<u>D: CoordinateEquation E4':</u>

$$n4' = \begin{vmatrix}0\\27{,}712\\-16\end{vmatrix}; \quad Po' = \begin{vmatrix}16\\3.732\\2.464\end{vmatrix}; \quad E\,4{:}\ ox + oy + oz + d = o;$$

n4’ in E4’: $\Rightarrow$ ax + by + cz + d = 0;

$\Rightarrow$ 0x + 27,712y -16z + d = 0;

Po’ in E4’: $\Rightarrow$ (0*16) + (27,712*3.732) - (16*2.464) + d = 0;

$\Rightarrow$ 103.421 – 39.424 + d = 0

$\Rightarrow$ -63.997 = d

$\Rightarrow$ -63.997 = d in: 0x + 27,712y -16z + d = 0;

$\Rightarrow$ **0x + 27.712y -16z –63.997 = 0** : $E4' = E4$

Sample: $\Rightarrow$ P6’: $\begin{vmatrix} 16 \\ 5.73 \\ 5.928 \end{vmatrix}$ in E4’ = 0

$\Rightarrow$ (0*16)+ (27.712*5.73) –(16*5.928) –63.997 = 0

$\Rightarrow$ (158.789) –(94.848) –63.997 = 0

$\Rightarrow$ 0 = 0

<u>E: Calculation of SectionPoint P6 :</u>

P6: SectionPoint of E2,E4,E7:

E2: -64x + 0y + 0 z = -1024;

E4: 0x + 27.712y - 16z = 63.997;

E7: -64x+0y – 64z = -1536;

$\Rightarrow$ *E1 : Determinant of Matrix:*

$$Det\ A4=\begin{vmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{vmatrix};$$

$$Det\ 4=\begin{vmatrix} -64 & 0 & 0 \\ 0 & 27.712 & -16 \\ -64 & 0 & -64 \end{vmatrix} = -64^*\begin{vmatrix} 27.712 & -16 \\ 0 & -64 \end{vmatrix} - 0^*\begin{vmatrix} 0 & 0 \\ 0 & -64 \end{vmatrix}$$

$$+(-64)^*\begin{vmatrix} 0 & 0 \\ 27.712 & -16 \end{vmatrix}$$

$$= -64^*(27.712^*(-64))$$

<u>Det A4 = 113508.352</u> ; Det A4 $\neq$ 0 $\Rightarrow$ SectionPoint;

E2 : Inverse of Matrix:

$$A^{-1} = 1/\ Det\ A4\ ^* \begin{Vmatrix} \begin{vmatrix} a22 & a23 \\ a32 & a33 \end{vmatrix} & \begin{vmatrix} a13 & a12 \\ a33 & a32 \end{vmatrix} & \begin{vmatrix} a12 & a13 \\ a22 & a23 \end{vmatrix} \\ \begin{vmatrix} a23 & a21 \\ a33 & a31 \end{vmatrix} & \begin{vmatrix} a11 & a13 \\ a31 & a33 \end{vmatrix} & \begin{vmatrix} a13 & a11 \\ a23 & a21 \end{vmatrix} \\ \begin{vmatrix} a21 & a22 \\ a31 & a32 \end{vmatrix} & \begin{vmatrix} a12 & a11 \\ a32 & a31 \end{vmatrix} & \begin{vmatrix} a12 & a11 \\ a32 & a31 \end{vmatrix} \end{Vmatrix}$$

$$A^{-1} = 1/\ Det\ A4\ ^* \begin{Vmatrix} \begin{vmatrix} 27.712 & -16 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & 0 \\ 27.712 & -16 \end{vmatrix} \\ \begin{vmatrix} -16.00 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ -64 & -64 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ -16 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 27.712 \\ -64 & 0 \end{vmatrix} & \begin{vmatrix} 0 & -64 \\ 0 & -64 \end{vmatrix} & \begin{vmatrix} -64 & 0 \\ 0 & 27.712 \end{vmatrix} \end{Vmatrix}$$

$$A^{-1} = 1/\ Det\ A4\ ^* \begin{vmatrix} (27.712*(-64)) & 0 & 0 \\ (-16*(-64)) & (-64)*(-64) & -(-16*(-64)) \\ -(-64*27.712) & 0 & (-64*27.712) \end{vmatrix}$$

$$A^{-1} = 1/\ 113508.352\ ^* \begin{vmatrix} -1773.568 & 0 & 0 \\ 1024 & 4096 & -1024 \\ 1773.568 & 0 & -1773.568 \end{vmatrix} * \begin{vmatrix} -1024 \\ 63.997 \\ -1536 \end{vmatrix}$$

$$A^{-1} = 1/\ 113508.352\ ^* \begin{vmatrix} 1816133.632+ & 0+ & 0 \\ -1048576+ & 262131.712+ & 1572864 \\ -1816133.632+ & 0+ & 2724200.448 \end{vmatrix}$$

$$\Rightarrow A^{-1} = 1/\ 113508.352\ ^* \begin{vmatrix} 1816133.632 \\ 786419.712 \\ 908066.816 \end{vmatrix} = \begin{vmatrix} 16 \\ 6.928 \\ 8 \end{vmatrix}$$

$\Rightarrow$ ***P6: (16, 6.928, 8);***

C4.7 Morphing Algorithm

1. RULES:

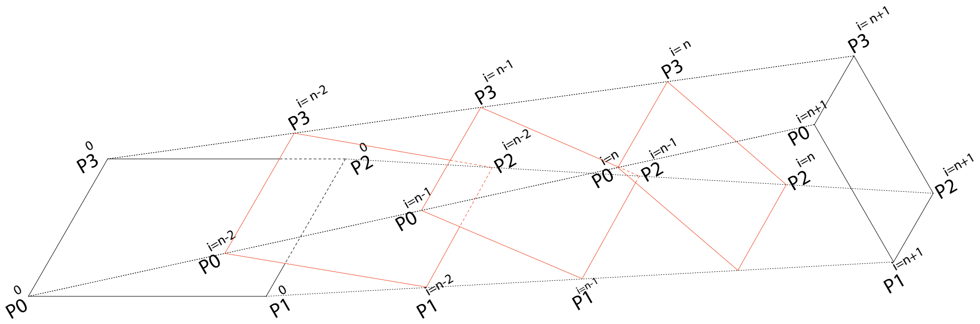
- Formula: $P_x^{\,i} = (P_x^{\,i=n+1} - P_x^{\,o} / n+1) * i + P_x^{\,o}$

x = Point Index;
n= Number of Morphs;
i = Morph Index (o <=i <=n);
e.g. ⇒ $P_x^{\,o}$ = StartPoint, i=o;
$P_x^{\,i=n+1}$ = GoalPoint; i= n+1;

2. EXAMPLE:

$\mathbf{E}_{1 \rightarrow 1'}$:

PLANE E1 (Start): $P_o^{\,o}$ (16,2,4); $P_1^{\,o}$ (16,18,4); $P_2^{\,o}$(8,18,4); $P_3^{\,o}$(8,2,4); PLANE E1' (Goal): $P_o^{\,i=n+1}$(14,18,12); $P_1^{\,i=n+1}$ (14,18,4); $P_2^{\,i=n+1}$ (10,18,4; $P_3^{\,i=n+1}$(10,18,12); → n= 2



i=1;

A: Calculation Morph i=1:

(1): $P_O{}^1 = (P_O{}^4 - P_O{}^o / 2+1) * 1 + P_O{}^o;$

$$P_O{}^1 = \left[\begin{vmatrix}14\\18\\12\end{vmatrix} - \begin{vmatrix}16\\2\\4\end{vmatrix} / 3\right] *1 + \begin{vmatrix}16\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\5.333\\2.666\end{vmatrix} *1 + \begin{vmatrix}16\\2\\4\end{vmatrix} = \begin{vmatrix}15.334\\7.333\\6.666\end{vmatrix};$$

(2) : $P_1{}^1 = (P_1{}^4 - P_1{}^o / 2+1) * 1 + P_1{}^o;$

$$P_1{}^1 = \left[\begin{vmatrix}14\\18\\4\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} / 3\right] *1 + \begin{vmatrix}16\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\0\\0\end{vmatrix} *1 + \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}15.334\\18\\4\end{vmatrix};$$

(3) : $P_2{}^1 = (P_2{}^4 - P_2{}^o / 2+1) * 1 + P_2{}^o;$

$$P_2{}^1 = \left[\begin{vmatrix}10\\18\\4\end{vmatrix} - \begin{vmatrix}8\\18\\4\end{vmatrix} / 3\right] *1 + \begin{vmatrix}8\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\0\\0\end{vmatrix} *1 + \begin{vmatrix}8\\18\\4\end{vmatrix} = \begin{vmatrix}8.666\\18\\4\end{vmatrix};$$

(4) : $P_3{}^1 = (P_3{}^4 - P_3{}^o / 2+1) * 1 + P_3{}^o;$

$$P_3{}^1 = \left[\begin{vmatrix}10\\18\\12\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} / 3\right] *1 + \begin{vmatrix}8\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\5.333\\2.666\end{vmatrix} *1 + \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}8.666\\7.333\\6.666\end{vmatrix};$$

i=2;

A: Calculation Morph i=2:

(1): $P_O{}^2 = (P_O{}^4 - P_O{}^o / 2+1) * 2 + P_O{}^o;$

$$P_O{}^2 = \left[\begin{vmatrix}14\\18\\12\end{vmatrix} - \begin{vmatrix}16\\2\\4\end{vmatrix} / 3\right] *2 + \begin{vmatrix}16\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\5.333\\2.666\end{vmatrix} *2 + \begin{vmatrix}16\\2\\4\end{vmatrix} = \begin{vmatrix}14.668\\12.666\\9.333\end{vmatrix};$$

(2): $P_1{}^2 = (P_1{}^4 - P_1{}^o / 2+1) * 2 + P_1{}^o;$

$$P_1{}^2 = \left[\begin{vmatrix}14\\18\\4\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} / 3\right] *2 + \begin{vmatrix}16\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\0\\0\end{vmatrix} *2 + \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}14.668\\18\\4\end{vmatrix};$$

(3): $P_2{}^2 = (P_2{}^4 - P_2{}^o / 2+1) * 2 + P_2{}^o;$

$$P_2{}^2 = \left[\begin{vmatrix}10\\18\\4\end{vmatrix} - \begin{vmatrix}8\\18\\4\end{vmatrix} / 3\right] *2 + \begin{vmatrix}8\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\0\\0\end{vmatrix} *2 + \begin{vmatrix}8\\18\\4\end{vmatrix} = \begin{vmatrix}9.333\\18\\4\end{vmatrix};$$

(4): $P_3{}^2 = (P_3{}^4 - P_3{}^o / 2+1) * 2 + P_3{}^o;$

$$P_3{}^2 = \left[\begin{vmatrix}10\\18\\12\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} / 3\right] *2 + \begin{vmatrix}8\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\5.333\\2.666\end{vmatrix} *2 + \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}9.333\\12.666\\9.333\end{vmatrix};$$

i=3;

A: Calculation Morph i=2:

(1): $P_O{}^3 = (P_O{}^4 - P_O{}^o / 2+1) * 3 + P_O{}^o;$

$$P_O{}^3 = \left[\begin{vmatrix}14\\18\\12\end{vmatrix} - \begin{vmatrix}16\\2\\4\end{vmatrix} / 3\right] *3 + \begin{vmatrix}16\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\5.333\\2.666\end{vmatrix} *3 + \begin{vmatrix}16\\2\\4\end{vmatrix} = \begin{vmatrix}14\\18\\12\end{vmatrix};$$

(2): $P_1{}^3 = (P_1{}^4 - P_1{}^o / 2+1) * 3 + P_1{}^o;$

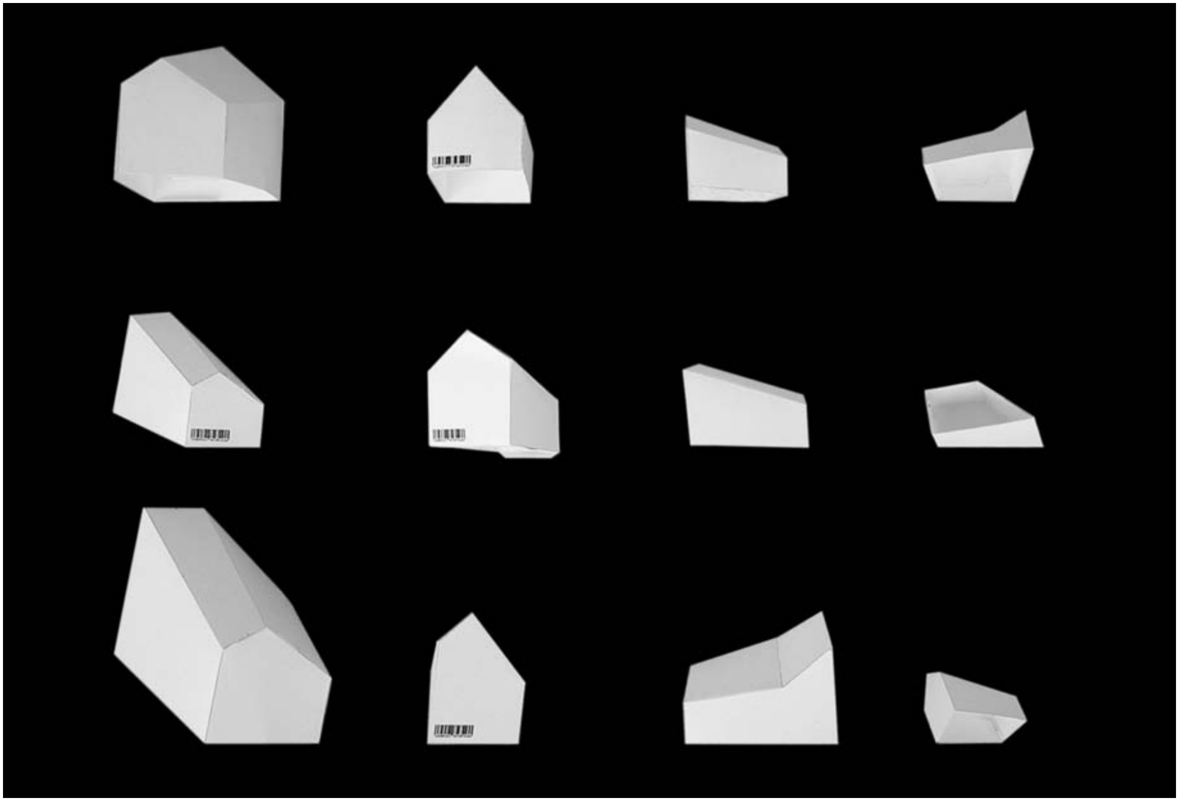
$$P_1{}^3 = \left[\begin{vmatrix}14\\18\\4\end{vmatrix} - \begin{vmatrix}16\\18\\4\end{vmatrix} / 3\right] *3 + \begin{vmatrix}16\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}-0.666\\0\\0\end{vmatrix} *3 + \begin{vmatrix}16\\18\\4\end{vmatrix} = \begin{vmatrix}14\\18\\4\end{vmatrix};$$

(3): $P_2{}^3 = (P_2{}^4 - P_2{}^o / 2+1) * 3 + P_2{}^o;$

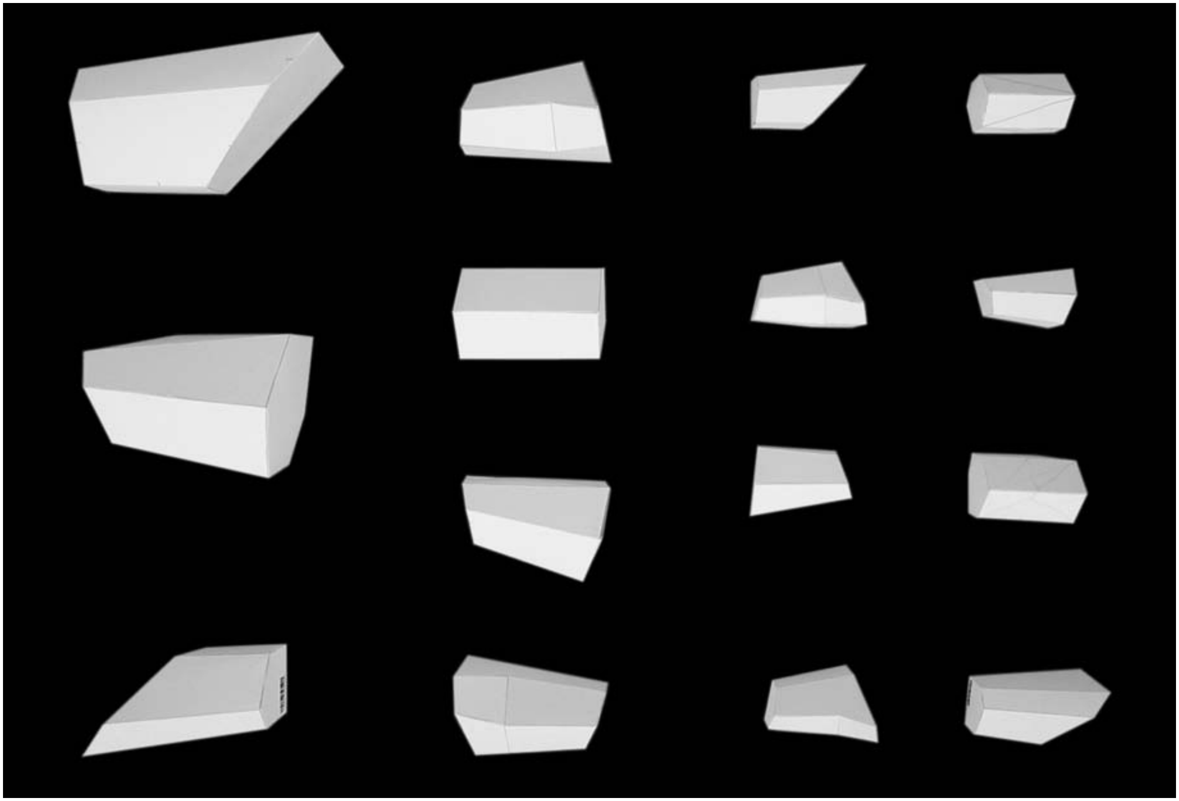
$$P_2{}^3 = \left[\begin{vmatrix}10\\18\\4\end{vmatrix} - \begin{vmatrix}8\\18\\4\end{vmatrix} / 3\right] *3 + \begin{vmatrix}8\\18\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\0\\0\end{vmatrix} *3 + \begin{vmatrix}8\\18\\4\end{vmatrix} = \begin{vmatrix}10\\18\\4\end{vmatrix};$$

