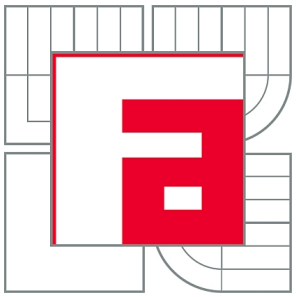
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ARCHITEKTURY
ÚSTAV ZOBRAZOVÁNÍ

FACULTY OF ARCHITECTURE
DEPARTMENT OF DRAWING

NOVÉ DIGITÁLNÍ METODY V PROCESU ARCHITEKTONICKÉHO NAVRHOVÁNÍ

NEW DIGITAL METHODS IN ARCHITECTURE DESIGNING PROCESS

DIZERTAČNÍ PRÁCE
DOCTORAL THESIS

AUTOR PRÁCE — Ing. arch. ADAM SIROTEK
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. arch. ZDENĚK MAKOVSKÝ
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně
Fakulta architektury
Poříčí 273/5, 63900 Brno 39

Zadání dizertační práce

Číslo dizertační práce: Akademický rok: **2011/2012**

Ústav:	Ústav zobrazování
Student(ka):	**Sirotek Adam, Ing. arch.**
Studijní program:	Architektura a urbanismus (P3501)
Studijní obor:	Architektura (3501V002)
Vedoucí dizertační práce:	**doc. Ing. arch. Zdeněk Makovský**
Konzultanti dizertační práce:	

Název dizertační práce:

Nové digitální metody v procesu architektonického navrhování

Zadání dizertační práce:

1. popis a analýza současných digitálních metod
2. vývoj nových metod
3. aplikace v architektonickém navrhování
4. analýza výstupů
5. popis, vývoj a aplikace komplexní geometrie

Rozsah grafických prací:

- analýza digitálních metod používaných současnými architekty
(rozbor z hlediska použitého procesu /algoritmu, geometrie, výroby)
- vývoj nových metod (genetické algoritmy, samoorganizační struktury...)
- testování metod na konkrétním zadání
- analýza - vyhodnocenlí úspěšnosti digitálních metod, porovnání s lidským výstupem
- využití nadzákladní geometrie v architektonickém navrhování

Seznam odborné literatury:

Asymptote: Flux, Couture Luse Anne - Rashid Hani
NOX: Machining Architecture, Lars Spuybroek
The Architecture of Continuity, Lars Spuybroek
Designing for a Digital World, Prof.Neil Leach
Further Architects in Cyberspace, Neil Spiller
Animate Form, Greg Lynn
From Control to Design: Parametric /algorithmic Architecture, Michael Meredith - Aranda lasch - Mutsuro Sasaki

Termín zadání dizertační práce: 25.2.2010

Termín odevzdání dizertační práce:

Dizertační práce se odevzdává v rozsahu stanoveném vedoucím práce; současně se odevzdává 1 výstavní panel formátu B1 a dizertační práce v elektronické podobě.

- -	- -	- -
Sirotek Adam, Ing. arch. Student(ka)	doc. Ing. arch. Zdeněk Makovský Vedoucí práce	doc. Ing. arch. Zdeněk Makovský Vedoucí ústavu

V Brně, dne 25.2.2010

- -
doc. Ing. Josef Chybík, CSc.
Děkan fakulty

Abstrakt

Tématem této dizertační práce jsou digitální metody v procesu architektonického navrhování, jejich analýza vývoj a aplikace.

Tato práce popisuje teoretická východiska užitečná k vytváření nových digitálních metod, systematicky popisuje existující metody a analyzuje jejich použití na různých projektech. V práci je popsán vývoj vlastních metod a ukázka jejich aplikací.

Klíčová slova

Architektura, geometrie, algoritmus, software.

Abstract

The topic of this thesis is digital methods in process of architectonic designing, analysis, research and application of new methods.

This thesis describes theoretical sources, which are useful for creating new digital methods. Thesis systematically describes existing methods and analyzes application on various projects. In thesis is also described research of new, own methods and preview of application.

Key words

Architecture, geometry, algorithm, software

Bibliografická citace

SIROTEK, A. *Nové digitální metody v procesu architektonického navrhování*. Brno: Vysoké učení technické v Brně, Fakulta architektury, 2012. 152 s. Vedoucí práce doc. Ing. arch. Zdeněk Makovský.

Prohlášení

Prohlašuji, že jsem zpracoval tuto disertační práci "Nové digitální metody v procesu architektonického navrhování" samostatně, pod vedením školitele a za použití zdrojů uvedených v seznamu použitých zdrojů.
Prohlašuji, že jsem jako autor této doktorské disertační práce neporušil autorská práva třetích osob (§ 11 Autorského zákona 121/2000Sb.)

V Brně dne 27. Června 2012 ……

Poděkování

Rudolfu Müllerovi a Petru Sovišovi za spolupráci na vývoji konceptu softwaru.
Gregu Lynnovi, Michaelu Weinstockovi za prezentace jejich výzkumu.
Všem organizátorům a autorům Bienále v Benátkách 2004 za prvotní impulz a zdroj inspirace.
Rodině, partnerce a přátelům za podporu psychickou i hmotnou.

Obsah

1 Úvod

1.1 Cíl práce

Cílem dizertační práce bylo popsat a analyzovat současné digitální metody v procesu architektonického navrhování. Dále byl cílem vývoj nových metod a jejich hodnocení, případná aplikace v architektonickém návrhu.

Metody architektonického navrhování by měly být v této práci hodnoceny systematicky dle kriteria množství a způsobu využití výpočetní techniky v jednotlivých fázích návrhu. Příklady v práci popisované by měly být doplněny ilustracemi konkrétních návrhů a staveb. Popisované metody by měly být zdokumentovány jak formou textu, tak ilustrativními schématy, popřípadě částmi programů.

Nově navrhované postupy a metody budou zdokumentovány ilustrativními schématy, částmi zdrojových kódů, popisem a grafickým zdokumentováním výstupů. Výstupy budou doplněny o analýzu z hlediska aplikace v architektonickém navrhování.

1.2 Obsah a struktura práce

Práce je rozdělena do dvou základních částí. V první jsou systematicky rozebírány poznatky z různých oborů, které podstatně ovlivňují architektonické navrhování. Dále jsou popsány jednotlivé metody navrhování, algoritmy a používané struktury dat. Na závěr první části jsou vybrané návrhy a realizace hodnoceny z hlediska digitálního navrhování a rozebrán proces jejich vzniku.

Ve druhé části bude vysvětlen nově navrhovaný komplexní systém pro architektonické navrhování, jehož jedna část bude dokumentována podrobně i s testovacími úlohami.

V kapitole 2 – Východiska Kriteria hodnocení projektů - jsou popisovány a členěny přístupy k navrhování z hlediska:

- geometrie
- konstrukce

- výroby

a popsány a vysvětleny:

- metody v procesu navrhování
- používané algoritmy
- používané datové struktury

Ve třetí kapitole – *Analýza projektů* - jsou vybrána některá díla předních i začínajících architektů, designérů a programátorů, na nichž je možné jednotlivé popsané přístupy vysledovat.

Ve čtvrté kapitole – *Návrh systému* – je rozebírána myšlenka maximálního možného využití výpočetní techniky (digitalizace) v procesu architektonického navrhování. Je zde prezentováno funkční schéma a popis konceptu vyvíjeného programu i základní řešení z hlediska programování.

V páté kapitole – *Návrh CAD Systému* – je rozebírána jedna ze základních částí celého systému. Program je popisován chronologicky tak jak vznikal na základě stanovených požadavků, je popisováno řešení jednotlivých problémů a na závěr je program testován na jednotlivých úlohách.

V závěru je popsán výhled a možnosti budoucího vývoje digitalizace architektury a další možný rozvoj vlastního programu.

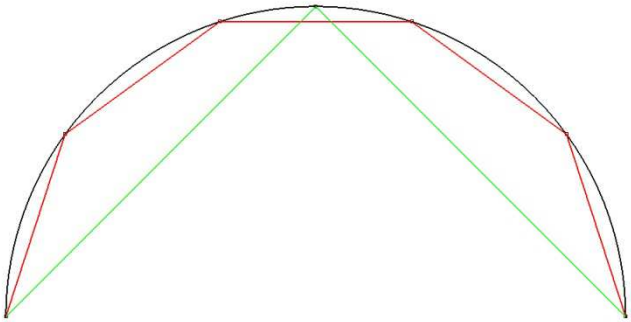
2 Východiska a kriteria hodnocení projektů

2.1 Geometrie

2.1.1 Přímky

Z hlediska jednoduchosti jsou přímky, potažmo úsečky, ideálním geometrickým primitivem pro jakoukoli metodu či proces jak pro člověka, tak pro počítač. Úsečka má oproti ostatním geometrickým pripitivům výrazně jednodušší matematický zápis a je i snadno pochopitelná - čitelná pro člověka. Naprostá většina i těch nejsložitějších algoritmů v počítačové grafice pracuje s úsečkami.

Úsečky však mají jedno zásadní omezení. Velmi obtížně se pomocí nich přesně definují křivky. Křivky sice můžeme interpolovat na lomené čáry (*polyline*), výsledná definice je ale náročnější na množství dat než původní křivka. Navíc je jen málokdy možné získat zpět z lomené čáry křivku identickou s křivkou původní.