(4): $P_3{}^3 = (P_3{}^4 - P_3{}^o / 2+1) * 3 + P_3{}^o;$

$$P_3{}^3 = \left[\begin{vmatrix}10\\18\\12\end{vmatrix} - \begin{vmatrix}8\\2\\4\end{vmatrix} / 3\right] *3 + \begin{vmatrix}8\\2\\4\end{vmatrix} \Rightarrow \begin{vmatrix}0.666\\5.333\\2.666\end{vmatrix} *3 + \begin{vmatrix}8\\2\\4\end{vmatrix} = \begin{vmatrix}10\\18\\12\end{vmatrix};$$

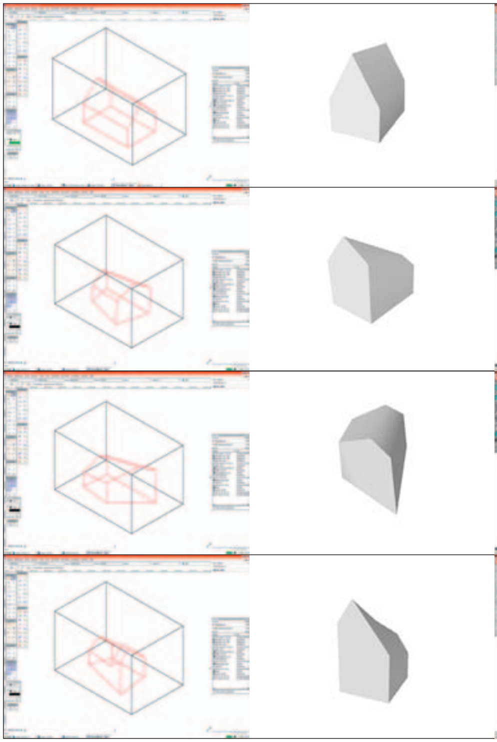


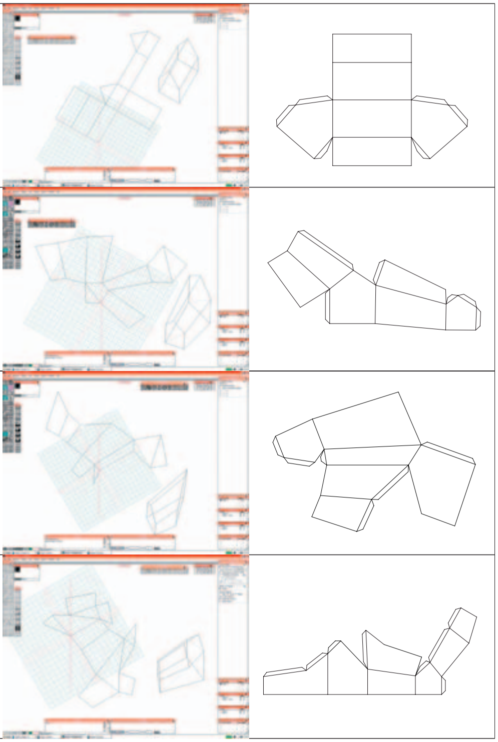
C5 First Prototypes



1) CAD-Software

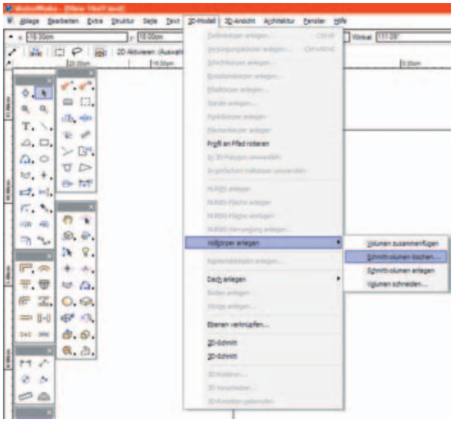
2) FormZ

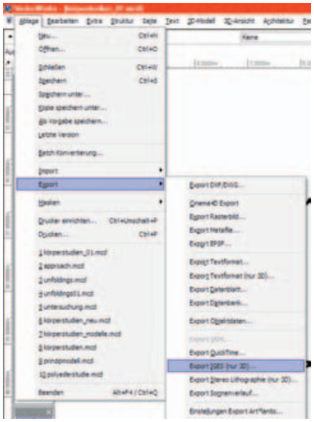




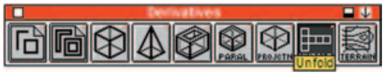
C7.1 Approach

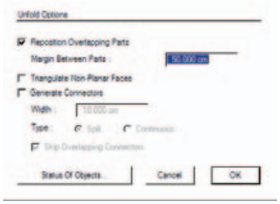
1) CAD - Software to shape out the model of a block, using boolean operation - functions for subtraction. Export of Model as STL (3D) or IGES (3D) format:





2) FormZ to generate unfolding of the model:





Unfold Options

[✓] Reposition Overlapping Parts

Margin Between Parts : 50.000 cm

[ ] Triangulate Non-Planar Faces

[ ] Generate Connectors

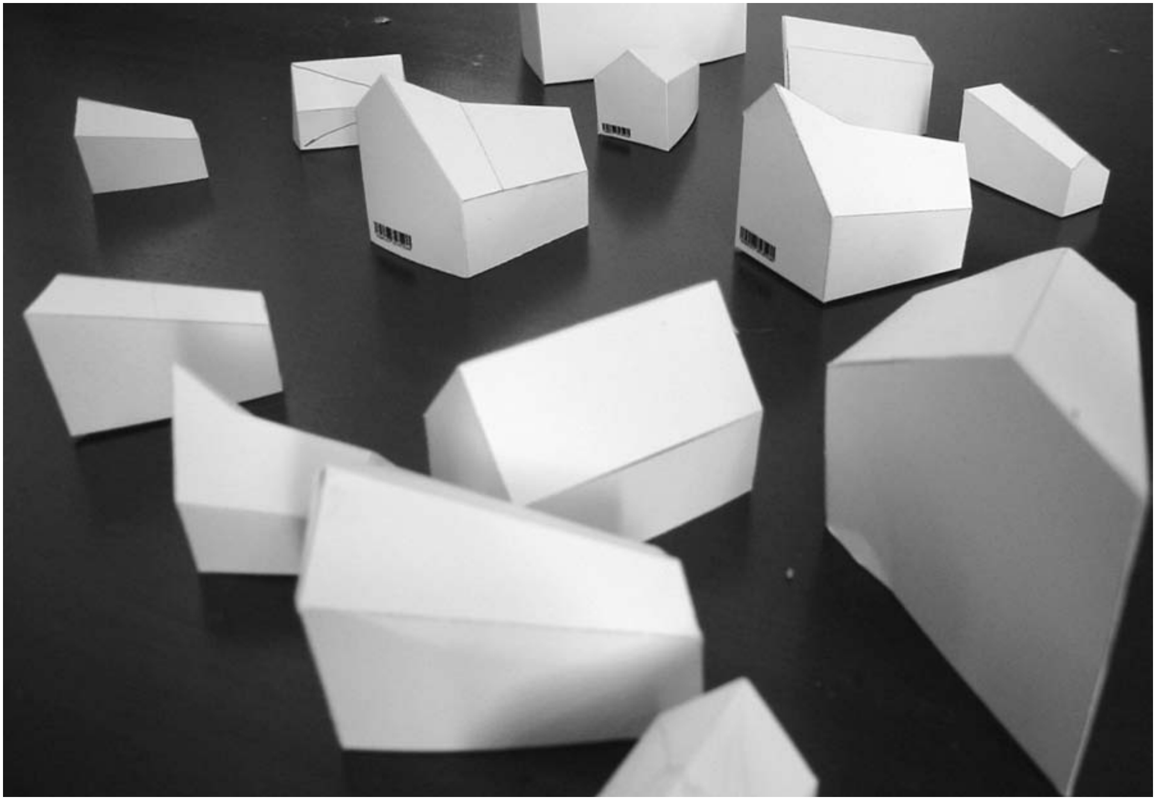
Width : 10.000 cm

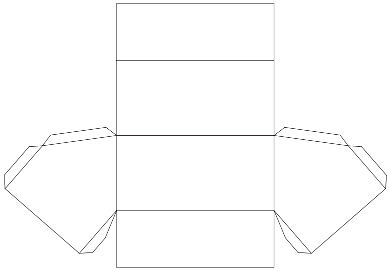
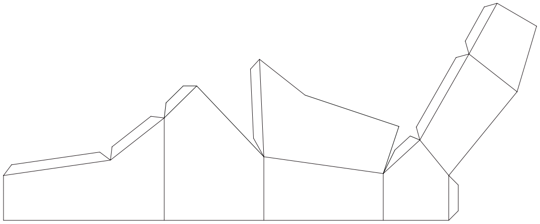
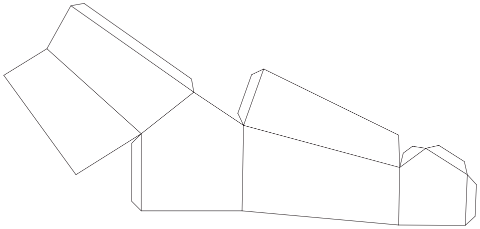
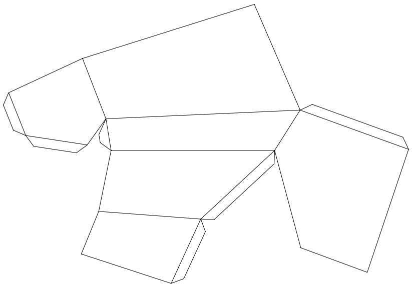
Type : [✓ Split] Continuous

[✓] Skip Overlapping Connectors

Status Of Objects... Cancel OK

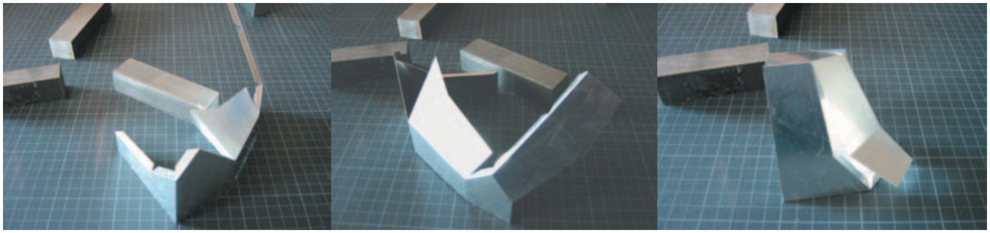
C7.2 Paper Models

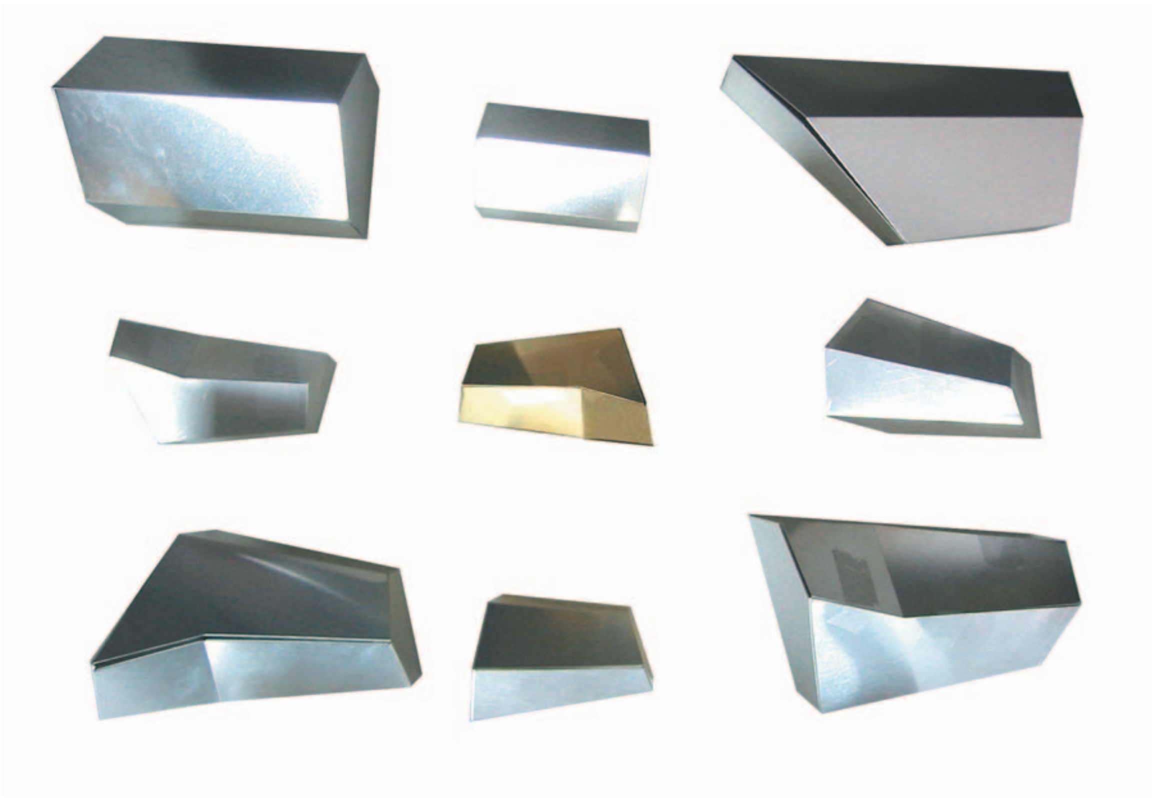


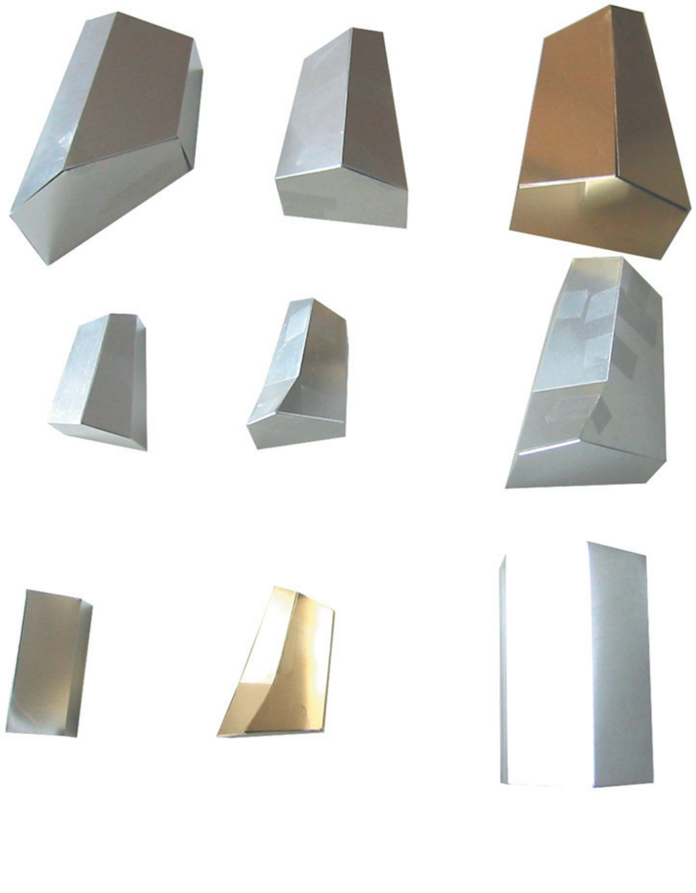





C7.3 Sheet Metal Models

Lasering the unfoldings out of metal sheet with the Bystronic - lasermachine Bystar 3015-2 at the Techno-park in Zurich. Beside construction steel and stainless steel, the machine is lasering wood and plexiglas, occasional plastics. The maximum size is 3000 x 1500 mm. It uses two different types of gas, oxygen for construction steel, and nitrogen for stainless steel.


Bending the unfoldings by hand with help of sharp-edged steel blocks.