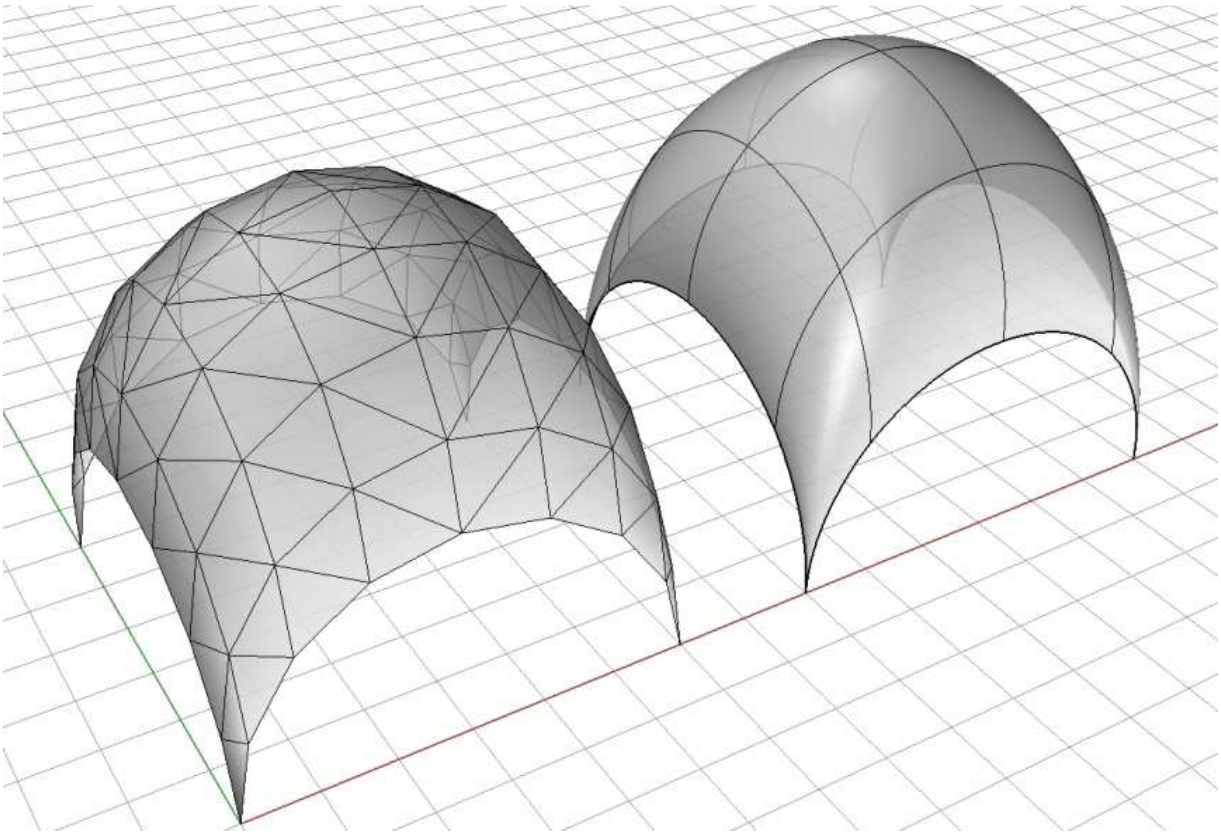


obr. 01 interpolace křivky

Plošné geometrické objekty založené striktně na úsečkách jsou obdobně nejjednodušší pro práci s nimi, jejich výčet je ale velmi malý a možnosti omezené. Základním stavením kamenem v počítačové grafice je buď trojúhelník, který má striktně rovinnou formu, nebo čtyřúhelníková přímková plocha (face). Jednotlivé softwary a algoritmy jsou založeny buď na trojúhelnících (např. 3D studio, AutoCAD), nebo na čtyřúhelnících (např. Rhinoceros). Pakliže je základem trojúhelník, pak čtverec je definován jako dva trojúhelníky. Je-li základem čtyřúhelník, pak trojúhelník je čtyřúhelníkem s jednou stranou nulovou – potlačenou. Tyto přístupy lze jen

obtížně kombinovat, přesto je v posledních letech viditelná snaha programátorů používat přístupy oba paralelně v rámci jednoho softwaru.

Podobně jako lze interpolovat křivku na lomenou čáru, lze převést zakřivenou plochu na polygonální síť (*polygon mesh, mesh*) – Práce s mnoha trojúhelníky je matematicky výrazně jednodušší než s křivou plochou. Problém nastává, když po nich požadujeme, aby se chovaly stejně jako původní zakřivená plocha.

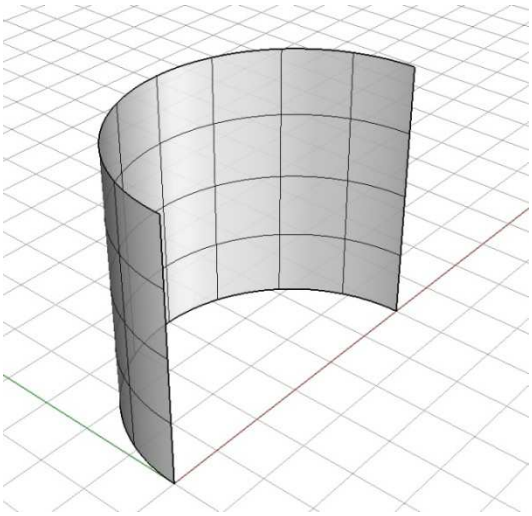


obr. 02 interpolace zakřivené plochy

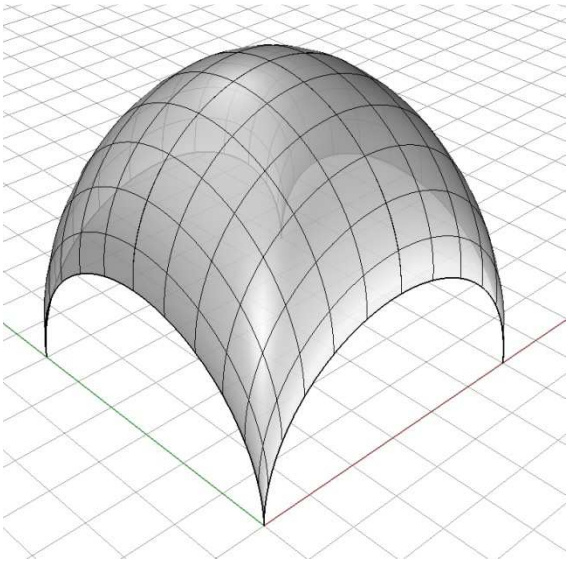
2.1.2 Kuželosečky

Dalšími jednoduchými geometrickými primitivy jsou kuželosečky. Z nich je však výrazněji používána pouze kružnice (*circle*), potažmo kruhový oblouk (*arc*), méně se objevují elipsy a jejich části, paraboly a hyperboly jsou výjimečné.

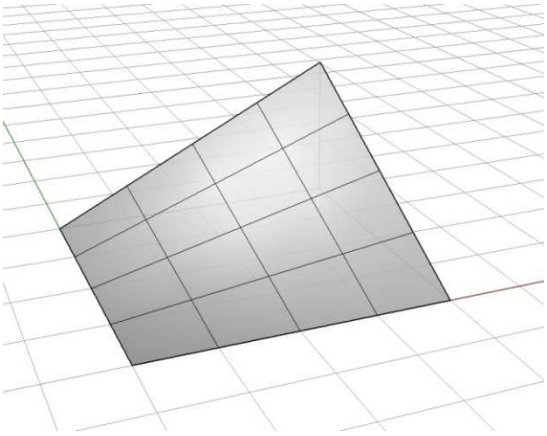
Jejich převedení do prostoru je možné buď kombinací s přímkami, nebo kombinací mezi sebou. Přímky jsou křivky prvního stupně, kuželosečky 2°. Hlavní – řídící směry ploch bývají nazývány "*u,v*".



Obr. 03 Zakřivená plocha (u2°v1°)



Obr. 04 Zakřivená plocha (u2°v2°)



Obr. 05 Zakřivená plocha (u1°v1°) – parabolický hyperboloid

Existuje mnoho dalších kombinací i mnoho dalších křivek, práce s nimi je obdobná a není cílem této práce systematicky třídit geometrická a stereometrická primitiva. Odkázal bych se na dizertační práci Jana Foretníka: Architektura, geometrie a výpočetní technika, FA VUT v Brně, 2010.

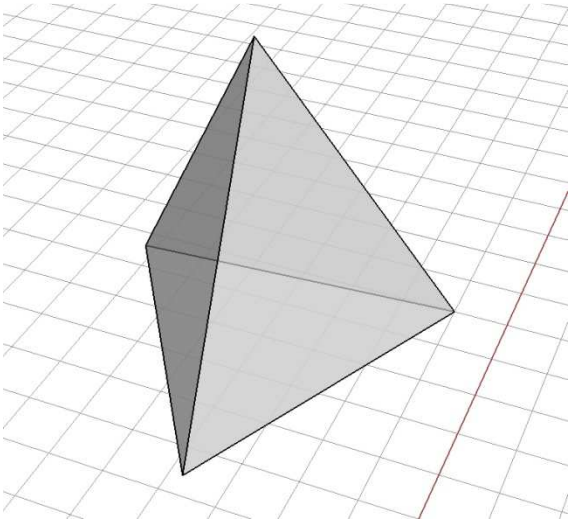
2.1.3 Prostorové objekty

Prostorové objekty lze dělit vedle klasické typologie i dle jejich prostorové interpretace v grafických softwarech. Základní euklidovská tělesa (čtyřstěn, krychle, osmistěn, dvanácti a dvacetistěn) jsou stejně jako všechna tělesa s rovnými plochami definována jako uzavřená soustava povrchových ploch, hran, vrcholů, jejich vlastností a vazeb v předdefinované struktuře nahrazující těleso. Informace o vlastnostech tělesa samotného získáváme výpočtem z jejich vzájemných vztahů.