Chapter D
PROGRAMMING

D1 Programming Code and Algorithms (JAVA)

```
/* HouseMachine.java */

import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.*;
import java.io.*;
import java.net.URL;
import java.net.MalformedURLException;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.behaviors.vp.*;
import javax.vecmath.*;
import java.util.*;
import java.lang.Object.*;
import com.sun.j3d.utils.geometry.GeometryInfo;
import javax.media.j3d.Shape3D;
import com.sun.j3d.utils.geometry.Text2D;
import java.text.*;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.geometry.Box;
import com.sun.j3d.utils.applet.MainFrame;
import java.applet.Applet;
import com.sun.j3d.utils.applet.MainFrame;
import javax.media.j3d.*;



public class HouseMachine
extends Frame
implements ActionListener, AdjustmentListener, ItemListener, GeometryUpdater
{

   private java.net.URL rootPath = null;

   public int sliderMode = 0; // 0=x, 1=y, 2=z
   public Scrollbar scb1,scb2,scb3;
   public Scrollbar rscb1, rscb2, rscb3;
   public Checkbox[] cbs = new Checkbox[6];
   public CheckboxGroup cbg;


                         public Texture tex;


   public Label jobIDlabel;

    public JPanel morphMorphsPanel;
    public JPanel morphsPanel = new JPanel();
```

```
   public JPanel viewPanel;

   public int sthMode = 0;
   public JPanel controlPanel;
   public JPanel newJobPanel;
   public JPanel newMorphsPanel;

                         public Button newJobButton;
                         public Button startHouseButton;
   public Button targetHouseButton;
   public Button morphButton;

   public Button sthSubmitButton;
   public Button sthParamButton;
                         public TextField sthHLtf, sthHWtf, sthEHtf, sthRHtf;
                         public Label sthHLl, sthHWl, sthEHl, sthRHl;

                         public String currentDate;

                         public TextField njPNrTf;
                         public TextField njNrMorphsTf;

  public TextField jobIDtf;
  public TextField morphNrTf;
  public JPanel emptyPanel;

  public TextField[] Tfx = new TextField[10];
  public    TextField[] Tfy = new TextField[10];
  public   TextField[] Tfz = new TextField[10];


   public int menuMode = 0;
   public int changePlane = 0;


   public THouse[] houseArray;
   public int houseArrayCount = 0;

   public THouse StartHouse;
   public THouse TargetHouse;

   public THouse modifyHouse;

   private boolean spin = false;
   private boolean noTriangulate = false;
   private boolean noStripify = false;
   private double creaseAngle = 60.0;

   public int morphsVisible = 0;

   public TProject project;
```

```
 public Transform3D basisRot = new Transform3D();




 public int changeMode = 0; // 0 = change Point
 public int changePoint = 0;
 public float changeX = 0f;
 public float changeY = 0f;
 public float changeZ = 0f;
 public int changeVertix = 0;

  public static final int[][] aPoint2Vertices = {

                    {  4,3, 2,2, 1,3, -1,-1 },  // P0 (f,v) (fläche/vertice)
                    {  2,0,             5,1, 1,1, -1,-1 },   // -1 = not used
                    {  3,2, 5,3, 1,0, -1,-1 },
                    {  4,1, 3,0, 1,2, -1,-1 },
                    {  4,0, 3,1, 6,0, -1,-1 },
                    {  7,3, 4,2, 6,1, 4,4 },   // P5
                    {  7,2, 4,5, 2,3, -1,-1 },
                    {  7,0, 2,1, 5,0, -1,-1 },
                    {  3,3, 5,5, 6,2, -1,-1 },
                    {  7,1, 5,2, 6,3, 5,4 }

  };


 public THouse createHouse(TProject project){

                     THouse house;
                     house = new THouse();
                     houseArray [houseArrayCount] = house;
   house.arrayNr = houseArrayCount;
                     houseArrayCount++;
                     house.project = project;

                     return (house);
 }


 public double convertObjC2HouseC(double value, double Qs, double Qe, double
Rs, double Re)
 {
                    return (( (value-Qs)*(Re-Rs)) / (Qe-Qs) + Rs);
 }



                    public class TPoint
                    {
                                        double x,y,z;
                    }

                    public class TVector
                    {
                                        double[] l = new double[4]; // 0..3  (1..3)
                    }

                    public class TLayer
                    {
                                        TVector n,u,v = new TVector();
                                        TVector a = new TVector(); // Stuetzvector
                      double ka,kb,kc,kd; // Koordinatengleichung der Ebene d
                    }

                    public class TMatrix3x3
                    {

                                        double[] e = new double[9];  //0..8
                    }


public class TProject
{

                     String date;
                     String projectNr;
                     String jobNr;
                     int morphSteps;

                     THouse startHouse;
                     THouse targetHouse;
}


                    public class THouse
                    {
                      TPoint[] P = new TPoint[10]; //0..9
                                        double HouseLength;
                                        double HouseWidth;
                                        double EavesHeight;
                                        double RidgeHeight;
                                        TLayer[] e = new TLayer[8];            //1..7


                                        TProject project;
                                        int morphIndex;


                                        double[][] mods = new double[8][6];

                                        int GUIwireFrame = 0;
 public double QsX = 0;
```

```
   public double QeX = 0;
   double RsX = 0;
   public double ReX = 0;

   public double QsY = 0;
   public double QeY = 0;
   public double RsY = 0;
   public double ReY = 0;

   public double QsZ = 0;
   public double QeZ = 0;
   public double RsZ = 0;
   public double ReZ = 0;

   public Hashtable hashFaces;
   public THousePoints startWerte;

   public int arrayNr = 0;
  private SimpleUniverse u;
   private BoundingSphere bounds;
   public TransformGroup objTrans = new TransformGroup();
   Label volumePanelLabel = new Label(„   V:“,Label.LEFT);
   public Button wireButton;

     TPoint[] rP = new TPoint[10]; // 0..9



                                    public THouse()
                                    {
                                                for (int i=0;i<10;i++) {
                                                            P[i] = new
TPoint();
                                                            rP[i] = new
TPoint();
                                                }
                                                for (int i=0;i<8;i++) {
                                                            e[i] = new
TLayer();

                                                }

                                    };



      public void getRsRe()
      {

        QsX = 10000;
        QsY = 10000;
        QsZ = 10000;
```

```
   QeX = 0;
   QeY = 0;
   QeZ = 0;

   for (int i=0;i<10;i++)
   {

                   if ((P[i].x) < QsX) { QsX = P[i].x; }
                   if ((P[i].x) > QeX) { QeX = P[i].x; }

                 if ((P[i].y) < QsY) { QsY = P[i].y; }
                   if ((P[i].y) > QeY) { QeY = P[i].y; }

                   if ((P[i].z) < QsZ) { QsZ = P[i].z; }
                   if ((P[i].z) > QeZ) { QeZ = P[i].z; }

                  }



  }


                                public void convertP2rP()
                                {


   for (int i=0;i<10;i++)
   {

                   rP[i].x = convertObjC2HouseC (P[i].x, QsX, QeX, RsX, ReX);
                   rP[i].y = convertObjC2HouseC (P[i].y, QsY, QeY, RsY, ReY);
                   rP[i].z = convertObjC2HouseC (P[i].z, QsZ, QeZ, RsZ, ReZ);

   }
                                }



  public void CreateKoordinatengleichungen()
  {




                                // Berechnung der Ebene anhand der Startpunkte
                                for (int i=1;i<8;i++) {

                    this.e[i].u = VectorDif(Point2Vector( this.P[LayerPoints[ i][ 2]] ),
Point2Vector(this.P[  LayerPoints[ i][ 1]  ]));

                    this.e[i].v = VectorDif(Point2Vector( this.P[LayerPoints[ i][ 3]] ),
Point2Vector(this.P[  LayerPoints[ i][ 1]  ]));
                    this.e[i].n = VectorCross(this.e[i].u, this.e[i].v);
```

```
                        this.e[i].a = Point2Vector( this.P[  LayerPoints[ i][ 1]  ]);
                        this.e[i] = Koordinatengleichung(  this.e[i]);
                                        }

  CalcAlleSchnittpunkte(this);

                         }
}


 public class THousePoints
 {
                        TPoint[] P = new TPoint[10]; //0..9
                        public THousePoints()
                        {
                                                        for (int i=0;i<10;i++) {
                                                                     P[i] = new TPoint();
                                                                     }
                        }

                        public void copyHouse2Points(THouse house)
                        {
        for (int i=0;i<10;i++){
                        P[i].x = house.P[i].x;
                        P[i].y = house.P[i].y;
                        P[i].z = house.P[i].z;
        }
                        }

                        public void copyPoints2House(THouse house)
                        {
        for (int i=0;i<10;i++){
                        house.P[i].x = P[i].x;
                        house.P[i].y = P[i].y;
                        house.P[i].z = P[i].z;
        }
                        }


 }

  public static final int[][] Point2Vertices = {

                        {  4,3, 2,2, 1,3, -1,-1 },  // P0 (f,v) (fläche/vertice)
                        {  2,0,                5,1, 1,1, -1,-1 },   // -1 = not used
                        {  3,2, 5,3, 1,0, -1,-1 },
                        {  4,1, 3,0, 1,2, -1,-1 },
                        {  4,0, 3,1, 6,0, -1,-1 },
                        {  7,3, 4,2, 6,1, 4,4 },
                        {  7,2, 4,5, 2,3, -1,-1 },
                        {  7,0, 2,1, 5,0, -1,-1 },
                        {  3,3, 5,5, 6,2, -1,-1 },
                        {  7,1, 5,2, 6,3, 5,4 }
  };
```

```
 private static final int XAxis = 0;
 private static final int YAxis = 1;
 private static final int ZAxis = 2;

// GUI
// Punkte, die man zum darstellen der Ebene braucht (alle eckpunkte der ebene)
  private static final int[][] PunkteForSurface = {

     {0, 0,0,0,0,0}, // ebene 0 ist nicht definiert
     {4, 0,1,2,3,  0},           // Ebene 1 besteht aus 4 Punkten
     {4, 0,1,7,6,  0},
     {4, 2,3,4,8,  0},
     {5, 3,0,6,5,4},
     {5, 1,2,8,9,7},
     {4, 8,4,5,9,  0},
     {4, 6,7,9,5,  0}
                               };

  // 3 ebenen geschnitten ergeben einen Schnittpunkt
  private static final int[][] LayerSchnitt = {
    {0, 1,2,4},
    {0, 1,2,5},
    {0, 1,3,5},
    {0, 1,3,4},
    {0, 3,4,6},
    {0, 4,6,7},
    {0, 2,4,7},
    {0, 2,5,7},
    {0, 3,5,6},
    {0, 5,6,7}

  };

 // Bei Welcher Ebene nehmen wir welchen Schnittpunkt zum Verschieben (2..7)
  private static final int[] MoveLayerMovePoint =  {
     0, //n.d.
     0, //n.d.


     7,
     4,
     6,
     8,
     5,
     9
   };


  // 3 Punkte pro Ebene definieren die Ebenenkoordinatengleichung
 private static final int[][] LayerPoints = {
     {0, 0,0,0},  // ebene 0 ist nicht definiert
     {0, 0,1,3},
     {0, 1,0,7},
     {0, 3,2,4},
    {0, 0,3,6},
    {0, 2,1,8},
    {0, 4,8,5},
    {0, 7,6,9}
  };



private static final String VERSION = „HouseMachine“;

private static final String[] MONTHS = {
  „Januar“,   „Februar“, „März“,    „April“,
  „Mai“,      „Juni“,   „Juli“,    „August“,
  „September“, „Oktober“, „November“, „Dezember“
};

public void PrintPoint(TPoint P)
{
			System.out.println(P.x+“/“+P.y+“/“+P.z);
}


public void PrintVector(TVector V)
{
			System.out.print(V.l[1]);
			System.out.print(„/“);
			System.out.print(V.l[2]);
			System.out.print(„/“);
			System.out.println(V.l[3]);
}

public void PrintKoord(TLayer L)
{

			//System.out.println(L.ka+“x+“+L.kb+“y+“+L.kc+“z+“+L.kd+“=0“);
}

public TVector VectorDif(TVector v1, TVector v2)
{
			TVector result = new TVector();

  result.l[1] = v1.l[1] - v2.l[1];
  result.l[2] = v1.l[2] - v2.l[2];
  result.l[3] = v1.l[3] - v2.l[3];

			return result;
 }

public TVector Point2Vector (TPoint P)
{
			TVector result = new TVector();

  result.l[1] = P.x;
  result.l[2] = P.y;
  result.l[3] = P.z;

			return result;
 }


public TPoint Vector2Point(TVector V)
{
			TPoint result = new TPoint();

  result.x = V.l[1];
  result.y = V.l[2];
  result.z = V.l[3];

  return result;
}


public TVector VectorCross(TVector u, TVector v)
{
			TVector result = new TVector();

  result.l[1] = (u.l[2] * v.l[3]) - (u.l[3] * v.l[2]);
  result.l[2] = (u.l[3] * v.l[1]) - (u.l[1] * v.l[3]);
  result.l[3] = (u.l[1] * v.l[2]) - (u.l[2] * v.l[1]);

  return result;
}


public TLayer Koordinatengleichung(TLayer E)
{
  TLayer result = new TLayer();

  result.ka = E.n.l[1];
  result.kb = E.n.l[2];
  result.kc = E.n.l[3];
  result.kd = - (result.ka * E.a.l[1])
        - (result.kb * E.a.l[2])
        - (result.kc * E.a.l[3]);


  return result;
}
```

```
{

  //System.out.println(String.valueOf(M.e[1]));
  //System.out.println(„j:"+Matrix2x2(M.e[1],M.e[2],M.e[4],M.e[5]));
  return (    M.e[0] * Matrix2x2(M.e[4],M.e[5],M.e[7],M.e[8])
          - M.e[3] * Matrix2x2(M.e[1],M.e[2],M.e[7],M.e[8])
          + M.e[6] * Matrix2x2(M.e[1],M.e[2],M.e[4],M.e[5])
         );
}


public TMatrix3x3 GenMatrix (TLayer E1, TLayer E2, TLayer E3)
{

                    TMatrix3x3 result = new TMatrix3x3();


   result.e[0] = E1.ka;
   result.e[1] = E1.kb;
   result.e[2] = E1.kc;

   result.e[3] = E2.ka;
   result.e[4] = E2.kb;
   result.e[5] = E2.kc;

   result.e[6] = E3.ka;
   result.e[7] = E3.kb;
   result.e[8] = E3.kc;

  return result;
}


public TVector MatrixMulVector(TMatrix3x3 M, TVector V)
{
                    TVector result = new TVector();

  result.l[1] = M.e[0]*V.l[1] + M.e[1]*V.l[2] + M.e[2]*V.l[3];
  result.l[2] = M.e[3]*V.l[1] + M.e[4]*V.l[2] + M.e[5]*V.l[3];
  result.l[3] = M.e[6]*V.l[1] + M.e[7]*V.l[2] + M.e[8]*V.l[3];

  return result;
}



public TVector Schnittpunktvektor(TLayer E1, TLayer E2, TLayer E3)
{
                    TVector result = new TVector();

  result.l[1] = -E1.kd;
  result.l[2] = -E2.kd;
  result.l[3] = -E3.kd;
  return result;
```

```
 }





public double VolumeTetraeder(TPoint P0, TPoint P1, TPoint P2, TPoint P3)
{

                    TVector a,b,c;
                    TMatrix3x3 M = new TMatrix3x3();


   a = VectorDif(Point2Vector(P0), Point2Vector(P1));
   b = VectorDif(Point2Vector(P2), Point2Vector(P1));
   c = VectorDif(Point2Vector(P3), Point2Vector(P1));

   M.e[0] = a.l[1];
   M.e[3] = a.l[2];
   M.e[6] = a.l[3];

   M.e[1] = b.l[1];
   M.e[4] = b.l[2];
   M.e[7] = b.l[3];

   M.e[2] = c.l[1];
   M.e[5] = c.l[2];
   M.e[8] = c.l[3];

   return (Math.abs(Determinate(M)/6));
 }



// Volumen des Hauses besteht aus Volumen von 9 Tetraedern
public double VolumeHouse(THouse House)
{

                    House.convertP2rP();

                    return (  VolumeTetraeder(House.rP[6],House.rP[0],House.rP[4],House.r
P[7])
        + VolumeTetraeder(House.rP[1],House.rP[0],House.rP[3],House.rP[4])
        + VolumeTetraeder(House.rP[1],House.rP[0],House.rP[7],House.rP[4])
        + VolumeTetraeder(House.rP[2],House.rP[8],House.rP[7],House.rP[4])
        + VolumeTetraeder(House.rP[2],House.rP[1],House.rP[7],House.rP[4])
        + VolumeTetraeder(House.rP[2],House.rP[3],House.rP[1],House.rP[4])
        + VolumeTetraeder(House.rP[9],House.rP[8],House.rP[7],House.rP[5])
        + VolumeTetraeder(House.rP[4],House.rP[5],House.rP[6],House.rP[8])
        + VolumeTetraeder(House.rP[8],House.rP[6],House.rP[7],House.rP[5])
        );
}

 public double VectorLength(TVector v)
```

```
{
  return (Math.sqrt( v.l[1]*v.l[1] + v.l[2]*v.l[2] + v.l[3]*v.l[3]));
 }


 public double SurfaceTriangle(TPoint P0, TPoint P1, TPoint P2)
 {

                    TVector n,u,v;

  u = VectorDif(Point2Vector(P0), Point2Vector(P1));
  v = VectorDif(Point2Vector(P2), Point2Vector(P1));
  n = VectorCross(u,v);

  return (VectorLength(n)/2);
 }

 public double SurfaceRectangle(TPoint P0, TPoint P1, TPoint P2, TPoint P3)
 {

  return ( SurfaceTriangle(P0,P1,P2) +
       SurfaceTriangle(P0,P3,P2));
 }




 public double SurfaceHouse(THouse House)
 {

                    House.convertP2rP();

 return (   SurfaceRectangle(House.rP[1],House.rP[0],House.rP[6],House.rP[7])
       + SurfaceRectangle(House.rP[2],House.rP[3],House.rP[4],House.rP[8])
       + SurfaceRectangle(House.rP[8],House.rP[4],House.rP[5],House.rP[9])
       + SurfaceRectangle(House.rP[7],House.rP[6],House.rP[5],House.rP[9])
       + SurfaceRectangle(House.rP[3],House.rP[0],House.rP[6],House.rP[4])
       + SurfaceRectangle(House.rP[2],House.rP[1],House.rP[7],House.rP[8])
       + SurfaceTriangle (House.rP[4],House.rP[6],House.rP[5])
       + SurfaceTriangle (House.rP[8],House.rP[7],House.rP[9]) );
  }

public TPoint calcLayerSchnittpunkt(TLayer E1, TLayer E2, TLayer E3)
{

 TMatrix3x3 M;
 TVector SEV;
 double Det;
 TMatrix3x3 IM = new TMatrix3x3();
 TVector IV;
 int i;

  /*
  PrintKoord(E1);
```

```
   PrintKoord(E2);
   PrintKoord(E3);
   */

   M = GenMatrix (E1,E2,E3);
   SEV = Schnittpunktvektor(E1,E2,E3);


// System.out.println(„Matrix“);

   for (i=0;i<=8;i++)
   {


 //System.out.println(i+“: „+M.e[i]);
}

   Det = Determinate (M);

// System.out.println(„det:“+Det);

   //if Det=0 then
   // writeln(‚kein schnittpunkt ebenen parallel‘,#7); // Sonderfall!!! Ebenen sind falsch

   IM.e[0] = Matrix2x2(M.e[4],M.e[5],M.e[7],M.e[8]);
   IM.e[1] = Matrix2x2(M.e[2],M.e[1],M.e[8],M.e[7]);
   IM.e[2] = Matrix2x2(M.e[1],M.e[2],M.e[4],M.e[5]);
   IM.e[3] = Matrix2x2(M.e[5],M.e[3],M.e[8],M.e[6]);
   IM.e[4] = Matrix2x2(M.e[0],M.e[2],M.e[6],M.e[8]);
   IM.e[5] = Matrix2x2(M.e[2],M.e[0],M.e[5],M.e[3]);
   IM.e[6] = Matrix2x2(M.e[3],M.e[4],M.e[6],M.e[7]);
   IM.e[7] = Matrix2x2(M.e[1],M.e[0],M.e[7],M.e[6]);
   IM.e[8] = Matrix2x2(M.e[0],M.e[1],M.e[3],M.e[4]);


// System.out.println(„IMatrix“);

   for (i=0;i<=8;i++)
   {

//System.out.println(i+“; „+M.e[i]);
}

   IV = MatrixMulVector(IM,SEV);

   for (i=1;i<=3;i++)
   {

   //System.out.println(i+‘*‘);
   IV.l[i] = IV.l[i]/Det; }

   return (Vector2Point(IV));

}
```

```
// Berechne alle Schnittpunkte neu, anhand der Ebenengleichungen
public void CalcAlleSchnittpunkte(THouse House)
{

   for (int i=0;i<10;i++)
   {
     //System.out.print(„E3:"+LayerSchnitt[i][3]);
     House.P[i] = calcLayerSchnittpunkt(House.e[LayerSchnitt[i][1]], House.e[LayerSchnitt
[i][2]], House.e[LayerSchnitt[i][3]]);


     //System.out.print(„Schnitt:");
     //PrintPoint(House.P[i]);
   }


 NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
 nf.setMaximumFractionDigits(3);
 nf.setMinimumFractionDigits(3);
 nf.setMaximumIntegerDigits(4);
 nf.setMinimumIntegerDigits(1);

 double volHouse = VolumeHouse(House);
 double surHouse = SurfaceHouse(House);
 double areHouse = volHouse/surHouse;

 House.volumePanelLabel.setText(„   V: „+nf.format(volHouse)+"    A:
"+nf.format(surHouse) + „   V/A: „+nf.format(areHouse));

 // System.out.println(„volumen:"+volHouse);

 }

// LayerNr von 1 bis 6
public void MoveLayer(THouse House, int LayerNr, double Value, int Achse)
{

 TPoint SP;


 House.mods[LayerNr][Achse] = Value;

 SP = House.P[MoveLayerMovePoint[LayerNr]];

 if (Achse==XAxis) // Verschiebung um x-Aches
   { SP.x = Value; }
 else
 if (Achse==YAxis) // Veschiebung um y-Achse
   { SP.y = Value; }
 else  // Verschiebumg um z-Aches
   { SP.z = Value; }

 House.e[LayerNr].kd =  -(SP.x * House.e[LayerNr].ka)
                    -(SP.y * House.e[LayerNr].kb)
                    -(SP.z * House.e[LayerNr].kc);
 CalcAlleSchnittpunkte(House);
}


public double getLayerPos(THouse House, int LayerNr, int Achse)
{
 TPoint SP;
 double result;

 SP = House.P[MoveLayerMovePoint[LayerNr]];

 if (Achse==XAxis) // Verschiebung um x-Aches
   {
     result =( -(SP.y* House.e[LayerNr].kb)
                                    -(SP.z* House.e[LayerNr].kc)
                                    -House.e[LayerNr].kd) / House.e[LayerNr].ka;

     if (House.e[LayerNr].ka == 0) {

                    result = 0;
     }
   }
 else
 if (Achse==YAxis) // Veschiebung um y-Achse
   {
     result =( -(SP.x* House.e[LayerNr].ka)
                                    -(SP.z* House.e[LayerNr].kc)
                                    -House.e[LayerNr].kd) / House.e[LayerNr].kb;

     if (House.e[LayerNr].kb == 0) {

                    result = 0;
     }
   }

 else  // Verschiebumg um z-Aches
     {
     result =( -(SP.y* House.e[LayerNr].kb)
                                    -(SP.x* House.e[LayerNr].ka)
                                    -House.e[LayerNr].kd) / House.e[LayerNr].kc;
       if (House.e[LayerNr].kc == 0) {

                    result = 0;
     }

   }

 return (result);
}



public double Grad2BogenMass(double Winkel)
```

```
{
                          return ((Winkel*Math.PI)/180);
}



public void RotateLayer(THouse House, int LayerNr, double Winkel, int Achse, double
urWinkel)
{


                          TPoint[] P = new TPoint[3];
   TMatrix3x3 M = new TMatrix3x3();
   TVector V;
   int j,i;


   House.mods[LayerNr][Achse+3] = urWinkel;

  for (i=0;i<3;i++)
  {
    P[i] = (House.P[ LayerPoints[ LayerNr] [ i+1] ]);
  }

 if (Achse==XAxis) {

   M.e[0] = 1;
   M.e[1] = 0;
   M.e[2] = 0;

   M.e[3] = 0;
   M.e[4] = Math.cos(Grad2BogenMass(Winkel));
   M.e[5] = Math.sin(Grad2BogenMass(Winkel));

   M.e[6] = 0;
   M.e[7] = -Math.sin(Grad2BogenMass(Winkel));
   M.e[8] = Math.cos(Grad2BogenMass(Winkel));
 }

 else
 if (Achse==YAxis) {

   M.e[0] = Math.cos(Grad2BogenMass(Winkel));
   M.e[1] = 0;
   M.e[2] = Math.sin(Grad2BogenMass(Winkel));

   M.e[3] = 0;
   M.e[4] = 1;
   M.e[5] = 0;

   M.e[6] = -Math.sin(Grad2BogenMass(Winkel));;
   M.e[7] = 0;
   M.e[8] = Math.cos(Grad2BogenMass(Winkel));
 }
```

```
 else  // Achse=ZAxis
 {
                        M.e[0] = Math.cos(Grad2BogenMass(Winkel));
  M.e[1] = Math.sin(Grad2BogenMass(Winkel));
  M.e[2] = 0;

  M.e[3] = -Math.sin(Grad2BogenMass(Winkel));
  M.e[4] = Math.cos(Grad2BogenMass(Winkel));
  M.e[5] = 0;

  M.e[6] = 0;
  M.e[7] = 0;
  M.e[8] = 1;
 }


 for (i=0;i<3;i++)
 {
  P[i] = Vector2Point (MatrixMulVector(M, Point2Vector(P[i])));
 }

 House.e[LayerNr].u = VectorDif(Point2Vector(P[1]), Point2Vector(P[0]));
 House.e[LayerNr].v = VectorDif(Point2Vector(P[2]), Point2Vector(P[0]));
 House.e[LayerNr].n = VectorCross(House.e[LayerNr].u,House.e[LayerNr].v);
 House.e[LayerNr].a = Point2Vector(P[0]);

 House.e[LayerNr] = Koordinatengleichung(  House.e[LayerNr]);
 CalcAlleSchnittpunkte(House);

}






  public Shape3D getFaceShape (int face, Hashtable hashFaces)
  {
    Shape3D myShape;
    myShape = (Shape3D)hashFaces.get(„f"+face);
    return myShape;

  }

public void setActiveFace(int face, Hashtable hashFaces) {


                        Appearance look = new Appearance();
                        Color3f objColor = new Color3f(1.0f, 0.2f, 0.2f);
                        Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
                        Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
                        look.setMaterial(new Material(objColor, black,
                                                                                    objColor, white,
100.0f));


    getFaceShape(face, hashFaces).setAppearance(look);
 }

 public void setSolidFace(int face, Hashtable hashFaces) {


                    Appearance look = new Appearance();
                    Color3f objColor = new Color3f(1.0f, 1.0f, 1.0f);
                    Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
                    Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
                    look.setMaterial(new Material(objColor, black,
                                                                  objColor,
white, 100.0f));


    getFaceShape(face, hashFaces).setAppearance(look);
 }


public void setWireFrame(int face, Hashtable hashFaces)
{

Appearance app= new Appearance();

// Set up the polygon attributes (um WireFrame zu zeigen)
PolygonAttributes pa = new PolygonAttributes();
pa.setPolygonMode(pa.POLYGON_LINE);
pa.setCullFace(pa.CULL_NONE);
app.setPolygonAttributes(pa);

 getFaceShape(face, hashFaces).setAppearance(app);
}

public void resetHouseFace(THouse house){


   for (int i=1;i<8;i++)
                                      {
                                                      if (house.GUIwireFrame == 1) {
setWireFrame(i,house.hashFaces); }
                                                      else { setSolidFace(i,
house.hashFaces); }
                                      };

   }

 public void changePoint(int P, double x, double y, double z, THouse house)
   {
                      int i;
                      int f,v;
                    Hashtable hashFaces;
                    hashFaces = house.hashFaces;

                    changeMode = 0; // punkt ändern
    changeX = (float)x;
    changeY = (float)y;
    changeZ = (float)z;

                    for (i=0; i<4;i++)
                    {
                                      f = Point2Vertices[P][i*2];


                                      if (f != -1)
                                      {

                      changeVertix = Point2Vertices[P][i*2+1];
       ((GeometryArray)(getFaceShape(f, hashFaces ).getGeometry())).updateD
ata(this);

                                      }

                    }

                    house.P[P].x = x;
                    house.P[P].y = y;
                    house.P[P].z = z;


                    GUIupdateSectionPoint(P,house);

  };



 public void getPointFromModel(int P, TPoint myP, Hashtable hashFaces)
 {
                    int f,v;
    float[] vi;
    GeometryArray geo1;

    f = Point2Vertices[P][0];
    v = Point2Vertices[P][1];


    geo1 = (GeometryArray)(getFaceShape(f, hashFaces).getGeometry());

    vi = geo1.getInterleavedVertices();

    myP.x = vi[v*6+3+0];
                    myP.y = vi[v*6+3+1];
                    myP.z = vi[v*6+3+2];
 }

public void updateData(Geometry geometry) {


  GeometryArray geo1;
  float[] vi;
  int vertix;

 if (changeMode == 0)  // punkt ändern
 {

  geo1 = (GeometryArray)(geometry);
  vi = geo1.getInterleavedVertices();

  vi[changeVertix*6+3+0] = changeX;
  vi[changeVertix*6+3+1] = changeY;
  vi[changeVertix*6+3+2] = changeZ;

  geo1.setInterleavedVertices(vi);
 }





}



  public BranchGroup createSceneGraph(THouse StartHouse) {

  GeometryArray geo1;

                        // Create the root of the branch graph
                        BranchGroup objRoot = new BranchGroup();

     // Create a Transformgroup to scale all objects so they
     // appear in the scene.
     TransformGroup objScale = new TransformGroup();
     Transform3D t3d = new Transform3D();
     t3d.setScale(0.5);
     objScale.setTransform(t3d);
     objRoot.addChild(objScale);

                        // Create the transform group node and initialize it to the
                        // identity.  Enable the TRANSFORM_WRITE capability so that
                        // our behavior code can modify it at runtime.  Add it to the
                        // root of the subgraph.

                        StartHouse.objTrans.setCapability(TransformGroup.ALLOW_
TRANSFORM_WRITE);
                        StartHouse.objTrans.setCapability(TransformGroup.ALLOW_TRANS-
TRANSFORM_READ);
                        objScale.addChild(StartHouse.objTrans);



                        int flags = ObjectFile.RESIZE;
                        if (!noTriangulate) flags |= ObjectFile.TRIANGULATE;
                        if (!noStripify) flags |= ObjectFile.STRIPIFY;
                        ObjectFile f = new ObjectFile(flags,
                          (float)(creaseAngle * Math.PI / 180.0));
                        Scene s = null;
                        try {
                          s = f.load(„default.dat“);
                        }
                        catch (FileNotFoundException e) {
                          System.err.println(e);
                          System.exit(1);
                        }
                        catch (ParsingErrorException e) {
                          System.err.println(e);
                          System.exit(1);
                        }
                        catch (IncorrectFormatException e) {
                          System.err.println(e);
                          System.exit(1);
                        }



                        StartHouse.RsX = 0;
                        StartHouse.RsY = 0;
                        StartHouse.RsZ = 0;

                        StartHouse.ReX = 8;
                        StartHouse.ReY = 16;
                        StartHouse.ReZ = 8;




Appearance look = new Appearance();
                        Color3f objColor = new Color3f(1.0f, 0.2f, 0.2f);
                        Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
                        Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
                        look.setMaterial(new Material(objColor, black,
                                                                                  objColor,
white, 100.0f));

                        StartHouse.objTrans.addChild(s.getSceneGroup());

   StartHouse.hashFaces = s.getNamedObjects();
  StartHouse.HouseWidth = 8.0;
  StartHouse.HouseLength = 16.0;
  StartHouse.EavesHeight = 4.0;
  StartHouse.RidgeHeight = 8.0;


  for (int i=0; i<10;i++)
  {
                      getPointFromModel(i,StartHouse.P[i], StartHouse.hashFaces);

  }



 StartHouse.getRsRe();



 // Berechnung der Ebene anhand der Startpunkte
 for (int i=1;i<8;i++) {
   StartHouse.e[i].u = VectorDif(Point2Vector( StartHouse.P[LayerPoints[ i][ 2]] ),
Point2Vector(StartHouse.P[  LayerPoints[ i][ 1]  ]));

   StartHouse.e[i].v = VectorDif(Point2Vector( StartHouse.P[LayerPoints[ i][ 3]] ),
Point2Vector(StartHouse.P[  LayerPoints[ i][ 1]  ]));
   StartHouse.e[i].n = VectorCross(StartHouse.e[i].u, StartHouse.e[i].v);
   StartHouse.e[i].a = Point2Vector( StartHouse.P[  LayerPoints[ i][ 1]  ]);
   StartHouse.e[i] = Koordinatengleichung(  StartHouse.e[i]);



 }

   CalcAlleSchnittpunkte(StartHouse);


for (int i=1;i<8;i++)
 {


                      StartHouse.mods[i][XAxis] = getLayerPos(StartHouse, i, XAxis);
                      StartHouse.mods[i][YAxis] = getLayerPos(StartHouse, i, YAxis);
                      StartHouse.mods[i][ZAxis] = getLayerPos(StartHouse, i, ZAxis);
                     }


   StartHouse.startWerte = new THousePoints();
   StartHouse.startWerte.copyHouse2Points(StartHouse);




 Shape3D myShape;
 for (int i=1;i<8;i++)
 {



   myShape = (Shape3D)StartHouse.hashFaces.get(„f"+i);
   myShape.setCapability(Shape3D.ALLOW_GEOMETRY_READ);
   myShape.setCapability(Shape3D.ALLOW_GEOMETRY_WRITE);
   myShape.setCapability(Shape3D.ALLOW_APPEARANCE_WRITE);


   geo1 = (GeometryArray)myShape.getGeometry(0);
   geo1.setCapability(GeometryArray.ALLOW_COORDINATE_READ);
   geo1.setCapability(GeometryArray.ALLOW_COUNT_READ);
   geo1.setCapability(GeometryArray.ALLOW_REF_DATA_READ);
   geo1.setCapability(GeometryArray.ALLOW_REF_DATA_WRITE);
   geo1.setCapability(GeometryArray.ALLOW_COORDINATE_WRITE);
   geo1.setCapability(GeometryArray.ALLOW_FORMAT_READ);
   geo1.setCapability(GeometryArray.ALLOW_COLOR_WRITE);
   geo1.setCapability(GeometryArray.ALLOW_FORMAT_READ);

   setSolidFace(i,StartHouse.hashFaces);

 }




                      StartHouse.bounds = new BoundingSphere(new
Point3d(0.0,0.0,0.0), 9000.0);

      if (spin) {
                      Transform3D yAxis = new Transform3D();
                      Alpha rotationAlpha = new Alpha(-1, Alpha.INCREASING_ENAB-
LE,

 0, 0,

 4000, 0, 0,

 0, 0, 0);

                      RotationInterpolator rotator =
                         new RotationInterpolator(rotationAlpha, StartHouse.objTrans,
yAxis,
                                                                                          0.0f, (float)
Math.PI*2.0f);
                      rotator.setSchedulingBounds(StartHouse.bounds);
                      StartHouse.objTrans.addChild(rotator);
                     }

      // Set up the background
      Color3f bgColor = new Color3f(0.0f, 0.0f, 0.f);
      Background bgNode = new Background(bgColor);
```

```
    bgNode.setApplicationBounds(StartHouse.bounds);
    objRoot.addChild(bgNode);


  StartHouse.objTrans.getTransform(basisRot);


                        return objRoot;
 }

public void createMorphs(){


                        String name;
                        int morphSteps;
                        double value;

 THouse[] morphs = new THouse[100];

 morphsPanel.setLayout(new GridLayout(2,1));

 houseArray = new THouse[100];
 name = „“;

                        morphSteps = project.morphSteps;



 TPoint myP,startP,targetP;

 myP = new TPoint();
 startP = new TPoint();
 targetP = new TPoint();



 THouse targetMorph;
 THouse startMorph;


 startMorph = StartHouse;
 targetMorph = TargetHouse;

 houseArrayCount=0;

     morphs[0] = createHouse(project);
                              morphs[0].morphIndex = 0;

     morphsPanel.add(createShape(morphs[0]));



NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
nf.setMaximumFractionDigits(0);
 nf.setMinimumFractionDigits(0);
 nf.setMaximumIntegerDigits(2);
 nf.setMinimumIntegerDigits(2);



 loadHouse(morphs[0],project.date+project.projectNr+project.jobNr+“00“);
//nf.format(„0“));
       morphs[0].CreateKoordinatengleichungen();
                              updateHouse(morphs[0]);


       for (int j=0; j<10; j++)
       {
         getPointFromModel(j, startP, startMorph.hashFaces);
         morphs[0].P[j].x = startP.x;
         morphs[0].P[j].y = startP.y;
         morphs[0].P[j].z = startP.z;

       }

    for (int i=1;i<morphSteps+1;i++)
    {


       morphs[i] = createHouse(project);

       morphs[i].morphIndex = i;
       morphsPanel.add(createShape(morphs[i]));


       for (int j=0; j<10; j++)
       {
         getPointFromModel(j, startP, startMorph.hashFaces);
         morphs[i].P[j].x = startP.x;
         morphs[i].P[j].y = startP.y;
         morphs[i].P[j].z = startP.z;

       }




       for (int k=2;k<8;k++)
       {

         value = ( (targetMorph.mods[k][XAxis] - startMorph.mods[k][XAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][XAxis];

         MoveLayer(morphs[i],k,value,XAxis);


         value = ( (targetMorph.mods[k][YAxis] - startMorph.mods[k][YAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][YAxis];
```

```
          MoveLayer(morphs[i],k,value,YAxis);



          value = ( (targetMorph.mods[k][ZAxis] - startMorph.mods[k][ZAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][ZAxis];

          MoveLayer(morphs[i],k,value,ZAxis);






          value = ( (targetMorph.mods[k][3+XAxis] - startMorph.mods[k][3+XAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][3+XAxis];
          RotateLayer(morphs[i],k,value-morphs[i].mods[k][3+XAxis],XAxis, value);


          value = ( (targetMorph.mods[k][3+YAxis] - startMorph.mods[k][3+YAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][3+YAxis];
          RotateLayer(morphs[i],k,value-morphs[i].mods[k][3+YAxis],YAxis, value);



          value = ( (targetMorph.mods[k][3+ZAxis] - startMorph.mods[k][3+ZAxis]) /
(morphSteps+1) )*i + startMorph.mods[k][3+ZAxis];
          RotateLayer(morphs[i],k,value-morphs[i].mods[k][3+ZAxis],ZAxis, value);

                          }


                         for (int j=0; j<10;j++){
    changePoint(j, morphs[i].P[j].x,morphs[i].P[j].y,morphs[i].P[j].z, morphs[i]);
                         }


                         saveHouse(morphs[i]);

    };



        morphs[0+morphSteps+1] = createHouse(project);

        morphs[0+morphSteps+1].morphIndex = 0+morphSteps+1;
        morphsPanel.add(createShape(morphs[0+morphSteps+1]));



                                              loadHouse(morphs[0+morphSteps+1],project.d
ate+project.projectNr+project.jobNr+nf.format(morphSteps+1));
        morphs[0+morphSteps+1].CreateKoordinatengleichungen();
                              updateHouse(morphs[0+morphSteps+1]);
```

```
 }

  public JPanel createShape(THouse house)
   {


                         JPanel control3Dpanel = new JPanel();

                                        house.wireButton = new Button(„Wireframe“);
                                        house.wireButton.addActionListener(this);
                                        house.wireButton.setActionCommand(„WFH“+
house.arrayNr);
                         control3Dpanel.add(house.wireButton);

                                        control3Dpanel.setBackground(new Co-
lor(100,100,100));

 NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
 nf.setMaximumFractionDigits(0);
 nf.setMinimumFractionDigits(0);
 nf.setMaximumIntegerDigits(2);
 nf.setMinimumIntegerDigits(2);




                                        JPanel control3DpanelInfo = new JPanel();
                                        control3DpanelInfo.setLayout(new BorderLay-
out());
                                        Label control3DpanelInfoLabel1 = new
Label(house.project.date+““+house.project.projectNr+““+house.project.jobNr+““
+nf.format(house.morphIndex)+“   „);
    control3DpanelInfoLabel1.setForeground(new Color(255,255,255));
                                        control3DpanelInfo.add(control3DpanelInfoLab
el1, BorderLayout.EAST);

                                        Label viewLabel = new Label();
                                        viewLabel.setForeground(new Co-
lor(255,255,255));
                                        if (house == house.project.startHouse)
                                        {

                                        viewLabel.setText(„   „+“VIEW_STARTHOUSE“);
                                        }
                                        else
                                        if (house == house.project.targetHouse)
                                        {

                                        viewLabel.setText(„   „+“VIEW_TARGETHOUSE“);
                                        }
                                        else
                                        {

                                        viewLabel.setText(„   „+“VIEW_MORPHHOUSE“);
                                        }
                                        control3DpanelInfo.add(viewLabel,
BorderLayout.WEST);
                                        control3DpanelInfo.setBackground(new
Color(255,0,0));


                    JPanel volumePanel = new JPanel();
                    volumePanel.setBackground(new Color(0,0,0));
                    volumePanel.setLayout(new GridLayout(1,1));
                    house.volumePanelLabel.setForeground(new Color(255,255,255));
                    volumePanel.add(house.volumePanelLabel);


                    JPanel masterControlPanel = new JPanel();
                    masterControlPanel.setLayout(new GridLayout(2,1));
                    masterControlPanel.add(volumePanel);
                    masterControlPanel.add(control3Dpanel);


     GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();

                                        Canvas3D c = new Canvas3D(config);


                    JPanel housePanel1 = new JPanel();
                    housePanel1.setLayout(new BorderLayout());
                    housePanel1.add(c, BorderLayout.CENTER);
                    housePanel1.add(masterControlPanel, BorderLayout.SOUTH);
                    housePanel1.add(control3DpanelInfo, BorderLayout.NORTH);


// Create a simple scene and attach it to the virtual universe
                    BranchGroup scene = createSceneGraph(house);
                    house.u = new SimpleUniverse(c);

                    // add mouse behaviors to the ViewingPlatform
                    ViewingPlatform viewingPlatform = house.u.getViewingPlatfor
m();

                    PlatformGeometry pg = new PlatformGeometry();

                    // Set up the ambient light
                    Color3f ambientColor = new Color3f(0.1f, 0.1f, 0.1f);
                    AmbientLight ambientLightNode = new AmbientLight(ambientC
olor);
                    ambientLightNode.setInfluencingBounds(house.bounds);
                    pg.addChild(ambientLightNode);

                    // Set up the directional lights
                    Color3f light1Color = new Color3f(1.0f, 1.0f, 1.0f);
                    Vector3f light1Direction  = new Vector3f(1.0f, 1.0f, 1.0f);

                    Color3f light2Color = new Color3f(1.0f, 1.0f, 1.0f);
```

```
                    Vector3f light2Direction  = new Vector3f(-1.0f, -1.0f, -1.0f);

                    Color3f light3Color = new Color3f(1.0f, 1.0f, 1.0f);
                    Vector3f light3Direction  = new Vector3f(-1.0f, 1.0f, -1.0f);

                    Color3f light4Color = new Color3f(0.5f, 0.5f, 0.5f);
                    Vector3f light4Direction  = new Vector3f(1.0f, 1.0f, -1.0f);

                    DirectionalLight light1
                       = new DirectionalLight(light1Color, light1Direction);
                    light1.setInfluencingBounds(house.bounds);
                    pg.addChild(light1);

                    DirectionalLight light2
                       = new DirectionalLight(light2Color, light2Direction);
                    light2.setInfluencingBounds(house.bounds);
                    pg.addChild(light2);

                    DirectionalLight light3
                       = new DirectionalLight(light3Color, light3Direction);
                    light3.setInfluencingBounds(house.bounds);
                    pg.addChild(light3);

 DirectionalLight light4
                       = new DirectionalLight(light4Color, light4Direction);
                    light4.setInfluencingBounds(house.bounds);
                    pg.addChild(light4);


                    viewingPlatform.setPlatformGeometry( pg );



                    // This will move the ViewPlatform back a bit so the
                    // objects in the scene can be viewed.
                    viewingPlatform.setNominalViewingTransform();




BufferedReader fr;
String line;

line = „“;

                    if (!spin) {
      OrbitBehavior orbit = new OrbitBehavior(c,

                       OrbitBehavior.REVERSE_ALL);
      BoundingSphere bounds =
         new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 500.0);
      orbit.setSchedulingBounds(bounds);
      viewingPlatform.setViewPlatformBehavior(orbit);
                    }
```

```
			house.u.addBranchGraph(scene);




			   return (housePanel1);


   }


   public BranchGroup POPcreateSceneGraph() {
			// Create the root of the branch graph
			BranchGroup objRoot = new BranchGroup();

			// Create the transform group node and initialize it to the
			// identity.  Enable the TRANSFORM_WRITE capability so that
			// our behavior code can modify it at runtime.  Add it to the
			// root of the subgraph.
			TransformGroup objTrans = new TransformGroup();
			objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_
WRITE);
			objRoot.addChild(objTrans);


			// Create appearance object for textured cube
			Appearance app = new Appearance();
			app.setTexture(tex);
			TextureAttributes texAttr = new TextureAttributes();
			texAttr.setTextureMode(TextureAttributes.MODULATE);
			app.setTextureAttributes(texAttr);

			// Create textured cube and add it to the scene graph.
			Box textureCube = new Box(0.7f, 0.55f, 0.7f,

 Box.GENERATE_TEXTURE_COORDS, app);
			objTrans.addChild(textureCube);

			// Create a new Behavior object that will perform the desired
			// operation on the specified transform object and add it into
			// the scene graph.
			Transform3D yAxis = new Transform3D();
			Alpha rotationAlpha = new Alpha(-1, Alpha.INCREASING_ENAB-
LE,

0, 0,

120000, 0, 0,

0, 0, 0);

			RotationInterpolator rotator =
			   new RotationInterpolator(rotationAlpha, objTrans, yAxis,
								0.0f, (float)
) Math.PI*2.0f);
			BoundingSphere bounds =
			   new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
			rotator.setSchedulingBounds(bounds);
			objTrans.addChild(rotator);

	// Have Java 3D perform optimizations on this scene graph.
	objRoot.compile();

			return objRoot;
   }



  public void GUInewJob()
  {


					NumberFormat nf = NumberFormat.getInstanc
e(java.util.Locale.US);
					nf.setMaximumFractionDigits(0);
					nf.setMinimumFractionDigits(0);
					nf.setMaximumIntegerDigits(2);
					nf.setMinimumIntegerDigits(2);

					GregorianCalendar cal = new GregorianCalen-
dar();

	newJobPanel = new JPanel();

					newJobPanel.setLayout(new GridLayout(3,1));

					Label jobNrLabel = new Label(„ JOB_NUMBER“);
			newJobPanel.add(jobNrLabel);


			JPanel Panel1 = new JPanel();
			Panel1.setLayout(new FlowLayout(FlowLayout.LEFT,0,0));;//new
GridLayout(1,6));

			Label Label1 = new Label(„ Date_Today“);
			Panel1.add(Label1);



			currentDate = nf.format(cal.get(Calendar.YEAR)-2000)+nf.form
at(cal.get(Calendar.MONTH)+1)+nf.format(cal.get(Calendar.DATE));
			TextField tf1 = new TextField( currentDate,6);
			tf1.setEnabled(false);
			Panel1.add(tf1);

			Label Label2 = new Label(„    Project_Nr.“);
			Label2.setForeground(new Color(255,0,0));
			Panel1.add(Label2);
```

```
                    njPNrTf = new TextField(„0815“,6);
                    njPNrTf.setFocusable(true);
                    Panel1.add(njPNrTf);


                    Label Label3 = new Label(„    Number_Of_Morphs“);
                    Label3.setForeground(new Color(255,0,0));
                    Panel1.add(Label3);

                    njNrMorphsTf = new TextField(„30“,2);
                    njNrMorphsTf.setEnabled(true);
                    Panel1.add(njNrMorphsTf);

                    Label Label4 = new Label(„         „);
                    Panel1.add(Label4);

                    Button submitButton = new Button(„ SUBMIT „);
                                    submitButton.addActionListener(this);
                                    submitButton.setActionCommand(„NJSUBMIT“
);
                    Panel1.add(submitButton);




                    newJobPanel.add(Panel1);





                    JPanel Panel2 = new JPanel();
                    Panel2.setLayout(new FlowLayout(FlowLayout.LEFT,0,0));;//
new GridLayout(1,6));
                    Label Label5 = new Label(„
„);
                    Panel2.add(Label5);


                    Label Label6 = new Label(„e.g. 1815“);
                    Panel2.add(Label6);



                    Label Label7 = new Label(„                      0 < morphs < 31“);
                    Panel2.add(Label7);


                    newJobPanel.add(Panel2);


                   startHouseButton.setEnabled(false);
                   targetHouseButton.setEnabled(false);
                   morphButton.setEnabled(false);
```

```
    emptyPanel = new JPanel();


    emptyPanel.setLayout(new FlowLayout());

     GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
                    Canvas3D c = new Canvas3D(config);
     c.setBackground(new Color(0,0,0));

  SimpleUniverse u = null;

// Create a simple scene and attach it to the virtual universe
                    BranchGroup scene = POPcreateSceneGraph();
                    u = new SimpleUniverse(c);

     // This will move the ViewPlatform back a bit so the
     // objects in the scene can be viewed.
                    u.getViewingPlatform().setNominalViewingTransform();

                    u.addBranchGraph(scene);



                      emptyPanel.add(c);



    emptyPanel.setLayout(new GridLayout(1,1));


       if (menuMode != 0)
    {

     if (menuMode == 1)
     {
                    remove(controlPanel);
       remove(viewPanel);
     }

    add (emptyPanel, BorderLayout.CENTER);
    add(newJobPanel,BorderLayout.SOUTH);

    show();
    repaint();
    }



    menuMode = 0;


  }
```

```
 public void GUImorphShow()
 {

 System.gc();


                         saveHouse(StartHouse);
                         saveHouse(TargetHouse);



   morphMorphsPanel = new JPanel();
  ScrollPane morphScrollPane = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEE-
DED);

   morphScrollPane.setSize(400,400);

   morphMorphsPanel.setLayout(new BorderLayout());



   morphScrollPane.add(morphsPanel);
   morphMorphsPanel.add(morphScrollPane, BorderLayout.CENTER);


       createMorphs();


     if (menuMode != 2)
     {

       if (menuMode == 1)
       {
                          remove(controlPanel);
           remove(viewPanel);
       }


                          add(morphMorphsPanel,BorderLayout.CENTER);
                         }

     show();
     repaint();

                                                   morphsVisible = 1;

                         menuMode = 2;
 }


  public void GUImorph()
  {
```

```
 NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
 nf.setMaximumFractionDigits(0);
 nf.setMinimumFractionDigits(0);
 nf.setMaximumIntegerDigits(2);
 nf.setMinimumIntegerDigits(2);




   newMorphsPanel = new JPanel();

                                                 newMorphsPanel.setLayout(new GridLay-
out(3,1));

                                                 Label mLabel = new Label(„ MORPHS“);
                              newMorphsPanel.add(mLabel);


                              JPanel Panel1 = new JPanel();
                              Panel1.setLayout(new FlowLayout(FlowLayout.LEFT,0,0));;//new
GridLayout(1,6));

                              Label Label1 = new Label(„ Number of Morphs“);
                              Label1.setForeground(new Color(255,0,0));
                              Panel1.add(Label1);



                              morphNrTf = new TextField(„“+project.morphSteps,2);
                              Panel1.add(morphNrTf);

                              Label Label4 = new Label(„         „);
                              Panel1.add(Label4);

                              Button submitButton = new Button(„ SUBMIT „);
                                                 submitButton.addActionListener(this);
                                                 submitButton.setActionCommand(„MPSUBMIT
“);
                              Panel1.add(submitButton);


                              newMorphsPanel.add(Panel1);





       emptyPanel = new JPanel();


       emptyPanel.setLayout(new FlowLayout());

        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