Příklad: *zkrácená definice čtyřstěnu v programu Rhinoceros*

```
Rhino object: polysurface
  name: ""
  id: DEBE651C-94DD-49f3-B4F1-604F1E369D28
  layer index: 0
  render material index: -1 (from layer)
  ON_Brep:
  surfaces:  4
  3d curve:  6
  2d curves: 12
  vertices:  4
  edges:     6
  trims:     12
  loops:     4
  faces:     4
  curve2d[ 0]: TL_NurbsCurve domain(0,6.245) start(-3.1225,-2.08833) end(3.1225,-2.08833)
  curve2d[ 1]: TL_NurbsCurve domain(0,7) start(3.1225,-2.08833) end(6.66134e-16,4.17665)
---
  curve2d[11]: TL_NurbsCurve domain(0,6.245) start(-1.80278,-3.1225) end(3.60555,0)
  curve3d[ 0]: ON_LineCurve domain(0,6.245) start(8,1,0) end(7.59808,7.23205,0)
  curve3d[ 1]: ON_LineCurve domain(0,7) start(7.59808,7.23205,0) end(6,4,6)
---
  curve3d[ 5]: ON_LineCurve domain(0,6.245) start(2.40192,3.76795,0) end(8,1,0)
  surface[ 0]: ON_PlaneSurface u(-3.1225,3.1225) v(-2.08833,4.17665)
  surface[ 1]: ON_PlaneSurface u(-3.1225,3.1225) v(-2.08833,4.17665)
  surface[ 2]: ON_PlaneSurface u(-3.1225,3.1225) v(-2.08833,4.17665)
  surface[ 3]: ON_PlaneSurface u(-1.80278,3.60555) v(-3.1225,3.1225)
  vertex[ 0]: (8.000000 1.000000 0.000000) tolerance(0)
    edges (0,2,5)
  vertex[ 1]: (7.598076 7.232051 0.000000) tolerance(0)
    edges (0,1,3)
  vertex[ 2]: (6.000000 4.000000 6.000000) tolerance(0)
    edges (1,2,4)
  vertex[ 3]: (2.401924 3.767949 0.000000) tolerance(0)
    edges (3,4,5)
  edge[ 0]: v0( 0) v1( 1) 3d_curve(0) tolerance(0)
    domain(0,6.245) start(8,1,0) end(7.59808,7.23205,0)
    trims (+0,+9)
  edge[ 1]: v0( 1) v1( 2) 3d_curve(1) tolerance(0)
    domain(0,7) start(7.59808,7.23205,0) end(6,4,6)
    trims (+1,-5)
---
  edge[ 5]: v0( 3) v1( 0) 3d_curve(5) tolerance(0)
    domain(0,6.245) start(2.40192,3.76795,0) end(8,1,0)
    trims (+6,+11)
  face[ 0]: surface(0) reverse(0) loops(0)
    Fast render mesh: 1 polygons
    loop[ 0]: type(outer) 3 trims(0,1,2)
      trim[ 0]: edge( 0) v0( 0) v1( 1) tolerance(0,0)
        type(mated   -south side iso) rev3d(0) 2d_curve(0)
        domain(0,6.245) start(-3.1225,-2.08833) end(3.1225,-2.08833)
```

```
        surface points start(8,1,0) end(7.59808,7.23205,-4.44089e-16)
      trim[ 1]: edge( 1) v0( 1) v1( 2) tolerance(0,0)
        type(mated   ) rev3d(0) 2d_curve(1)
        domain(0,7) start(3.1225,-2.08833) end(6.66134e-16,4.17665)
        surface points start(7.59808,7.23205,-4.44089e-16) end(6,4,6)
      trim[ 2]: edge( 2) v0( 2) v1( 0) tolerance(0,0)
        type(mated   ) rev3d(0) 2d_curve(2)
        domain(0,7) start(6.66134e-16,4.17665) end(-3.1225,-2.08833)
        surface points start(6,4,6) end(8,1,0)
---
  face[ 3]: surface(3) reverse(1) loops(3)
    Fast render mesh: 1 polygons
    loop[ 3]: type(outer) 3 trims(9,10,11)
      trim[ 9]: edge( 0) v0( 0) v1( 1) tolerance(0,0)
        type(mated   ) rev3d(0) 2d_curve(9)
        domain(0,6.245) start(3.60555,0) end(-1.80278,3.1225)
        surface points start(8,1,0) end(7.59808,7.23205,0)
      trim[10]: edge( 3) v0( 1) v1( 3) tolerance(0,0)
        type(mated   -west side iso) rev3d(0) 2d_curve(10)
        domain(0,6.245) start(-1.80278,3.1225) end(-1.80278,-3.1225)
        surface points start(7.59808,7.23205,0) end(2.40192,3.76795,0)
      trim[11]: edge( 5) v0( 3) v1( 0) tolerance(0,0)
        type(mated   ) rev3d(0) 2d_curve(11)
        domain(0,6.245) start(-1.80278,-3.1225) end(3.60555,0)
        surface points start(2.40192,3.76795,0) end(8,1,0)
```



obr. 06 čtyřstěn v programu Rhinoceros

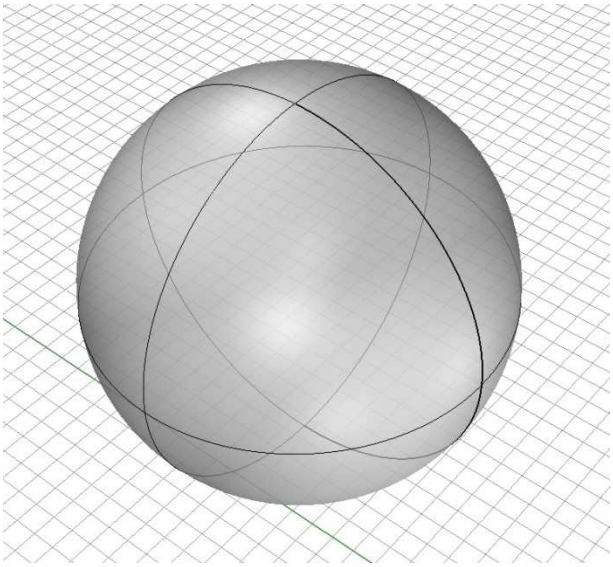
Podobným způsobem jsou definovány i prostorové objekty zakřivené:

Příklad: definice koule v programu Rhinoceros

Kouli tvoří jedna uzavřená rotační plocha – rotace oblouku 180° ,360°

```
Rhino object: surface
  id: 4D022583-20C6-4132-9F20-7F85F461DEE0
  layer index: 0
  render material index: -1 (from layer)
  ON_Brep:
  (B-rep geometry is the same as underlying surface.)
  surfaces:  1
  3d curve:  1
  2d curves: 4
  vertices:  2
  edges:     1
  trims:     4
  loops:     1
  faces:     1
  curve2d[ 0]: TL_NurbsCurve domain(0,62.8319) start(0,-15.708) end(62.8319,-15.708)
  curve2d[ 1]: TL_NurbsCurve domain(-15.708,15.708) start(62.8319,-15.708) end(62.8319,15.708)
  curve2d[ 2]: TL_NurbsCurve domain(0,62.8319) start(62.8319,15.708) end(0,15.708)
  curve2d[ 3]: TL_NurbsCurve domain(-15.708,15.708) start(0,15.708) end(0,-15.708)
  curve3d[ 0]: ON_ArcCurve domain(-15.708,15.708) start(10,30,-10) end(10,30,10)
  surface[ 0]: TL_RevSurface u(0,62.8319) v(-15.708,15.708)
    surface details:
      class name: TL_RevSurface
      class uuid: 0A8401B6-4D34-4b99-8615-1B4E723DC4E5
        Paramerization: (angle,curve)
```

```
    Axis: (10, 30, 0) to (10, 30, 1)
    Rotation angle: 0 to 6.28319 radians.
    Angle evaluation parameter interval: [0,62.8319].
    Revolute:
      ON_ArcCurve:  domain = [-15.708,15.708]
        center = (10, 30, 0)
        radius = 10
        length = 31.4159
vertex[ 0]: (10.000000 30.000000 -10.000000) tolerance(0)
  edges (0)
vertex[ 1]: (10.000000 30.000000 10.000000) tolerance(0)
  edges (0)
edge[ 0]: v0( 0) v1( 1) 3d_curve(0) tolerance(0)
  domain(-15.708,15.708) start(10,30,-10) end(10,30,10)
  trims (+1,-3)
face[ 0]: surface(0) reverse(0) loops(0)
  Fast render mesh: 10352 polygons
  (Face geometry is the same as underlying surface.)
  loop[ 0]: type(outer) 4 trims(0,1,2,3)
    trim[ 0]: edge(-1) v0( 0) v1( 0) tolerance(0,0)
      type(singular-south side iso) rev3d(0) 2d_curve(0)
      domain(0,62.8319) start(0,-15.708) end(62.8319,-15.708)
      surface points start(10,30,-10) end(10,30,-10)
    trim[ 1]: edge( 0) v0( 0) v1( 1) tolerance(0,0)
      type(seam    -east side iso) rev3d(0) 2d_curve(1)
      domain(-15.708,15.708) start(62.8319,-15.708) end(62.8319,15.708)
      surface points start(10,30,-10) end(10,30,10)
    trim[ 2]: edge(-1) v0( 1) v1( 1) tolerance(0,0)
      type(singular-north side iso) rev3d(0) 2d_curve(2)
      domain(0,62.8319) start(62.8319,15.708) end(0,15.708)
      surface points start(10,30,10) end(10,30,10)
    trim[ 3]: edge( 0) v0( 1) v1( 0) tolerance(0,0)
      type(seam    -west side iso) rev3d(1) 2d_curve(3)
      domain(-15.708,15.708) start(0,15.708) end(0,-15.708)
      surface points start(10,30,10) end(10,30,-10)
```



obr. 07 koule v programu Rhinoceros

Druhým způsobem definování prostorových objektů je voxelová interpolace matematické definice. Objekt je vyjádřen rovnicí (povrch) nebo nerovnicí (objem).

(např. koule je definována středem a poloměrem)

p – bod uvnitř koule, s – střed, r - poloměr

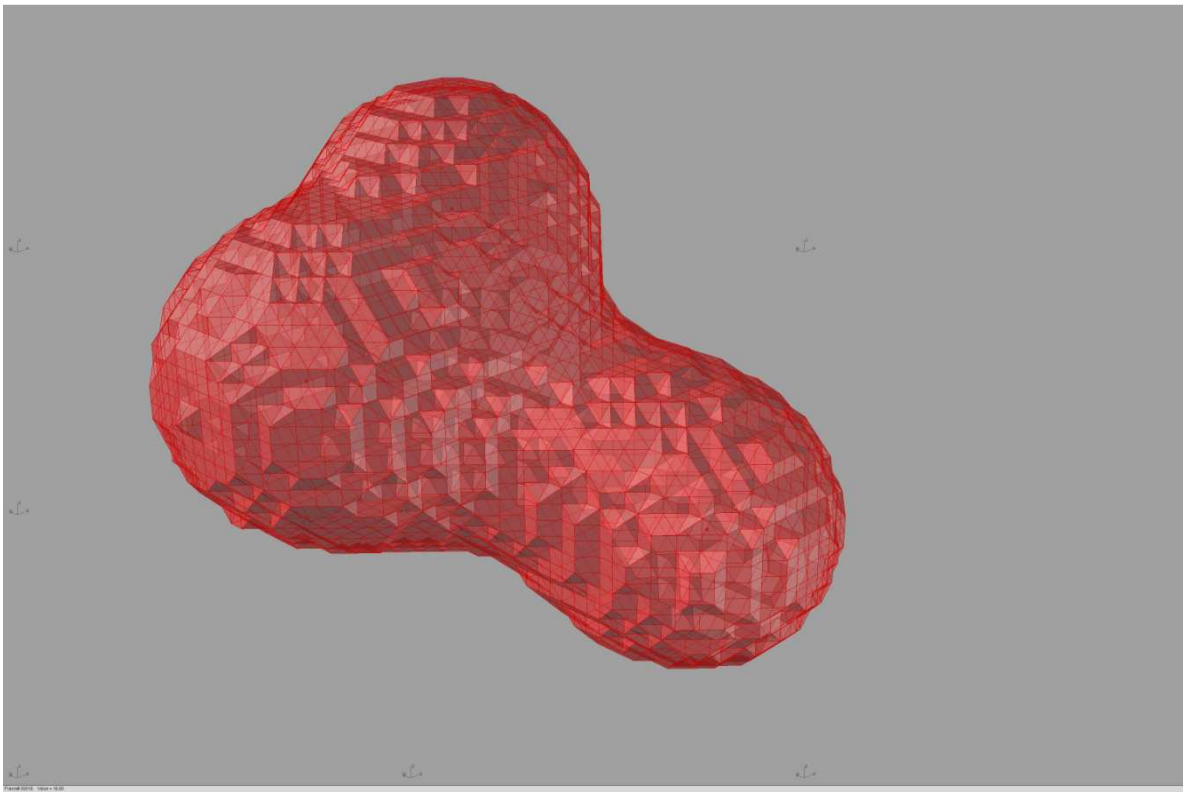
pro libovolný bod uvnitř koule platí: p: |p,s| < r

Pro zobrazení definujeme prostor jako trojrozměrnou síť. Řešený prostor vyplníme kvádry (voxel) o malé hraně (délka hrany určuje přesnost výstupu, ale i délku výpočtu). Následně dle vztahu vybereme či vyloučíme voxely.

Touto metodou umí pracovat jen minimum programů. Důvodem pro využití může být potřeba práce s vnitřními silami v objektu (statické programy) nebo nemožnost zobrazit požadované objekty jednodušší metodou. Příkladem takového objektu jsou izomorfní objekty, tzv. metabally .

Příklad: mataball (Rhinoceros – Grasshopper – VBScript)

Voxely byly pro lepší znázornění výstupu rozřezány a stejným algoritmem vybrány

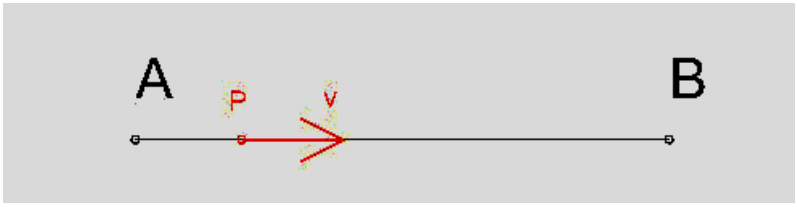


obr. 08 mataball – voxelová interpolace

2.1.4 Parametrické lineární objekty

Vývojáři počítačové grafiky postupně reagovali na sílící požadavky uživatelů na práci s volnými křivkami. První křivkou implementovanou do grafických programů byla spline. Princip spočívá ve využití infinitezimální geometrie. Pro pochopení je nejsnazší si představit křivku jako stopu pohybujícího se bodu.

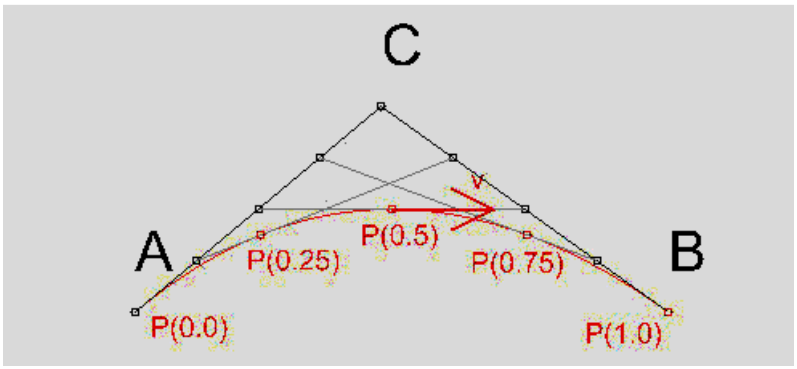
Bude-li se bod pohybovat z bodu *A* do bodu *B* po vektoru *v=(B-A)*, jeho stopou bude úsečka *AB*



obr. 09 spline 1°

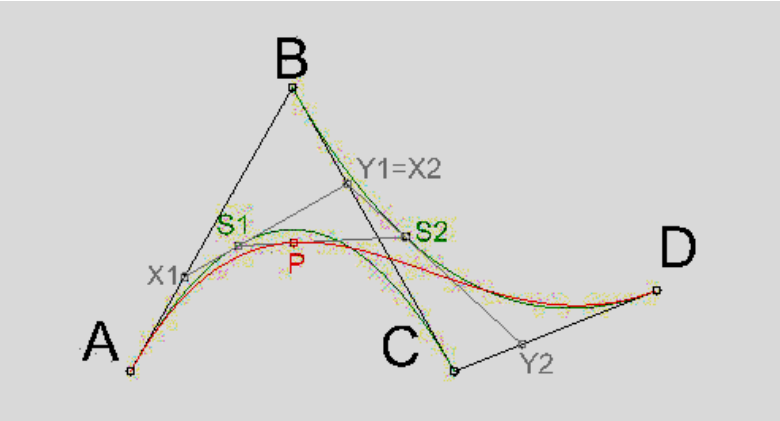
Stanovíme li trasu třemi body *(A,B,C)*, bude se bod pohybovat na trase z *A(t=0)* okolo *B(t=0,5)* do *C(t=1)*. Směr pohybu je kombinací vektorů *v1=(B-A)* a *v2=(C-B)*. V čase *t=0* bude směr pohybu z bodu *A* k bodu *B* *v=v1*1+v2*0*, v průběhu bude vektor *v1* slábnout a vektor *v2* sílit *v=v1*(1-t)+v2*(t)* až na konci *(t=1)* v bodě *C* bude působit jen vektor *v2=v1*0+v2*1*. Takto vzniká kvadratická spline, mj. parabola.

Geometricky to lze vyjádřit tak že se bod *P* nachází na pohybující se spojnici *XY* mezi úsečkami *AB* a *BC*, na začátku je *X* blíže k *A*, *Y* k blíže k *C* a *P* blíže k *X*, na konci je *X* blíže k *B*, *Y* k *C* a *P* k *Y.*



obr. 10 spline 2°

Dalším postupným krokem je přidání čtvrtého bodu. Vytvoříme dvě křivky spline 2° s1(A,B,C) a s2(B,C,D). Dále postupujeme podobně jako v předchozím případě, jen se pohybujeme na spojnici mezi křivkami s1 a s2. Výsledná rovnice má třetí stupeň.



obr. 11 spline 3°

Dále lze přidávat libovolný počet bodů a zvyšovat tak stupeň křivky s každým dalším bodem. Úprava rovnice Cox-deBoorovým algoritmem umožňuje zůstat u 3°. To má za následek diskontinuitu C3 u křivek s více jak čtyřmi řídícími body, což ale běžné smrtelníky trápit rozhodně nebude.

Následný parametrický zápis kubické B-spline:

$$S_i(x) = \frac{z_i(x-t_{i-1})^3}{6h_i} + \frac{z_{i-1}(t_i-x)^3}{6h_i} + \left[\frac{f(t_i)}{h_i} - \frac{z_i h_i}{6}\right](x-t_{i-1}) + \left[\frac{f(t_{i-1})}{h_i} - \frac{z_{i-1}h_i}{6}\right](t_i-x)$$

Další úprava umožnila stanovit sílu působení jednotlivých bodů přidáním proměnné (nyní je křivka definována parametry: *x,y,z,w*). Při maximální síle *w* se křivka řídících bodů téměř dotýká a připomíná lomenou čáru, při minimální síle *w* prochází přímo od prvního bodu k poslednímu. Sílu bodů *w* lze měnit nezávisle. To způsobilo i změnu názvu křivky na Non-uniform rational B-spline, zkráceně NURBS. Význam této změny tkví v možnosti definovat velmi přesně libovolnou požadovanou křivku pomocí minimálního počtu bodů a s jednotnou definicí. Příkladem, bez této úpravy prakticky neřešitelným, byl šikmý řez zakřivené NURBS plochy.