```

```
                    Canvas3D c = new Canvas3D(config);
      c.setBackground(new Color(0,0,0));
                         emptyPanel.add(c);



     emptyPanel.setLayout(new GridLayout(1,1));

      if (menuMode != 2)
     {

       if (menuMode == 1)
       {
                         remove(controlPanel);
         remove(viewPanel);
       }

     add (emptyPanel, BorderLayout.CENTER);
     add (newMorphsPanel,BorderLayout.SOUTH);

     show();
     repaint();
     menuMode = 2;
                         }



                    saveHouse(StartHouse);
                    saveHouse(TargetHouse);


        //createMorphs();
                      //add(morphMorphsPanel,BorderLayout.CENTER);
   }

                                        public void GUIupdateHouseParams(THouse
house)
                                        {


                                                    sthHWtf.setText(„“+house.Hou
seWidth);
                                                    sthHLtf.setText(„“+house.HouseLen
gth);
                                                    sthEHtf.setText(„“+house.EavesHei
ght);
                                                    sthRHtf.setText(„“+house.RidgeHei
ght);

                                        }


                      public void GUIupdateSectionPoints(THouse house)
    {
```

```
                     for (int i=0;i<10;i++)
                     {
                                GUIupdateSectionPoint(i,house);
                     }
   }

                      public void GUIupdateSectionPoint(int P, THouse house)
                      {

                        house.convertP2rP();

 NumberFormat nf = NumberFormat.getInstance( java.util.Locale.US);
 nf.setMaximumFractionDigits(2);
 nf.setMinimumFractionDigits(2);
 nf.setMaximumIntegerDigits(2);
 nf.setMinimumIntegerDigits(2);
                                Tfx[P].setText(nf.format( house.rP[P].x));
                                Tfy[P].setText(nf.format( house.rP[P].y));
                                Tfz[P].setText(nf.format( house.rP[P].z));
                      }



public void GUIsthDatabase()
{


                      jobIDlabel.setForeground(new Color(255,0,0));
                      jobIDtf.setEnabled(true);
                      sthSubmitButton.setEnabled(true);

                     sthHLl.setForeground(new Color(0,0,0));
                     sthHWl.setForeground(new Color(0,0,0));
                     sthEHl.setForeground(new Color(0,0,0));
                     sthRHl.setForeground(new Color(0,0,0));

                      sthHLtf.setEnabled(false);
   sthHWtf.setEnabled(false);
   sthEHtf.setEnabled(false);
   sthRHtf.setEnabled(false);

                     sthMode = 1;

}


public void GUIsthParams()
{

   sthHLtf.setEnabled(true);
   sthHWtf.setEnabled(true);
   sthEHtf.setEnabled(true);
   sthRHtf.setEnabled(true);
                      sthSubmitButton.setEnabled(true);
```

```
                        sthHLl.setForeground(new Color(255,0,0));
                        sthHWl.setForeground(new Color(255,0,0));
                        sthEHl.setForeground(new Color(255,0,0));
                        sthRHl.setForeground(new Color(255,0,0));

                        jobIDtf.setEnabled(false);
                        jobIDlabel.setForeground(new Color(0,0,0));

                        sthMode = 2;
}


public String getNewJobNr()
{

  return („0001“);

}


public void updateHouse(THouse house)
{

  for (int i=0; i<10;i++)
  {
   changePoint(i, house.P[i].x,house.P[i].y,house.P[i].z, house);
  }
}


public void         createProject()
{


   String ProjectNumber = njPNrTf.getText();//(int) (Float.parseFloat(morphNrT
f.getText()));
   int NrMorphs = (int) (Float.parseFloat(njNrMorphsTf.getText()));


   if ((ProjectNumber.length() == 4 ) &&
      (NrMorphs >0) &&
      (NrMorphs <31)
     )
   {

                                        project = new TProject();
                        project.date = currentDate;


                          project.projectNr = njPNrTf.getText();//(int) (Float.parseFloat(
morphNrTf.getText()));

      project.jobNr = getNewJobNr();
      project.morphSteps = NrMorphs;
```

```
                        houseArray = new THouse[2];
                        houseArrayCount = 0;
                        StartHouse = createHouse(project);
                        TargetHouse = createHouse(project);


                                          StartHouse.morphIndex = 0;
                                          TargetHouse.morphIndex =
project.morphSteps+1;


                        project.startHouse = StartHouse;
                        project.targetHouse = TargetHouse;

                        modifyHouse = StartHouse;


                                          GUIstartHouse();
                        }
}


// Generiere House anhand HouseLength, etc.
public void generateHouse(THouse house)
{


   double bV = 0;


   if (house.HouseWidth> bV) { bV = house.HouseWidth; };
   if (house.HouseLength> bV) { bV = house.HouseLength; };
   if (house.EavesHeight > bV) { bV = house.EavesHeight; };
   if (house.RidgeHeight > bV) { bV = house.RidgeHeight; };


   bV = 16;
   bV = bV/2;

   house.HouseWidth = house.HouseWidth/bV;
   house.HouseLength = house.HouseLength/bV;
   house.EavesHeight = house.EavesHeight/bV;
   house.RidgeHeight = house.RidgeHeight/bV;




   // Startpunkt ist normalerweise auf 0/0/0
   house.P[3].x = -house.HouseWidth/2;
   house.P[3].y = -house.HouseLength/2;
   house.P[3].z = -house.RidgeHeight/2;

   house.P[0].x = house.P[3].x + house.HouseWidth;
   house.P[0].y = house.P[3].y;
   house.P[0].z = house.P[3].z;

   house.P[1].x = house.P[0].x;
   house.P[1].y = house.P[0].y + house.HouseLength;
   house.P[1].z = house.P[0].z;

   house.P[2].x = house.P[3].x;
   house.P[2].y = house.P[1].y;
   house.P[2].z = house.P[3].z;

   house.P[4].x = house.P[3].x;
   house.P[4].y = house.P[3].y;
   house.P[4].z = house.P[3].z + house.EavesHeight;

   house.P[5].x = house.P[3].x + house.HouseWidth/2;
   house.P[5].y = house.P[3].y;
   house.P[5].z = house.P[3].z + house.RidgeHeight;

   house.P[6].x = house.P[0].x;
   house.P[6].y = house.P[0].y;
   house.P[6].z = house.P[0].z + house.EavesHeight;

   house.P[7].x = house.P[1].x;
   house.P[7].y = house.P[1].y;
   house.P[7].z = house.P[1].z + house.EavesHeight;


   house.P[8].x = house.P[2].x;
   house.P[8].y = house.P[2].y;
   house.P[8].z = house.P[2].z + house.EavesHeight;

   house.P[9].x = house.P[8].x + house.HouseWidth/2;
   house.P[9].y = house.P[8].y;
   house.P[9].z = house.P[5].z;


   modifyHouse.CreateKoordinatengleichungen();
                  updateHouse(modifyHouse);
}



public void GUIsthSubmit()
{


                     sthHLl.setForeground(new Color(0,0,0));
                     sthHWl.setForeground(new Color(0,0,0));
                     sthEHl.setForeground(new Color(0,0,0));
                     sthRHl.setForeground(new Color(0,0,0));

                     sthHLtf.setEnabled(false);
   sthHWtf.setEnabled(false);
```

```
   sthEHtf.setEnabled(false);
   sthRHtf.setEnabled(false);

                  jobIDtf.setEnabled(false);
                  jobIDlabel.setForeground(new Color(0,0,0));

                  sthSubmitButton.setEnabled(false);


   // parameters
                 if (sthMode == 2)
                 {

                 modifyHouse.HouseLength = Float.parseFloat(sthHLtf.getText());
                 modifyHouse.HouseWidth = Float.parseFloat(sthHWtf.getText())
;
                 modifyHouse.EavesHeight = Float.parseFloat(sthEHtf.getText());
                 modifyHouse.RidgeHeight = Float.parseFloat(sthRHtf.getText());

  generateHouse(modifyHouse);
                 }


  //database
  if (sthMode == 1)
  {

                  String job;

                  job =           jobIDtf.getText();
                  if (job.length()==16)
                  {


     loadHouse(modifyHouse, job);

     modifyHouse.CreateKoordinatengleichungen();
                   updateHouse(modifyHouse);

    }

  }

}




 public void GUIinit()
 {


                 setLayout(new BorderLayout(0,0));
 setBackground(new Color(255,255,255));


JPanel menuPanel = new JPanel();
menuPanel.setLayout(new FlowLayout(FlowLayout.LEFT,0,0));
menuPanel.setBackground(new Color(255,255,255));




newJobButton =  new Button(„New Job“);
newJobButton.setBackground(new Color(255,255,255));
newJobButton.addActionListener(this);
menuPanel.add(newJobButton);

startHouseButton = new Button(„StartHouse“);
startHouseButton.setBackground(new Color(255,255,255));
startHouseButton.addActionListener(this);
menuPanel.add(startHouseButton);

targetHouseButton = new Button(„TargetHouse“);
targetHouseButton.setBackground(new Color(255,255,255));
targetHouseButton.addActionListener(this);
menuPanel.add(targetHouseButton);


morphButton = new Button(„Morph“);
morphButton.setBackground(new Color(255,255,255));
morphButton.addActionListener(this);
menuPanel.add(morphButton);


Button databaseButton = new Button(„Database“);
databaseButton.setBackground(new Color(255,255,255));
databaseButton.addActionListener(this);
menuPanel.add(databaseButton);



 add(menuPanel, BorderLayout.NORTH);

menuMode = -1;

 }


                public void itemStateChanged(ItemEvent event)
                {
                                Checkboxcb=(Checkbox)event.getItemSelectable(
);
                                for (int i=0; i<6; i++)
                                {
                                                if(cb==cbs[i]){changePlane=i+2;
}
```

```
                                }


      scb1.setEnabled(true);
                  scb2.setEnabled(true);
                  scb3.setEnabled(true);
                  rscb1.setEnabled(true);
                  rscb2.setEnabled(true);
                  rscb3.setEnabled(true);

      scb1.setValue( (int)(modifyHouse.mods[changePlane][0]*100+100) );
      scb2.setValue( (int)(modifyHouse.mods[changePlane][1]*100+100) );
      scb3.setValue( (int)(modifyHouse.mods[changePlane][2]*100+100) );

      rscb1.setValue( (int)(modifyHouse.mods[changePlane][3]+180) );
      rscb2.setValue( (int)(modifyHouse.mods[changePlane][4]+180) );
      rscb3.setValue( (int)(modifyHouse.mods[changePlane][5]+180) );


      modifyHouse.startWerte.copyHouse2Points(modifyHouse);

                                //System.out.println(„plane:“+changePlane);

                                for (int i=1;i<8;i++)
                                {
                                                if (modifyHouse.GUIwireFrame
== 1) { setWireFrame(i,modifyHouse.hashFaces); }
                                                else { setSolidFace(i,modifyHou
se.hashFaces); };
                                }

                                setActiveFace(changePlane,
modifyHouse.hashFaces);

                }


public void adjustmentValueChanged(AdjustmentEvent event)
{
  Adjustable sb = event.getAdjustable();
  double valueS, valueW;



  valueS = event.getValue();
  valueS = (valueS-100)/100;

  valueW = event.getValue();
  valueW = (valueW-180);



  if (sb == scb1)
  {
```

```
  MoveLayer(modifyHouse,changePlane,valueS,XAxis);
 }

  if (sb == scb2)
   {

  MoveLayer(modifyHouse,changePlane,valueS,YAxis);
 }
  if (sb == scb3)
   {

  MoveLayer(modifyHouse,changePlane,valueS,ZAxis);
  //System.out.println(„Z:"+valueS);
 }



   if (sb == rscb1)
   {

   //System.out.println(valueW);
   //System.out.println(„xmods:"+modifyHouse.mods[changePlane][3]);
  RotateLayer(modifyHouse,changePlane,valueW-modifyHouse.mods[changePla
ne][3+XAxis],XAxis, valueW);


   //System.out.println(„ymods:"+modifyHouse.mods[changePlane][3]);

 }

  if (sb == rscb2)
   {

  RotateLayer(modifyHouse,changePlane,valueW-modifyHouse.mods[changePla
ne][3+YAxis],YAxis, valueW);
 }
  if (sb == rscb3)
   {

  RotateLayer(modifyHouse,changePlane,valueW-modifyHouse.mods[changePla
ne][3+ZAxis],ZAxis, valueW);
 }



 updateHouse(modifyHouse);

}


 public void loadHouse(THouse house, String job) //String date, String projectNr,
String jobNr, int morphIndex)
 {
```

```
  StartHouse.RsX = 99;

   BufferedReader fr;
   String line;
                        double px,py,pz;


  NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
  nf.setMaximumFractionDigits(0);
  nf.setMinimumFractionDigits(0);
  nf.setMaximumIntegerDigits(2);
  nf.setMinimumIntegerDigits(2);

   line = „";
   try {

                          fr = new BufferedReader(new FileReader(„data/"+job));
//date+"_"+projectNr+"_"+jobNr+"_"+nf.format(morphIndex)));


 for (int i=0; i<10;i++)
   {


     px = Float.parseFloat(fr.readLine());
     py = Float.parseFloat(fr.readLine());
     pz = Float.parseFloat(fr.readLine());

     changePoint(i, px, py, pz, house);
    }



                        house.RsX = Float.parseFloat(fr.readLine()); //.toValue();
                        house.RsY = Float.parseFloat(fr.readLine()); //.toValue();
                        house.RsZ = Float.parseFloat(fr.readLine()); //.toValue();

                        house.ReX = Float.parseFloat(fr.readLine()); //.toValue();
                        house.ReY = Float.parseFloat(fr.readLine()); //.toValue();
                        house.ReZ = Float.parseFloat(fr.readLine()); //.toValue();


    house.HouseLength = Float.parseFloat(fr.readLine()); //.toValue();
    house.HouseWidth = Float.parseFloat(fr.readLine()); //.toValue();
    house.EavesHeight = Float.parseFloat(fr.readLine()); //.toValue();
    house.RidgeHeight = Float.parseFloat(fr.readLine()); //.toValue();



     for (int j=0; j<8;j++) {
                        for (int k=0; k<6;k++)
                        {
                                    house.mods[j][k]=Float.parseFloat(fr.readLine());
```

```
                    }
   }
                    fr.close();


  }
  catch (IOException e) {
                    System.out.println(„filreaderror“);
  }


 }

 public void saveHouse(THouse house)
 {
                     Writer f1;
  BufferedWriter f2;


   NumberFormat nf = NumberFormat.getInstance(java.util.Locale.US);
   nf.setMaximumFractionDigits(0);
   nf.setMinimumFractionDigits(0);
   nf.setMaximumIntegerDigits(2);
   nf.setMinimumIntegerDigits(2);



try {
                     f1 = new FileWriter(„data/“+house.project.date+““+house.proje
ct.projectNr+““+house.project.jobNr+““+nf.format(house.morphIndex));
                     f2 = new BufferedWriter(f1);


  for (int i=0;i<10;i++){


  getPointFromModel(i,house.P[i], house.hashFaces);


                    f2.write(„“+house.P[i].x);
                    f2.newLine();
                    f2.write(„“+house.P[i].y);
                    f2.newLine();
                    f2.write(String.valueOf(house.P[i].z));
                    f2.newLine();

}


                    f2.write(„“+house.RsX);
                    f2.newLine();
                    f2.write(„“+house.RsY);
                    f2.newLine();
                    f2.write(„“+house.RsZ);
                    f2.newLine();

                    f2.write(„“+house.ReX);
                    f2.newLine();
                    f2.write(„“+house.ReY);
                    f2.newLine();
                    f2.write(„“+house.ReZ);
                    f2.newLine();




  f2.write(„“+house.HouseLength);
  f2.newLine();
                    f2.write(„“+house.HouseWidth);
  f2.newLine();

  f2.write(„“+house.EavesHeight);
  f2.newLine();
                    f2.write(„“+house.RidgeHeight);
  f2.newLine();


  for (int j=0; j<8;j++) {
                    for (int k=0; k<6;k++)
                    {
                                    f2.write(„“+house.mods[j][k]);
                                    f2.newLine();
                    }
  }


                     f2.close();
                     f1.close();
 }
 catch (IOException e) {
                    System.out.println(„ERROR“);
 }



}




 public void GUIstHouse()
 {



                    viewPanel = new JPanel();
  viewPanel.setLayout(new GridLayout(1,2));

 controlPanel = new JPanel();
```

```
controlPanel.setLayout(new GridLayout(2,1));


JPanel definePanel = new JPanel();
definePanel.setLayout(new BorderLayout()); //GridLayout(2,1));

JPanel dP1 = new JPanel();
dP1.setLayout(new BorderLayout());
Label label1 = new Label(„ START HOUSE_PARAMETERS“);

JPanel dp1_1 = new JPanel();
dp1_1.setLayout(new FlowLayout(FlowLayout.LEFT));
Button button1 = new Button(„Database_Select“);
button1.setActionCommand(„STHDB“);
button1.addActionListener(this);
dp1_1.add(button1);



sthParamButton = new Button(„Parameters“);
sthParamButton.setActionCommand(„STHPARAMS“);
sthParamButton.addActionListener(this);
dp1_1.add(sthParamButton);


sthSubmitButton = new Button(„ SUBMIT „);
sthSubmitButton.setActionCommand(„STHSUBMIT“);
sthSubmitButton.setEnabled(false);
sthSubmitButton.addActionListener(this);
dp1_1.add(sthSubmitButton);



dP1.add(label1, BorderLayout.NORTH);
dP1.add(dp1_1, BorderLayout.SOUTH);



definePanel.add(dP1,BorderLayout.NORTH);


JPanel dP2 = new JPanel();
//dP2.setLayout(new GridLayout(1,2));

dP2.setLayout(new FlowLayout(FlowLayout.LEFT));

JPanel dp2_1 = new JPanel();
dp2_1.setLayout(new FlowLayout(FlowLayout.LEFT));
jobIDlabel = new Label(„Job_ID“);
dp2_1.add(jobIDlabel);

jobIDtf = new TextField(„0“,14);
jobIDtf.setEnabled(false);
dp2_1.add(jobIDtf);
dP2.add(dp2_1);

JPanel panelNull = new JPanel();
Label label8 = new Label(„  „);
panelNull.add(label8);
dP2.add(panelNull);

JPanel dp2_2 = new JPanel();
dp2_2.setLayout(new GridLayout(2,3));

JPanel dp2_2_1 = new JPanel();


sthHLl = new Label(„L: House_Length“);
dp2_2_1.add(sthHLl);
sthHLtf = new TextField(„16.0“,6);
sthHLtf.setEnabled(false);
dp2_2_1.add(sthHLtf);
dp2_2.add(dp2_2_1);


JPanel dp2_2_2 = new JPanel();
sthHWl = new Label(„W: House_Width“);
dp2_2_2.add(sthHWl);

sthHWtf = new TextField(„8.0“,6);
sthHWtf.setEnabled(false);
dp2_2_2.add(sthHWtf);
dp2_2.add(dp2_2_2);

JPanel dp2_2_3 = new JPanel();
dp2_2.add(dp2_2_3);

JPanel dp2_2_4 = new JPanel();
sthEHl = new Label(„E: Eaves_Height“);
dp2_2_4.add(sthEHl);

sthEHtf = new TextField(„4.0“,6);
sthEHtf.setEnabled(false);
dp2_2_4.add(sthEHtf);
dp2_2.add(dp2_2_4);

JPanel dp2_2_5 = new JPanel();
sthRHl = new Label(„R: Ridge_Height“);
dp2_2_5.add(sthRHl);
sthRHtf = new TextField(„8.0“,6);
sthRHtf.setEnabled(false);
dp2_2_5.add(sthRHtf);
dp2_2.add(dp2_2_5);

JPanel dp2_2_6 = new JPanel();
Label label7 = new Label(„0 < Parameters < 100“);
dp2_2_6.add(label7);
dp2_2.add(dp2_2_6);
```

```
dP2.add(dp2_2);

definePanel.add(dP2,BorderLayout.CENTER);

controlPanel.add(definePanel);




JPanel manipulationPanel = new JPanel();


manipulationPanel.setBackground(new Color(180,180,180));

                    manipulationPanel.setLayout(null);
                    Label label9 = new Label („LAYER_MANIPULATION“);
                    label9.setBounds(4,0, 200,20);
                    manipulationPanel.add(label9);


cbg = new CheckboxGroup();

for (int i=0;i<6;i++) {

                    cbs[i] = new Checkbox(„Plane „+(i+2),cbg,false);
   cbs[i].addItemListener(this);

   cbs[i].setBounds(60+i*70,30,70,20);
   manipulationPanel.add(cbs[i]);
};

                    Label label10 = new Label („MOVE“);
                    label10.setBounds(4,45, 50,20);
                    manipulationPanel.add(label10);

                    Label label11 = new Label („ROTATE“);
                    label11.setBounds(4,105, 50,20);
                    manipulationPanel.add(label11);


Label label12 = new Label („x“);
label12.setBounds(60,45,120,20);
manipulationPanel.add(label12);

scb1 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,210);
scb1.setBounds(60,65,120,20);
scb1.addAdjustmentListener(this);
manipulationPanel.add(scb1);



Label label13 = new Label („y“);
label13.setBounds(200,45,120,20);
manipulationPanel.add(label13);

scb2 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,210);
scb2.setBounds(200,65,120,20);
scb2.addAdjustmentListener(this);
manipulationPanel.add(scb2);


Label label14 = new Label („z“);
label14.setBounds(340,45,120,20);
manipulationPanel.add(label14);

scb3 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,210);
scb3.setBounds(340,65,120,20);
scb3.addAdjustmentListener(this);
manipulationPanel.add(scb3);

controlPanel.add(manipulationPanel);
add (controlPanel, BorderLayout.SOUTH);
controlPanel.setBackground(new Color(255,255,100));



Label label15 = new Label („x“);
label15.setBounds(60,85,120,20);
manipulationPanel.add(label15);

rscb1 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,370);
rscb1.setBounds(60,105,120,20);
rscb1.addAdjustmentListener(this);
manipulationPanel.add(rscb1);


Label label16 = new Label („y“);
label16.setBounds(200,85,120,20);
manipulationPanel.add(label16);

rscb2 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,370);
rscb2.setBounds(200,105,120,20);
rscb2.addAdjustmentListener(this);
manipulationPanel.add(rscb2);


Label label17 = new Label („z“);
label17.setBounds(340,85,120,20);
manipulationPanel.add(label17);

rscb3 = new Scrollbar(Scrollbar.HORIZONTAL,100,10,0,370);
rscb3.setBounds(340,105,120,20);
rscb3.addAdjustmentListener(this);
manipulationPanel.add(rscb3);

controlPanel.add(manipulationPanel);
add (controlPanel, BorderLayout.SOUTH);
controlPanel.setBackground(new Color(255,255,100));
```

```
 Label[] PLabel = new Label[10];

 for (int i=0; i<10;i++)
 {

                    PLabel[i] = new Label(„P“+i);
                    PLabel[i].setBounds(500+(i%2)*155, ((i/2))*20+25,20,20);
                    manipulationPanel.add(PLabel[i]);


                    Tfx[i] = new TextField(„0.00“,7);
                    Tfx[i].setBounds(520+(i%2)*155, ((i/2))*20+25,42,20);
                    Tfx[i].setEnabled(false);
                    manipulationPanel.add(Tfx[i]);

                    Tfy[i] = new TextField(„0.00“,7);
                    Tfy[i].setBounds(520+(i%2)*155+45, ((i/2))*20+25,42,20);
                    Tfy[i].setEnabled(false);
                    manipulationPanel.add(Tfy[i]);

                    Tfz[i] = new TextField(„0.00“,7);
                    Tfz[i].setBounds(520+(i%2)*155+90, ((i/2))*20+25,42,20);
                    Tfz[i].setEnabled(false);
                    manipulationPanel.add(Tfz[i]);

 }


                    Label label19 = new Label („SECTION_POINTS“);
                    label19.setBounds(500,0, 200,20);
                    manipulationPanel.add(label19);



  viewPanel.add(createShape(StartHouse));
  viewPanel.add(createShape(TargetHouse));



                      startHouseButton.setEnabled(true);
                      targetHouseButton.setEnabled(true);
                      morphButton.setEnabled(true);




}

public void GUIstartHouse()
{


                    if (menuMode == 0) // New Job
                    {
                                                                  remove(emptyPanel);
                                                                  remove(newJobPanel);

                                                                  GUIstHouse();


     add(viewPanel, BorderLayout.CENTER);
                    add (controlPanel, BorderLayout.SOUTH);
     show();
     repaint();
   }


                    if (menuMode == 2)
                    {


   remove(morphMorphsPanel);
   add(viewPanel,BorderLayout.CENTER);
                                        add (controlPanel, BorderLayout.SOUTH);
   show();
   repaint();
   System.gc();
                    }

 menuMode = 1;


                    saveHouse(modifyHouse);
                    modifyHouse = StartHouse;

  for (int i=0;i<6;i++) {

                    cbs[i].setState(false);
                    };

                    scb1.setEnabled(false);
                    scb2.setEnabled(false);
                    scb3.setEnabled(false);
                    rscb1.setEnabled(false);
                    rscb2.setEnabled(false);
                    rscb3.setEnabled(false);

                    changePlane = 0;
                    cbg.setSelectedCheckbox(null);
                    resetHouseFace(modifyHouse);
                    resetHouseFace(TargetHouse);


                                        GUIupdateHouseParams(modifyHouse);
                    GUIupdateSectionPoints(modifyHouse);


}
```

```
public void GUItargetHouse()
{


                    if (menuMode == 2)
                    {


     remove(morphMorphsPanel);
     add(viewPanel,BorderLayout.CENTER);
                                          add (controlPanel, BorderLayout.SOUTH);
     show();
     repaint();
                    }


                    menuMode = 1;

                    saveHouse(modifyHouse);
                    modifyHouse = TargetHouse;

    for (int i=0;i<6;i++) {

                    cbs[i].setState(false);
                    };
                    scb1.setEnabled(false);
                    scb2.setEnabled(false);
                    scb3.setEnabled(false);
                    rscb1.setEnabled(false);
                    rscb2.setEnabled(false);
                    rscb3.setEnabled(false);

                    changePlane = 0;
                    cbg.setSelectedCheckbox(null);
                    resetHouseFace(modifyHouse);
                    resetHouseFace(StartHouse);

                                         GUIupdateHouseParams(modifyHouse);
    GUIupdateSectionPoints(modifyHouse);

 }


public void actionPerformed(ActionEvent event)
{



 THouse house;

 String cmd = event.getActionCommand();

 //System.out.println(cmd);
```