Obecný zápis NURBS:

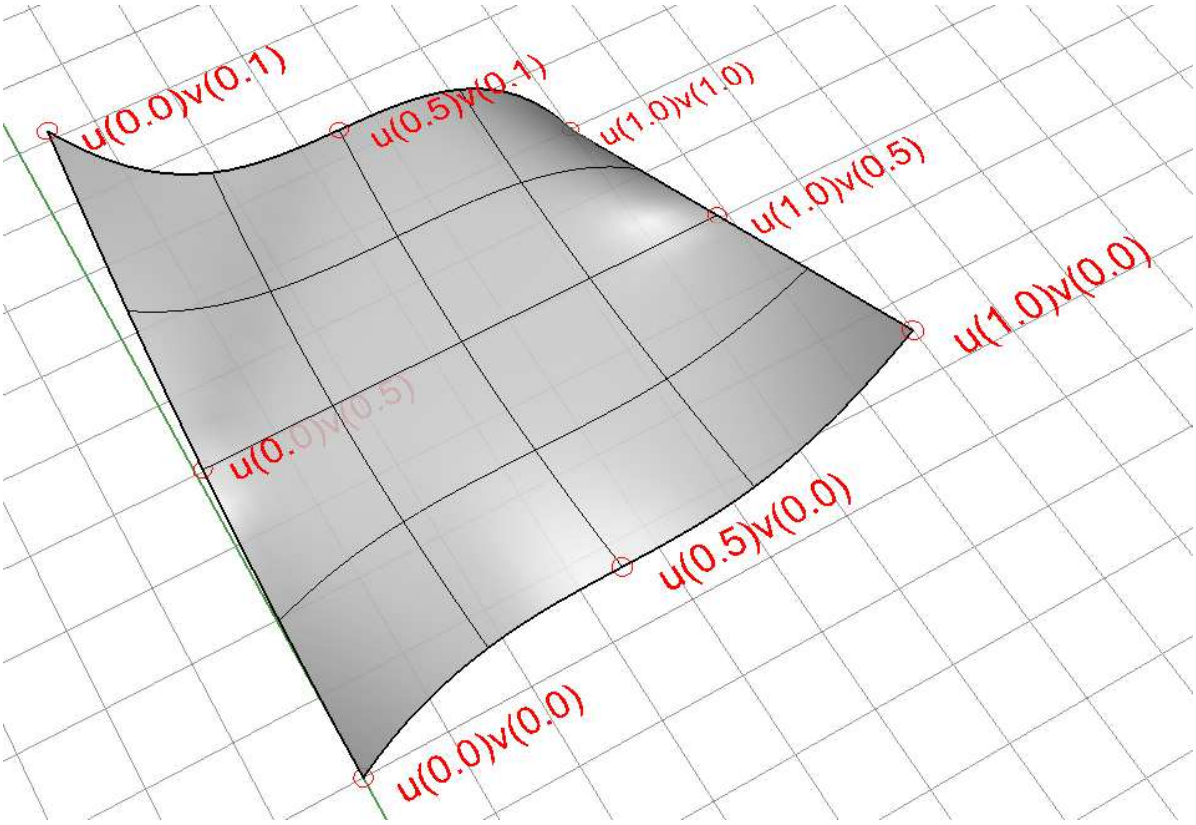
$$C(u) = \sum_{i=1}^{k} \frac{N_{i,n} w_i}{\sum_{j=1}^{k} N_{j,n} w_j} \mathbf{P}_i$$

Křivka NURBS se stala v poslední době standardem počítačové grafiky.

Detailní rozbor vzniku a funkce NURBS vysvětluje lépe Philip Schneider z Drexelovy univerzity. (https://www.cs.drexel.edu/~david/Classes/CS430/Lectures/L-09_BSplines_NURBS.pdf).

2.1.5 Parametrické plošné objekty

Odhlédnuto od dalších a historických možností, plochy v prostoru lze definovat rozvedením NURBS do plochy. Všechny používané NURBS plochy mají základní tvar čtvercový, mají vlastní souřadný systém (*u,v*), jenž je podsystémem systému (*x,y,z*). *u* a *v* nahrazují proměnnou *t* v křivce jednorozměrné, každá v jednom řídícím směru pohybu. Výsledkem je tedy čtverec na zakřivené ploše.



obr. 12 plocha NURBS

2.1.6 Parametrický prostorový objekt

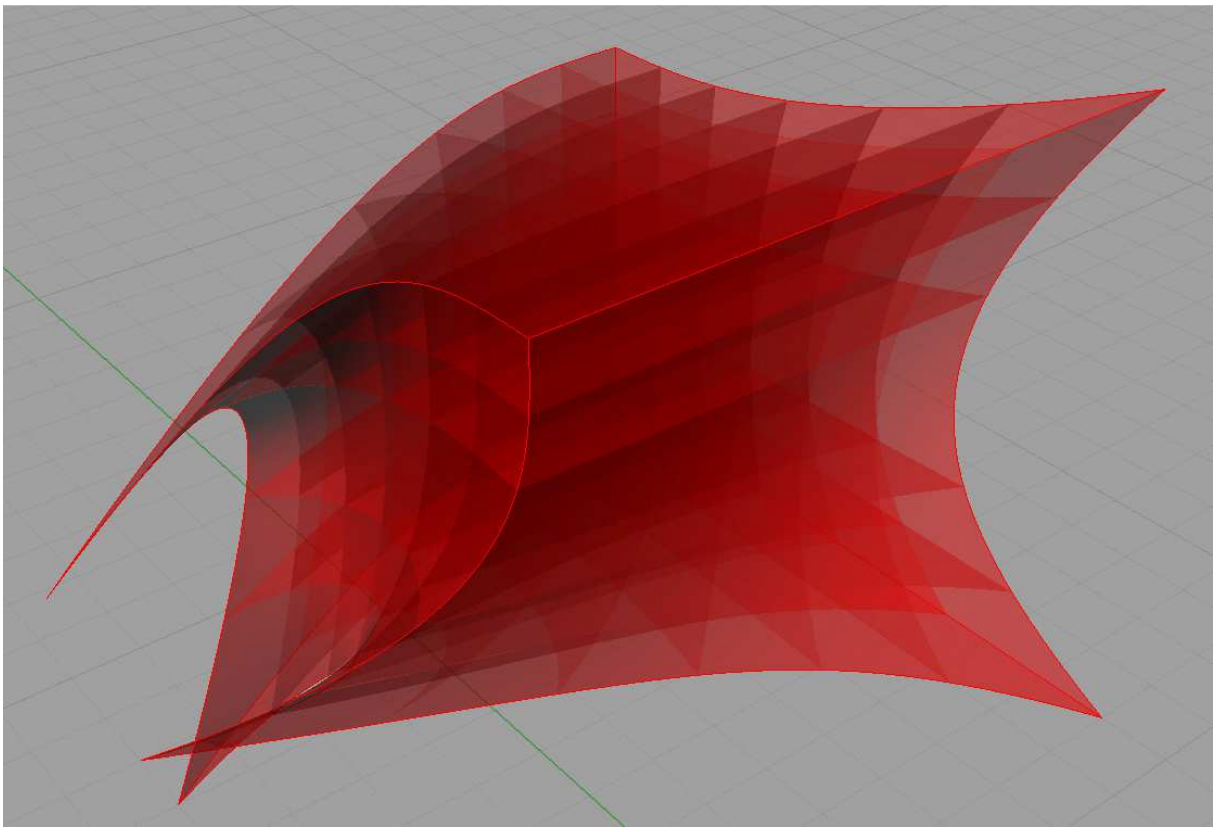
Předcházející dva příklady ukazovaly chování jednorozměrného objektu (křivky) v prostoru a dvourozměrného objektu (plochy) v prostoru. Logickým

postupem je tedy definice třírozměrného parametrického objektu v třírozměrném prostoru.

Podobně jako u plochy byly dvě časové osy řídících směrů nazvány lokálním souřadným systémem, u třírozměrného objektu můžeme definovat souřadný systém (x',y',z'), který je podsystémem (x,y,z). Vzniká tak plný – volumetrický parametrický objekt.

Tato varianta je matematicky možná, vzhledem k nesmírné náročnosti na představivost není současnými grafickými programy raději nabízena. Objekt je definován jako krychle (nebo jiné těleso) v zakřiveném prostoru. Podobně jako křivka může sama sebe protnout (loop) a v jednom bodě mít dva stavy, může se takto protínat i prostorový objekt a existovat tak v jednom bodě ve více stavech.

Praktická aplikace těchto matematických teorií je např. znázornění fyzikálních modelů obecné teorie relativity. V architektuře může být toto východisko velmi užitečné.



obr. 13 parametrický zakřivený prostor

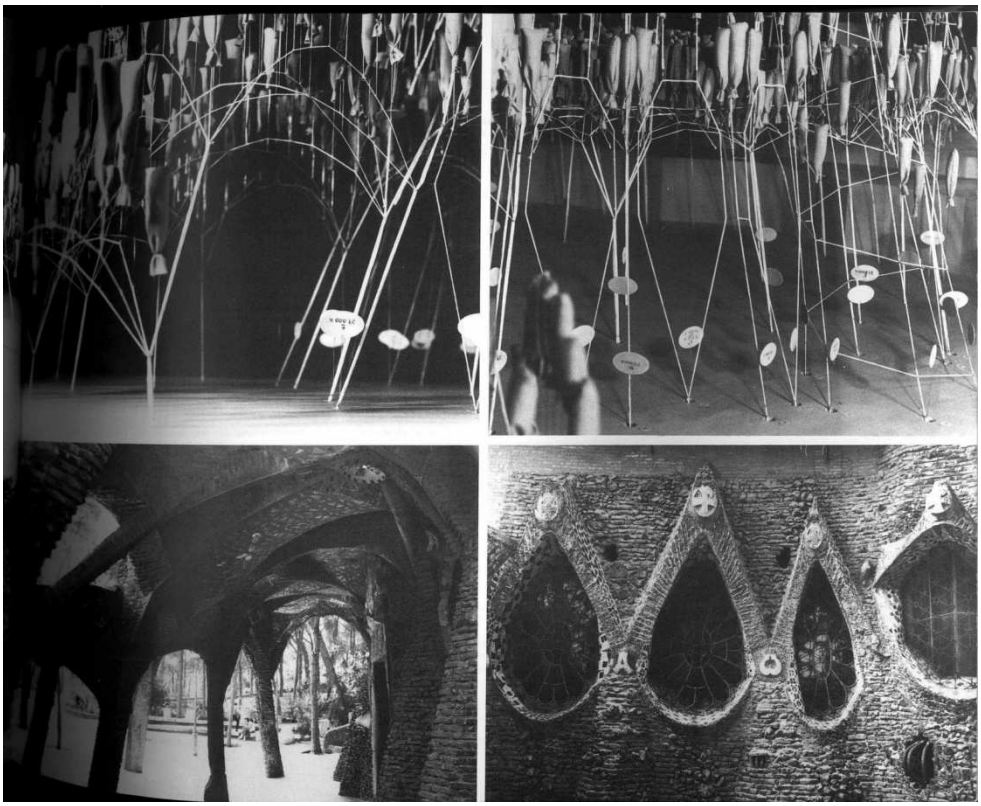
2.2 Konstrukce

Existuje mnoho kriterií jak hodnotit projekty z hlediska konstrukce. Já jsem pro tuto práci vybral dva pohledy, které jsou důležité i z hlediska digitalizace.

První kriterium je jednoduché. Hodnotí, zda je statické řešení v projektu obsaženo, a pakliže ano, pak v jaké míře je digitalizováno.

Druhé kriterium je o něco složitější. Hodnotí, zda jsou statické výpočty využity jen pro dimenzování konstrukcí, případně jejich úpravu (klasická varianta), nebo jsou již součástí primárního konceptu a přímo od začátku definují tvar a formu objektu.

Historickým příkladem takového konceptu jsou Gaudího projekty v Barceloně, kde pomocí tížných řetězovek ručně (!) vytvořil modely následně realizovaných staveb.



obr. 14 Gaudí

2.3 Import, export dat, výroba

2.3.1 Import dat

Proces architektonického navrhování je výrazně ovlivněn technologickou úrovní společnosti, je přímo závislý na typu a kvalitě vstupních dat. Tato práce se zabývá digitálními metodami, pro které je nutné buď vstupní digitální data získat v dále použitelné formě, nebo data upravit tak, aby s nimi bylo možné zvolenou metodou pracovat. V případě, že vstupní data v digitální formě neexistují, pak je nutné je automaticky vygenerovat nebo zdlouhavě manuálně zadat.

Jako příklad vstupních dat pro architektonické navrhování lze uvést informaci o výškovém uspořádání pozemku a okolí. Digitální data je možné získat buď přímo ze zaměření současnými přístroji, nebo ze starších map v papírově formě. Mapy jsou naskenovány, digitalizovány a nakonec je vygenerován trojrozměrný digitální terénní model (DTM).

Obvyklým typem vstupních dat v různých metodách jsou informace z různých receptorů sledujících vlivy z okolí - např. akustické či světelné snímače.

Zvláštním případem jsou snímače pohybu. Lze jimi sledovat pohyby lidí, či objektů v prostoru a jejich interakce v čase. Na snímání pohybu je založená např. metoda Motion Capture (dále viz kapitola 2.4).

Při hodnocení lze sledovat typ vstupních dat, s jakým jednotlivé metody pracují a jak je získávají.

2.3.2 Export dat

Forma výstupních dat je závislá na tom, s jakými daty je v procesu navrhování zacházeno, na digitálním prostředí zvolené metody a na tom, k čemu je která metoda určena.

Ve většině případů jsou výstupní data ve formě textových a číselných dat a dvou nebo třírozměrné výkresové dokumentace s různou vypovídací hodnotou. V některých případech jsou data zvenčí zcela nesrozumitelná a slouží jako vstupní data pro další zpracování jinou metodou.

Finální úprava je závislá na svém účelu. Pro prezentaci jsou většinou požívány 2D výkresy a 3D modely, vizualizace. Pro účely stavby je forma dat závislá na metodě výroby, data jsou buď ve formě stavebních či výrobních výkresů, nebo jsou též digitální data uzpůsobena přímo pro výrobní systémy.

2.3.3 Výroba

Vývoj digitálních technologií se projevil v oblasti stavebnictví, ale hlavně a velmi zásadně v oblasti strojírenství, což zpětně otevřelo celou škálu nových možností v architektuře.

Zásadním krokem je masové použití počítačově řízených obráběcích (computer numerical control - CNC) strojů a digitálně řízené výroby (computer aided manufacturing – CAM). Výroba probíhá automaticky na základě vstupních dat, data jsou zpracovávána stejně při každém výrobku a obráběcí stroje jsou navrženy tak, aby uměly vyrobit velmi různé výrobky. Proces je stejný, vyrábíme li jeden výrobek opakovaně nebo každý výrobek trochu jiný. Tímto se ztrácí ekonomická výhodnost modularity, normalizace. Tento technický pokrok uvolnil tvarová omezení ve stavebnictví a potažmo v architektuře. Vzniká tak možnost a tedy i požadavek na vývoj nových architektonických forem a metod jak s nimi zacházet.

Dalším zlomem je vznik obráběcích, výrobních strojů, které na základě digitálních dat vytvoří velmi přesný výrobek libovolných nových tvarů. Jako příklad bych uvedl jeden výrobní proces vybraný z těch zajímavých pro architekturu. Je jím výroba skel zakřivených ve více směrech. Forma, ve které se skleněná plocha tvaruje a chladne, je automaticky tvarovatelná na základě vstupních digitálních dat. Forma je opakovatelně použitelná a její tvar prakticky okamžitě upravitelný.

Dalším velmi cenným vynálezem je 3D tiskárna (rapid prototyping). Tyto tiskárny jsou zatím ve stádiu zrodu, vyrábějí trojrozměrné objekty na základě digitálních 3D modelů. Jejich výhodou je schopnost vyrobit prakticky jakýkoli geometrický tvar ve velmi krátkém čase. Nevýhod je zatím mnoho, tiskárny jsou omezeny rozměrově, materiál výrobku je dán striktně technologií tiskárny (většina tiskáren je založena na tepelně aktivované polymeraci polymerového prášku), cena tisku je zatím vysoká. Využití zatím nemůžeme hledat ve výrobě pro produkci, ale velkou hodnotu má při výrobě modelů, tvarových prototypů či forem pro výrobu.

Posledním bodem je možnost digitálního řízení stavby a manipulace s objekty přímo na stavbě. Koordinace dopravy tisíců jednotlivých prvků (vzhledem k výše popsaným možnostem výroby může být každý prvek unikátní) ve správném čase přesně na správné místo vyžaduje použití logistických systémů vyvinutých pro skladové haly. Výrobky jsou označeny unikátním čárovým kódem a na místo montáže jsou dopraveny dle připravených dat z velmi detailního projektu organizace výstavby. Přesná poloha výrobku při montáži může být určena pomocí lokálního pozičního systému (LPS), případně dle GPS dat kontrolována přes družice.

2.4 Metody v procesu navrhování

Digitální metody v procesu architektonického navrhovaní vznikají ze dvou základních důvodů - usnadnění práce nebo vývoj.

První případ je založen na digitalizaci již existujících (klasických) metod, které jsme byli schopni provádět ručně. Účelem změny je zrychlení a zpřesnění práce.

Ve druhém případě jsou vyvíjeny nové metody a prostředky pro navrhování. Jejich cílem je umožnit navrhování staveb, založených na složitějších principech či složitější geometrii, jejichž manuální návrh by byl příliš zdlouhavý, ne- li snad nemožný realizovat.

2.4.1 Digitální forma klasického kreslení

Základní metodou je digitální kreslení. Na procesu architektonického navrhování se po zavedení digitálního kreslení v principu nemusí nic měnit. Myšlenkový proces zůstává zcela stejný, pouze prostředky, jimiž se architekt vyjadřuje, se lehce mění. Digitalizace přináší do výkresů přesnost a geometrickou dokonalost. Některé postupy se výrazně zrychlují, jiné se naopak zbytečně komplikují.

Určitou kvalitativní změnu přináší komplexní softwary pro technické kreslení (computer aided design – CAD). Nabízí nám škálu předem definovaných prvků, s nimiž lze velmi efektivně a rychle manipulovat. Na jednu stranu je to usnadnění práce, na druhou stranu jsme rychlostí těchto standardizovaných primitiv předem motivováni k jejich používání. To má za následek přizpůsobování návrhů softwaru,

nikoli jeho účelu. Konečným důsledkem tak může být i omezování použitého tvarosloví či architektonického a výtvarného vyjadřování autora.

Pozitivní změnou je možnost kreslení ve 3D, zvyšuje se tím přehlednost a není tak zatížena představivost autora při navrhování stavby. Autor dostává okamžitou optickou zpětnou vazbu o svém návrhu.

Vzhledem k tomu, že se digitální forma kreslení stala v posledních letech obecným standardem v projektování, nemá smysl zde uvádět ani vytvářet příklady.

2.4.2 Navrhování s využitím složitých geometrických prvků

S vývojem grafických softwarů jsou nabízeny i nové možnosti a geometrické prvky, kterých lze využít. Jako příklad zde uvedu NURBS křivky a objekty (viz kapitola 2.1). Tyto objekty mají sice velmi složitý matematický zápis a komplikovanou geometrickou podstatu, jsou ale navrženy tak, aby byly jednoduše ovladatelné a práce s nimi přehledná. Pro běžného uživatele tak není vůbec nutné jejich studium či pochopení. Příkladem softwaru, který je přímo založen na práci s NURBS křivkami je program Rhinoceros. Jeho ovládání je natolik jednoduché, že se dá nazvat intuitivním. Otevřela se tak cesta k masovému navrhování zakřivených objektů.