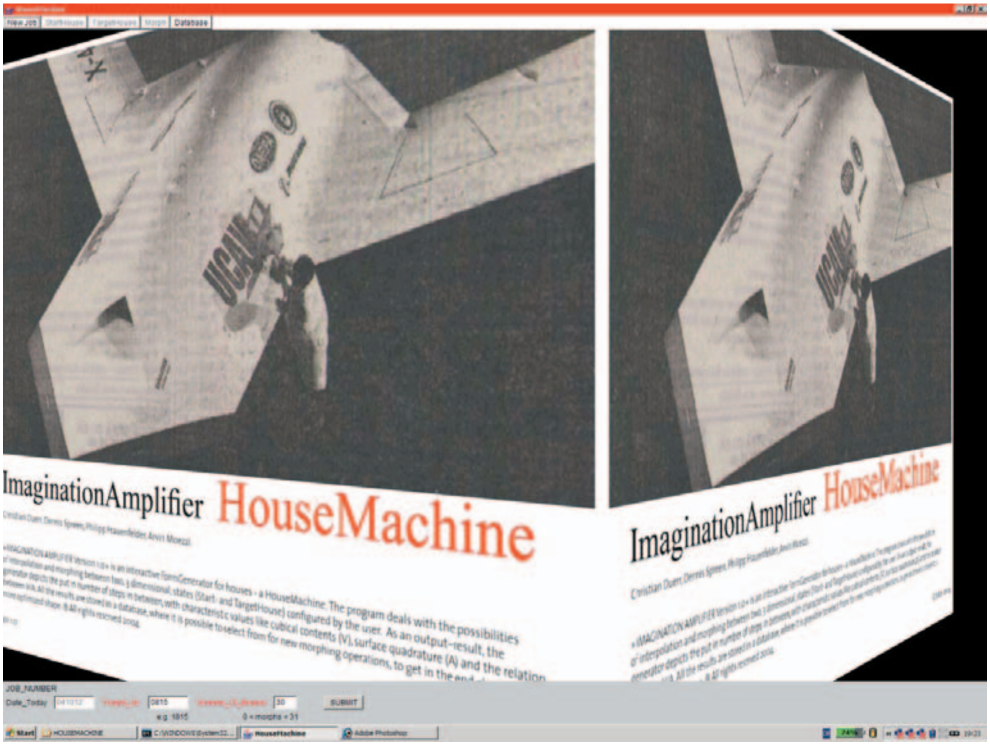
```
  for (int j=0;j<=houseArrayCount;j++){


   if (cmd.equals(„WFH“+j)) {

                  house = houseArray[j];

                  if (house.GUIwireFrame == 1) {

                                      house.wireButton.setLabel(„Wireframe“);
                                      house.GUIwireFrame = 0;
                  }
                  else
                  {
                                      house.wireButton.setLabel(„Solid“);
                                      house.GUIwireFrame = 1;
                  }


                  resetHouseFace(house);


                  if (changePlane>0) { setActiveFace(changePlane,
house.hashFaces); }

   }
  }


  if (cmd.equals(„New Job“))
  {

                  GUInewJob();
  }


  if (cmd.equals(„STHSUBMIT“))
  {

                  GUIsthSubmit();

  }

  if (cmd.equals(„STHPARAMS“))
  {
                  GUIsthParams();

  }


  if (cmd.equals(„STHDB“))
  {

                                      GUIsthDatabase();
```

```
 }


 if (cmd.equals(„NJSUBMIT“))
 {

                    createProject();

 }


  if (cmd.equals(„MPSUBMIT“))
  {

                    GUImorphShow();
  }

  if (cmd.equals(„StartHouse“))
  {

    GUIstartHouse();
  }



  if (cmd.equals(„TargetHouse“))
  {
   GUItargetHouse();

  }


  if (cmd.equals(„Morph“))
  {

                                          GUImorphShow();

                     }


   if (cmd.equals(„Database“)) {

                    }

}



public HouseMachine()
{


 super(VERSION);
```

```
  //Panel zur Versionsanzeige hinzufügen

  setSize(850,700);

  tex = new TextureLoader(„welcome.jpg“, this).getTexture();

  GUIinit();
  GUInewJob();

  setVisible(true);
 }






 public static void main(String[] args)
 {

  HouseMachine frame = new HouseMachine();
  frame.setLocation(100, 100);
  frame.pack();
  frame.setVisible(true);

 }



 //---Private Methoden---------------


 private void setCtrlAccelerator(JMenuItem mi, char acc)
 {
  KeyStroke ks = KeyStroke.getKeyStroke(
   acc, Event.CTRL_MASK
  );
  mi.setAccelerator(ks);
 }


}
```