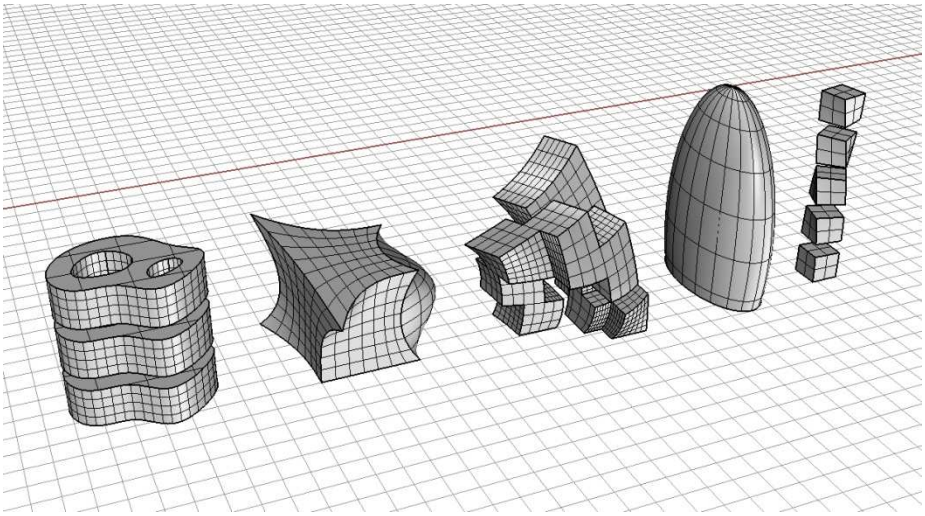
Formálně připomínají všechny takto vytvořené návrhy bionickou architekturu. Inspirace přírodou u mnoha návrhů přítomná není, bylo by tedy takové návrhy lepší obecně popisovat jako např. "infinitezimálně geometrické" podle podstaty geometrie, kterou byly definovány.

Návrhy oblých budov jsou často v odborných kruzích odsuzovány za tzv. formalismus – neopodstatněné užívání jiných forem než je "zvykem". V mnoha případech tomu tak i může být. Nejčastěji jsou takto napadány budovy s klasickými ortogonálními dispozicemi zabalenými do různě pokroucených fasád. Při takovém odsuzování je opomíjena nesmírně důležitá funkce estetická. Je také možné, že jsou nové formy kritikům jednoduše proti srsti a hledají pro svůj odpor nějaké zdůvodnění.

Hodnotit u této metody lze míru její aplikace. Je-li používána na celý návrh nebo jen na některé části. Někdy lze hodnotit i míru pochopení geometrické podstaty.

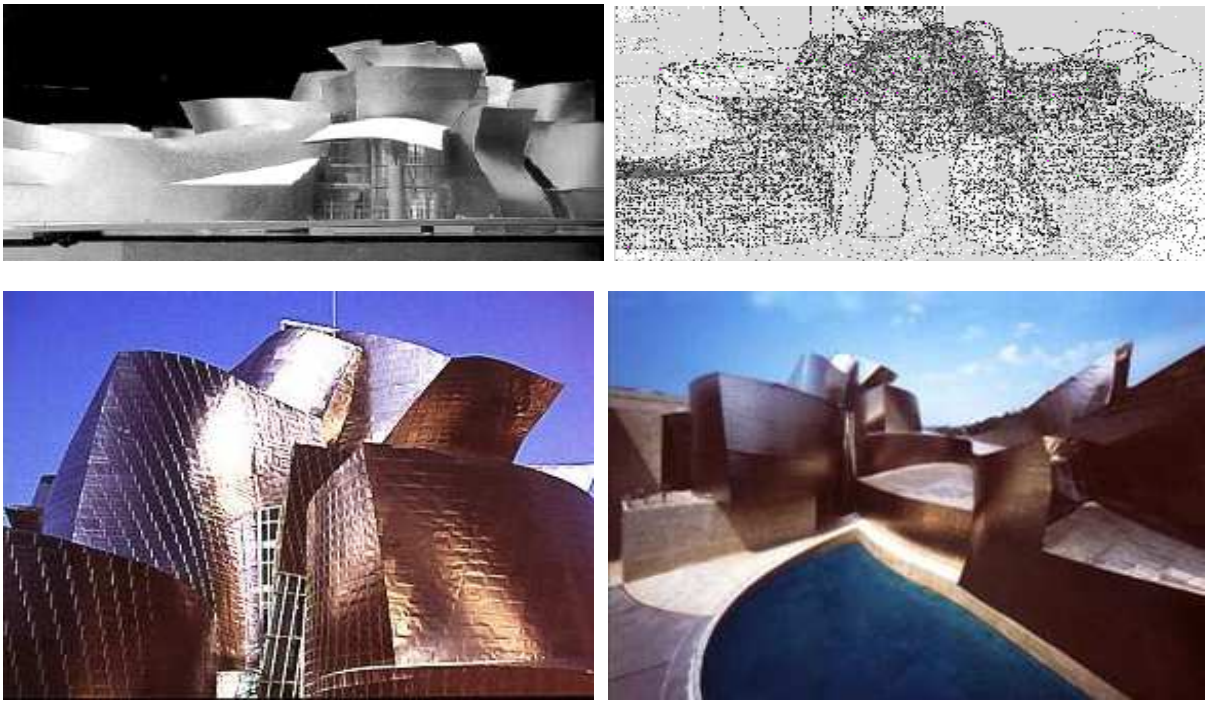
Pro názornost uvádím několik narychlo vytvořených objektů s geometrickou podstatou NURBS, vytvořit tyto objekty celkově mi trvalo asi 5 minut. Tím nechci shazovat návrhy, které takto vznikají, naopak chci prezentovat zrychlení

a uživatelskou pohodlnost metody kterou byly vytvořeny. Při výuce a hodnocení studentských prací je ale vhodné nenechat se zmást představou o náročnosti jejich tvorby.



obr. 16 Zakřivené geometrické objekty, Rhinoceros

Příkladem využití metody jsou práce F. Gehryho. Gehry je jedním z průkopníků topologické architektury a na jeho metodách je to stále znát. Kombinuje práci s digitální geometrií a práci s reálnými modely. Z jeho návrhů nikdy není zcela patrné, je-li tvar definován nejdříve digitálně nebo ručně – sochařsky.



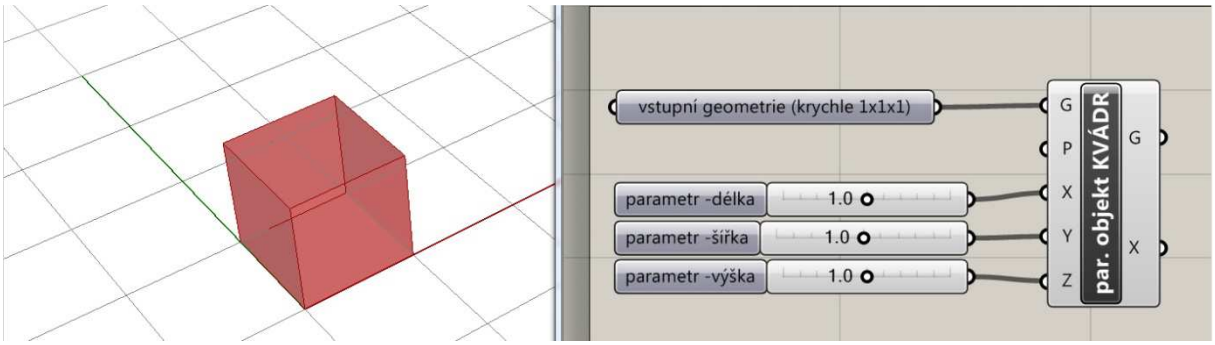
obr. 16 Guggenheimovo muzeum, Bilbao, Frank Gehry

2.4.3 Parametrická architektura

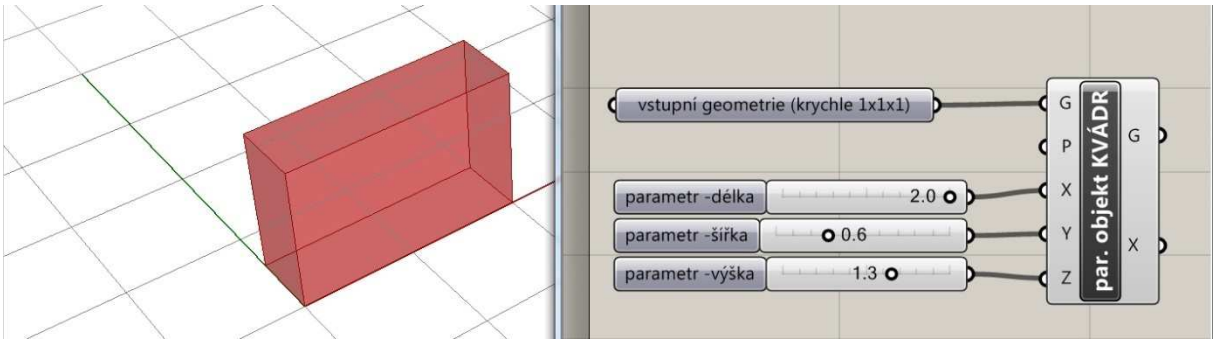
Principem parametrické metody návrhu je automatická modifikace vlastností navrhovaného objektu bez nutnosti celý návrh znovu kreslit.

Navrhovaný objekt je definován jako logicky provázaná struktura parametrických objektů a jejich parametrů – modifikátorů. Změníme-li v průběhu návrhu některý parametr, změna se projeví na všech objektech, které jsou na parametru závislé.

Jako příklad bych uvedl parametrický objekt kvádr, jeho parametry jsou např. délka, šířka, výška nebo poloha. V průběhu návrhu můžeme tyto vlastnosti kvádru libovolně měnit, aniž bychom jej museli znovu kreslit.