D2 The HouseMachine

D2.1 NewJob | Programm opener, input of Project_Nr. and Number of Morphs



ImaginationAmplifier HouseMachine

Cristian Duerr, Dennis Spreen, Philipp Hauenfelder, Arvin Moezzi

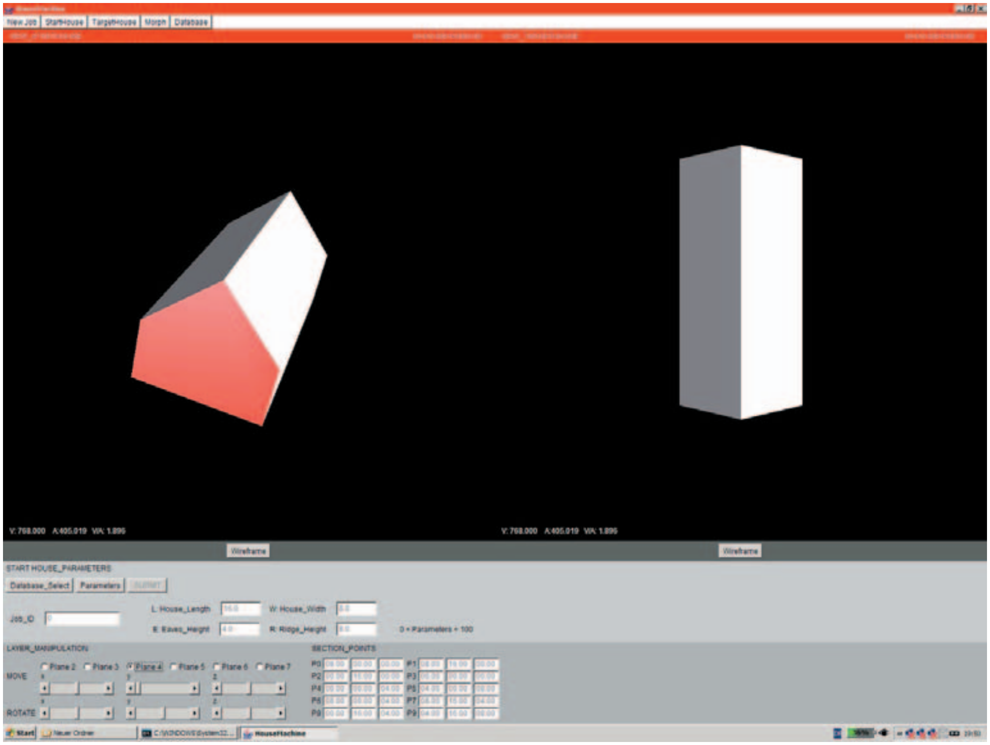
»IMAGINATION AMPLIFIER Version 1.0« is an interactive FormGenerator for houses – a HouseMachine. The program deals with the possibilities of interpolation and morphing between two, 3 dimensional, states (Start- and TargetHouse) configured by the user. As an output-result, the generator depicts the put in number of steps in between, with characteristic values like cubical contents (V), surface quadrature (A) and the relation between V/A. All the results are stored in a database, where it is possible to select from for new morphing operations, to get in the end

more optimized shape. © All rights reserved 2004.

JOB_NUMBER

Date_Today 041012 [illegible] 0815 e.g. 1815 [illegible] 30 0 < morphs < 31

D2.2 StartHouse | Configuration of StartHouse, Input of House_Parameters or Database_Select as well as Planes-Manipulation, rotation and moving.



V: 768.000 A:405.019 V/A: 1.896

V: 768.000 A:405.019 V/A: 1.896

START HOUSE_PARAMETERS

Job_ID: 0

L: House_Length: 16.0 W: House_Width: 8.0

E: Eaves_Height: 4.0 R: Ridge_Height: 8.0

0 < Parameters < 100

LAYER_MANIPULATION

Plane 2 · Plane 3 · [✓ Plane 4] · Plane 5 · Plane 6 · Plane 7

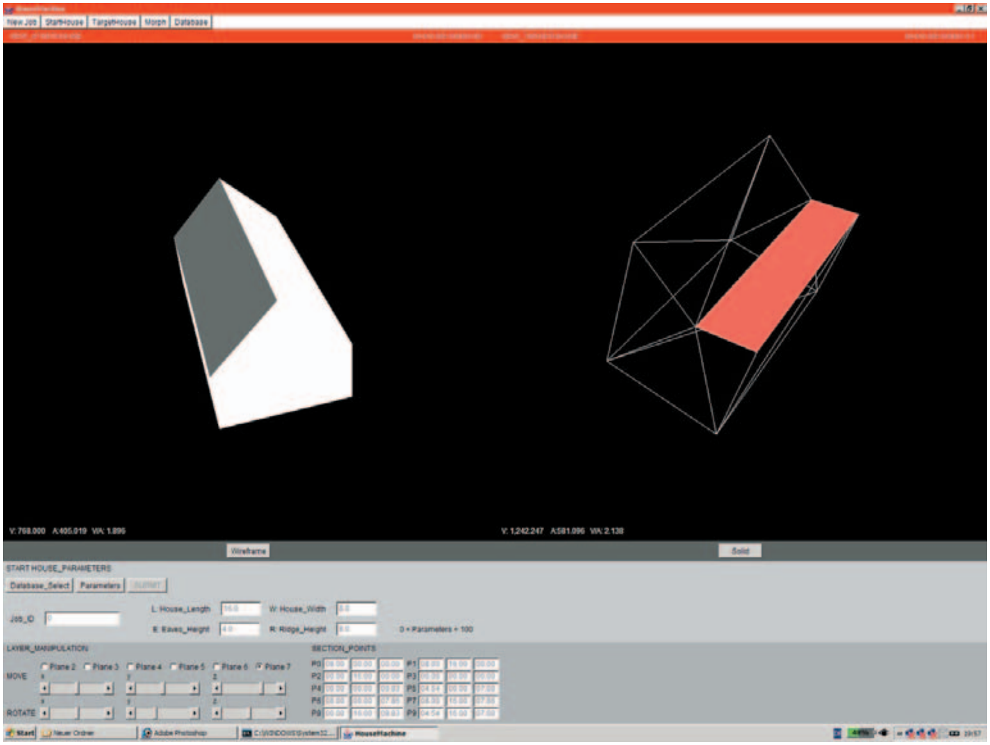
MOVE x y z

ROTATE x y z

SECTION_POINTS

P0	08.00	00.00	00.00	P1	08.00	16.00	00.00
P2	00.00	16.00	00.00	P3	00.00	00.00	00.00
P4	00.00	00.00	04.00	P5	04.00	00.00	08.00
P6	08.00	00.00	04.00	P7	08.00	16.00	04.00
P8	00.00	16.00	04.00	P9	04.00	16.00	08.00

D2.3 TargetHouse | Configuration of TargetHouse, Input of House_Parameters or Database_Select as well as Planes-Manipulation, rotation and moving.



V: 768.000 A:405.019 V/A: 1.896

V: 1,242.247 A:581.096 V/A: 2.138

START HOUSE_PARAMETERS

Database_Select | Parameters | SUBMIT

Job_ID: 0

L: House_Length: 16.0 W: House_Width: 8.0

E: Eaves_Height: 4.0 R: Ridge_Height: 8.0 0 < Parameters < 100

LAYER_MANIPULATION

MOVE: Plane 2 Plane 3 Plane 4 Plane 5 Plane 6 [✓ Plane 7]

x y z

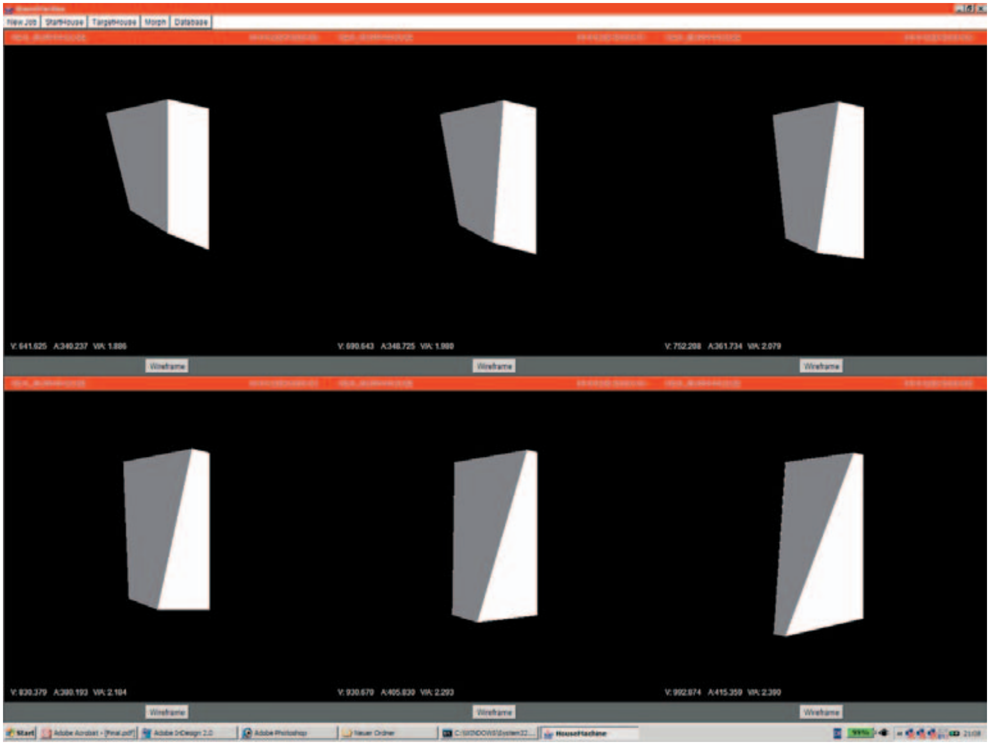
ROTATE

x y z

SECTION_POINTS

P0	08.00	00.00	00.00	P1	08.00	16.00	00.00
P2	00.00	16.00	00.00	P3	00.00	00.00	00.00
P4	00.00	00.00	03.83	P5	04.54	00.00	07.00
P6	08.00	00.00	07.85	P7	08.00	16.00	07.85
P8	00.00	16.00	03.83	P9	04.54	16.00	07.00

D2.4 Morph | Interpolation between Start- and TargetHouse in Number_of_Morphs-Steps.



V: 641.625 A:340.237 V/A: 1.886

V: 690.643 A:348.725 V/A: 1.980

V: 752.208 A:361.734 V/A: 2.079

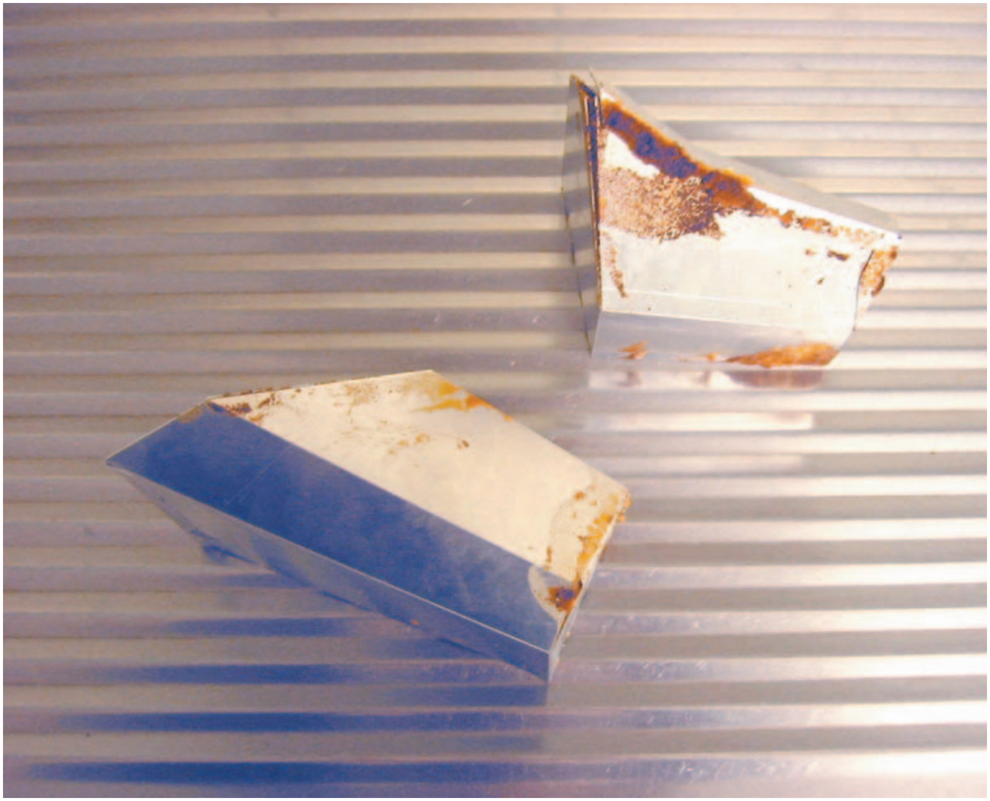
V: 830.379 A:380.193 V/A: 2.184

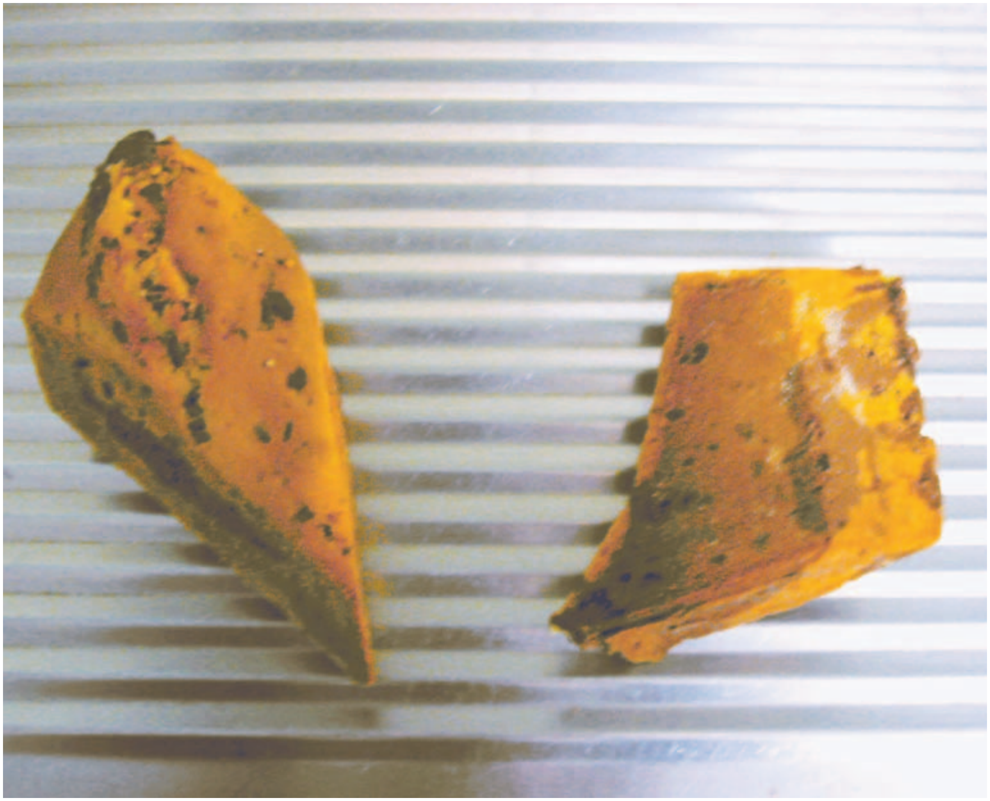
V: 930.670 A:405.830 V/A: 2.293

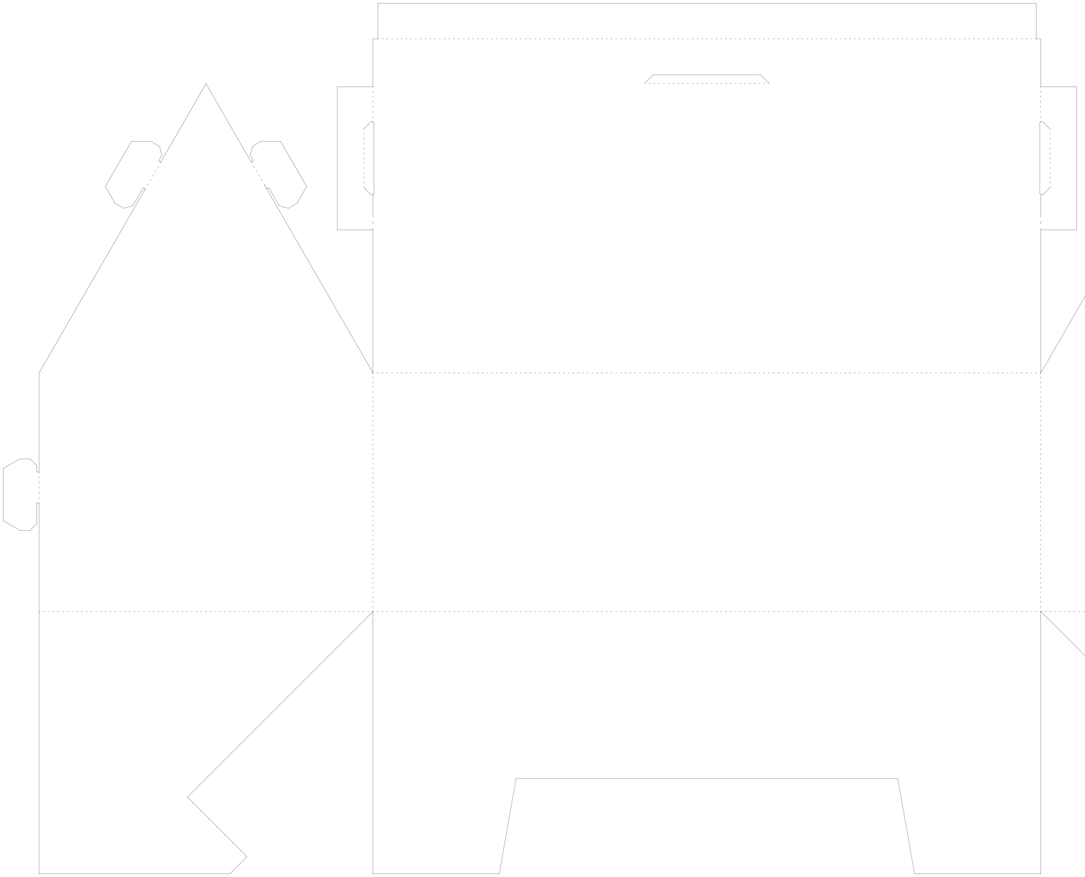
V: 992.874 A:415.359 V/A: 2.390

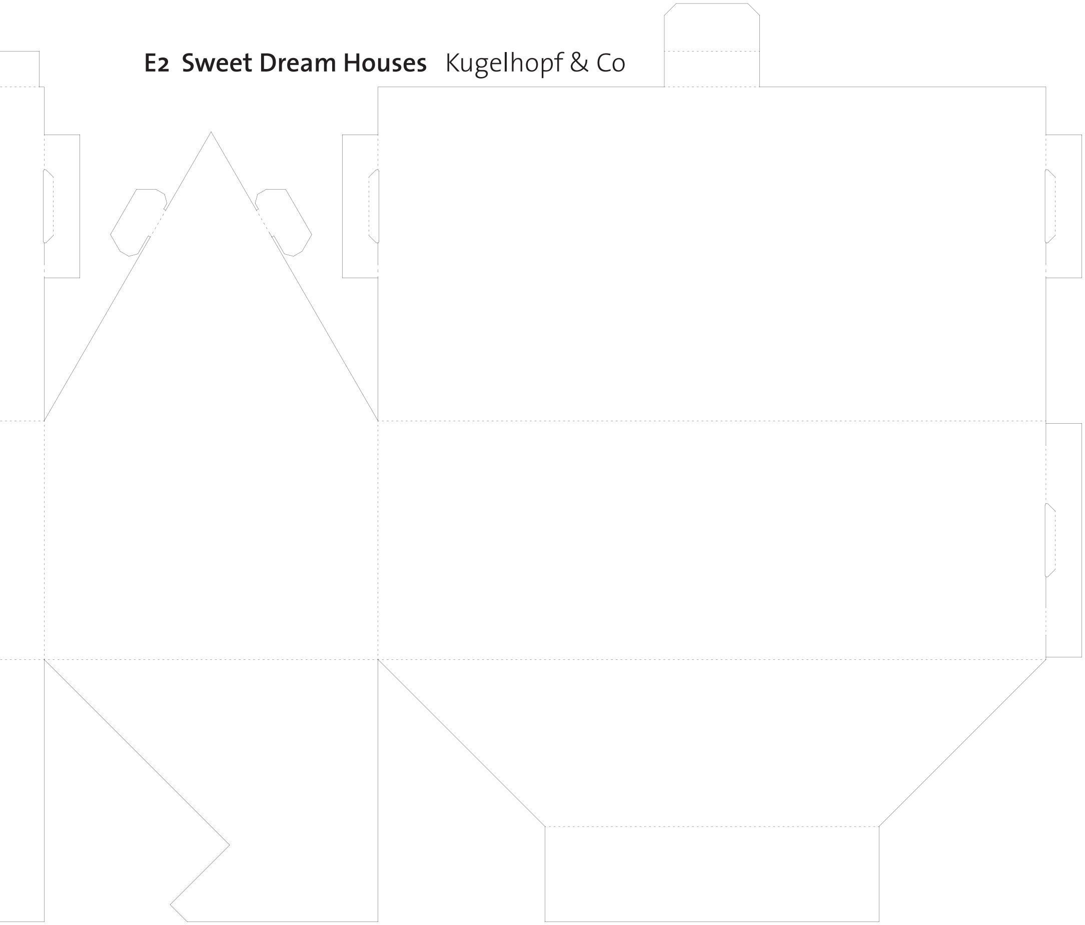
Chapter E
MARKETING

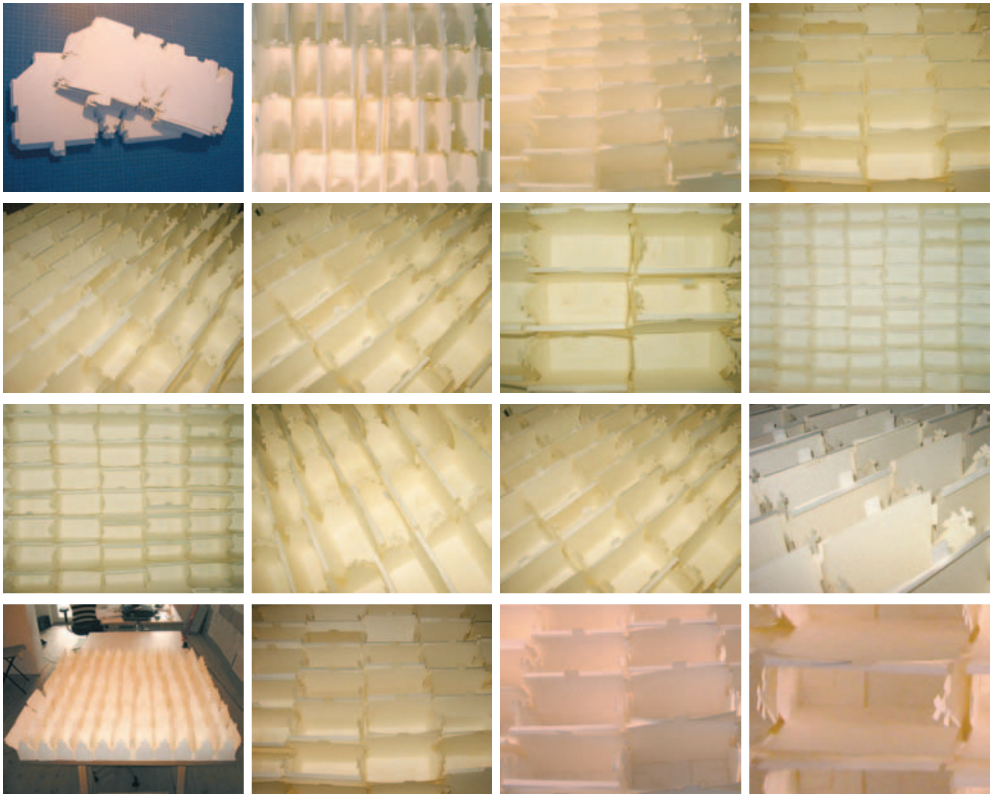
E1 Baking Tins Experiment

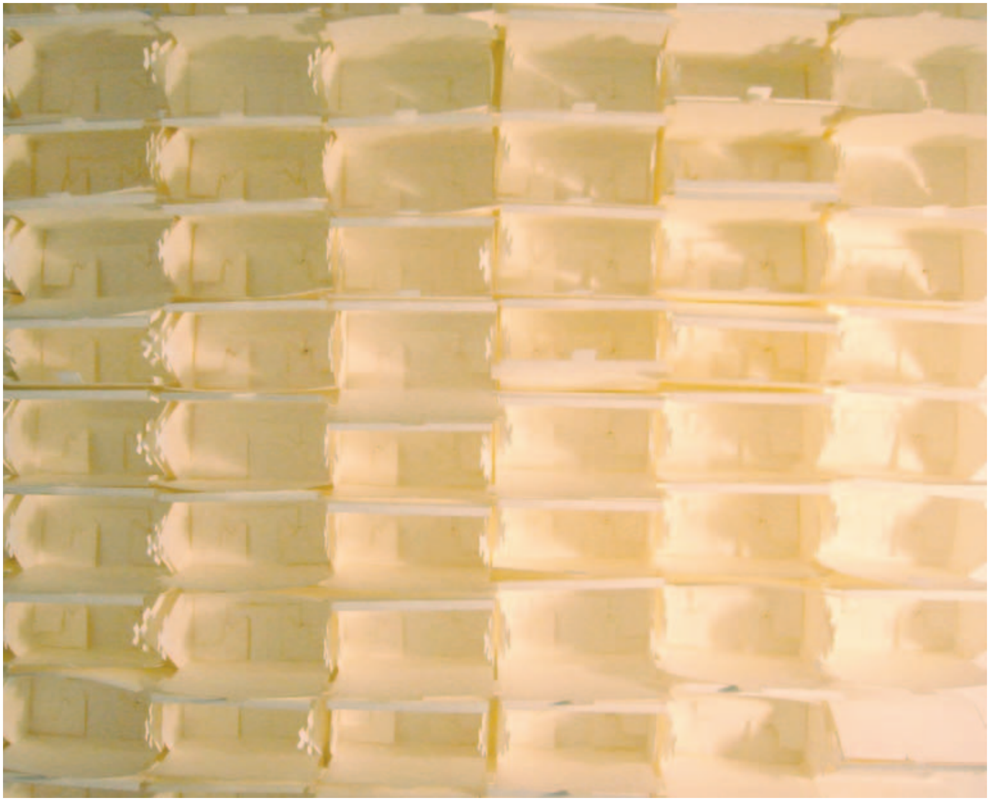



Experiment successful !!!

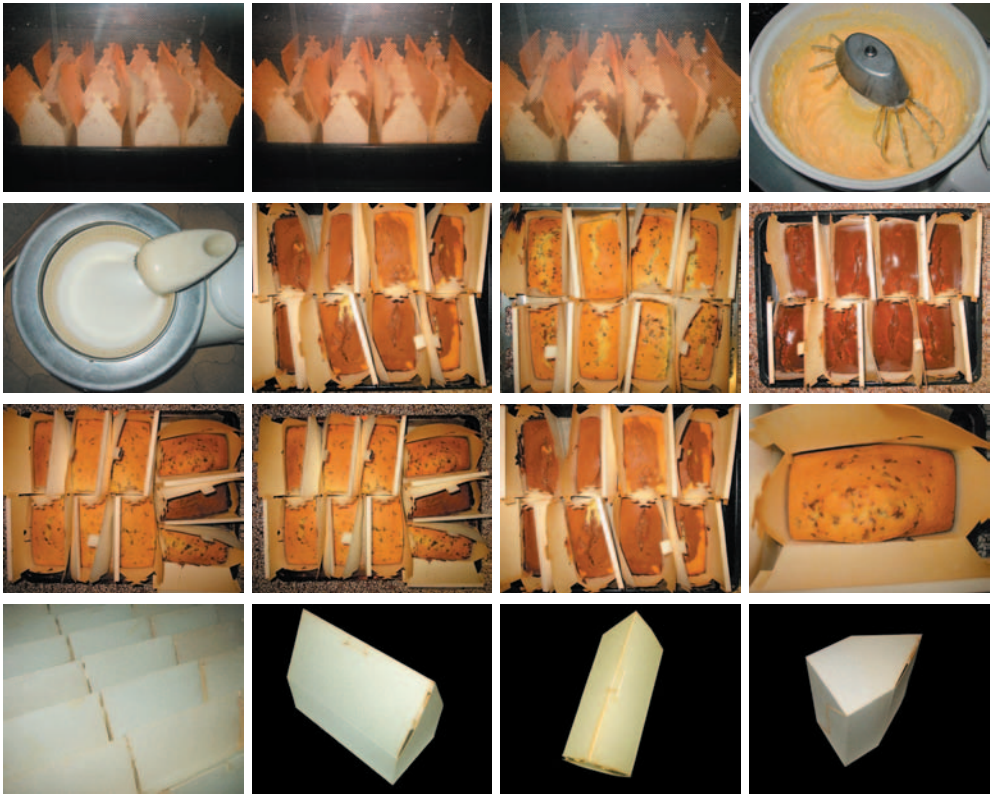








Cardboard Baking Tins



The Result

ACKNOWLEDGEMENT

I would like to thank Philipp Frauenfelder and Arvin Moezzi, for their straightforward mathematical support, as well as Dennis Spreen for his untiring support in programming the HouseMachine.
Thanks also to the »branopac company«, http://www.branopac.de, for sponsoring the cardbord to produce the baking tins, and Gabriele Duerr for baking the Kugelhupfs.

Branopac
PAPIERVEREDLUNG