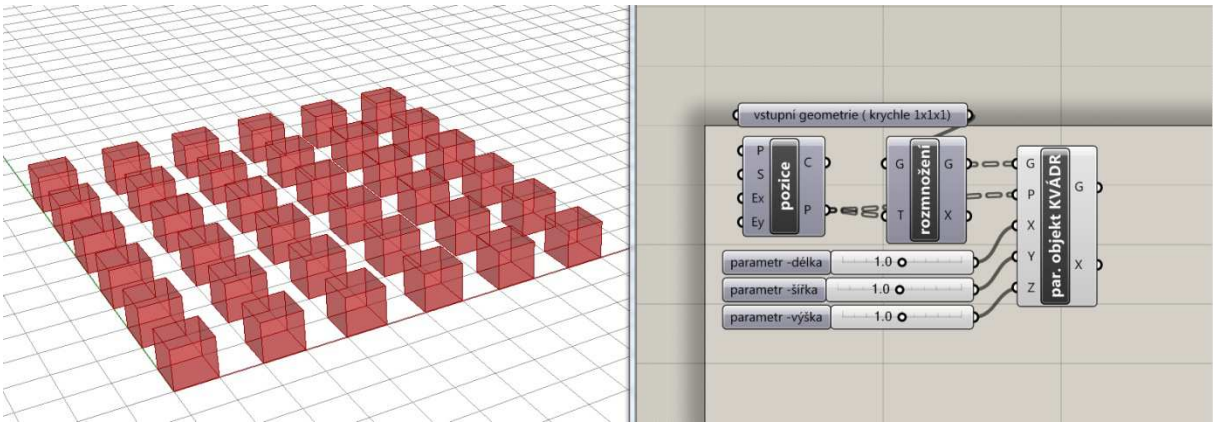


obr. 17 Parametrický kvádr – počáteční stav

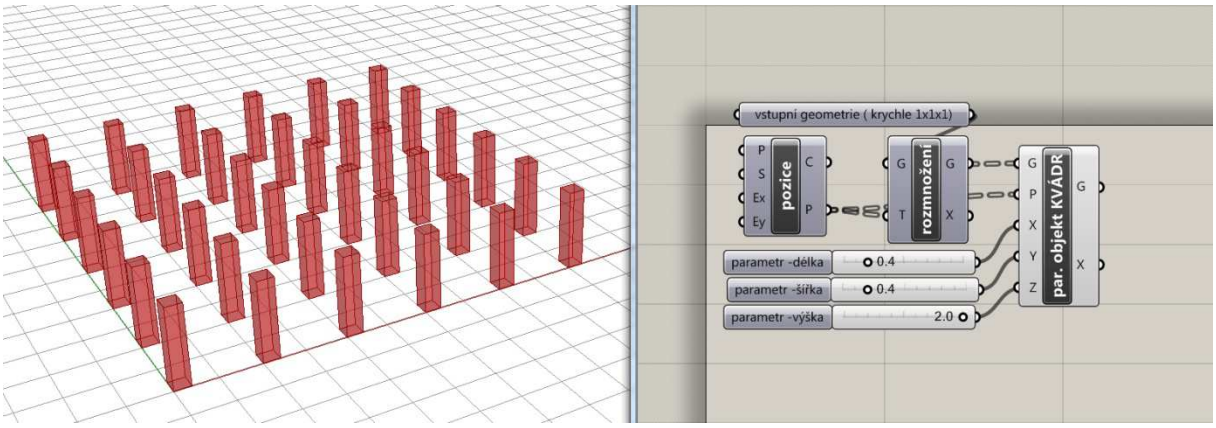


obr. 18 Parametrický kvádr – změna parametrů

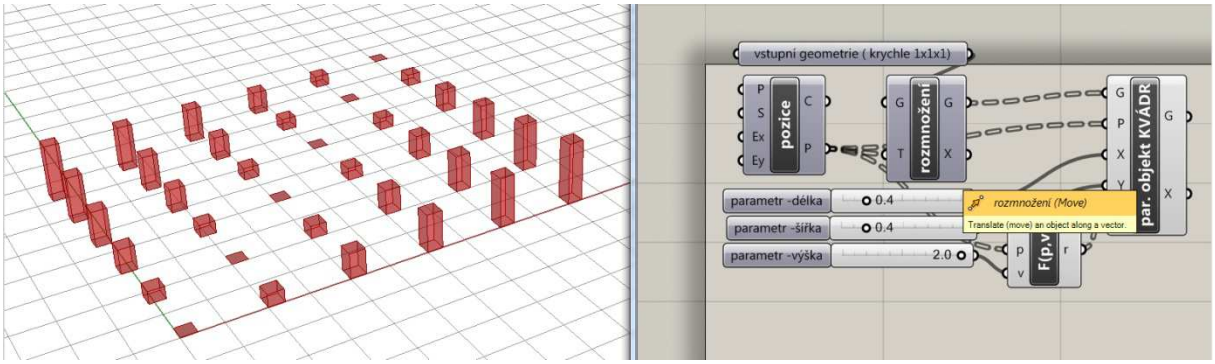
Metoda je velmi účinná, protože pomocí jednoho parametru můžeme měnit více podobných objektů najednou. Největším přínosem je ale možnost měnit skupiny parametrických objektů na základě pravidel každý trochu jinak.



obr. 19 skupina objektů – počáteční stav

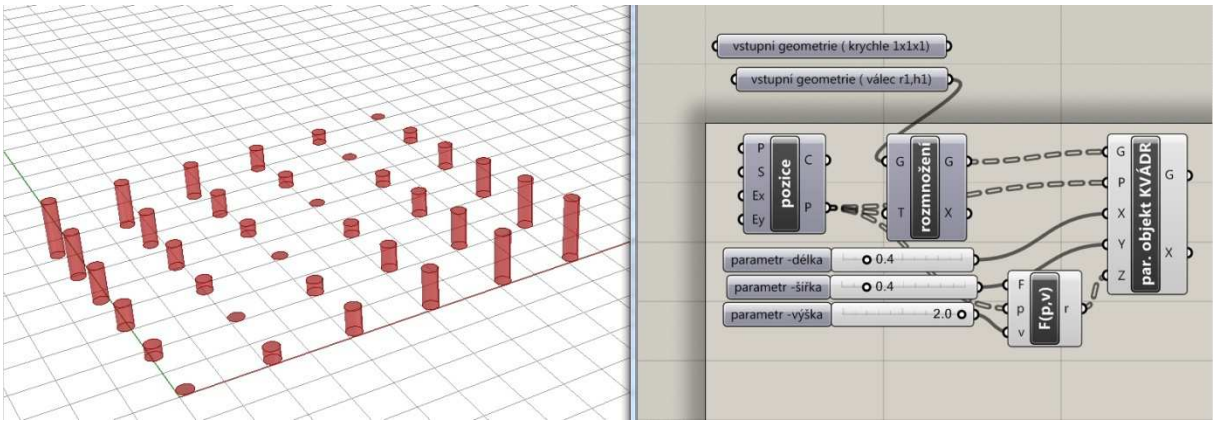


obr. 20 skupina objektů – identická změna aplikovaná na všechny objekty



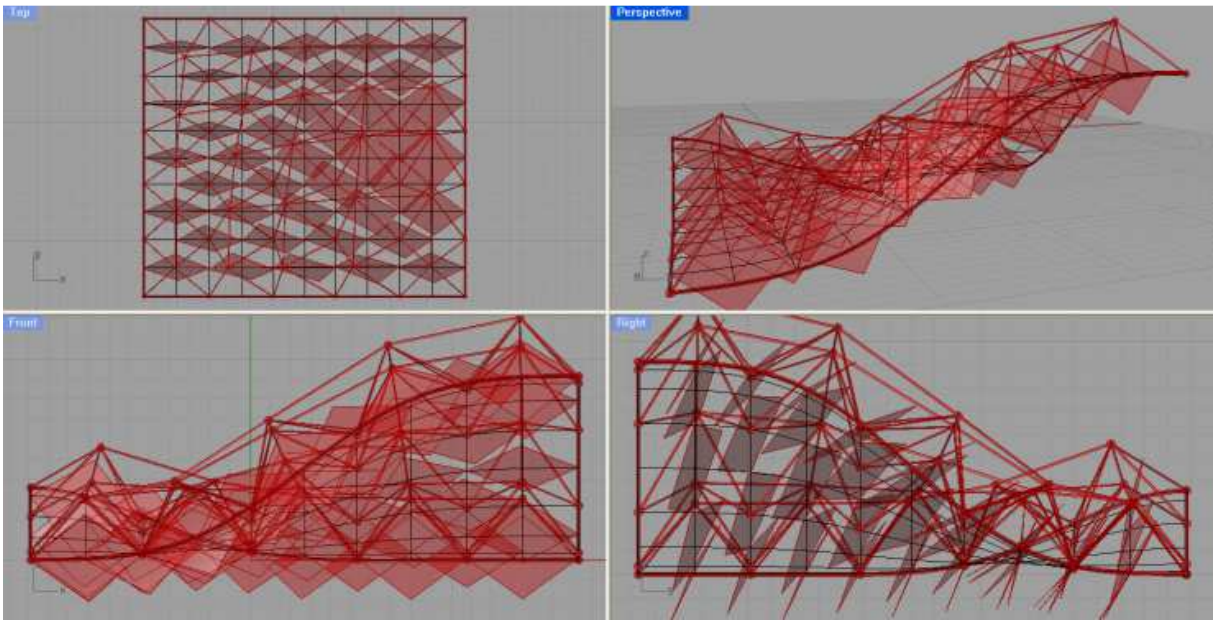
obr. 21 skupina objektů – parametry výšky závislé na poloze

Parametrický objekt jde v průběhu práce nahradit jiným, závislým na stejných parametrech



obr. 22 skupina objektů – záměna objektu ve struktuře

Je možné vytvářet různé komplikované struktury, hierarchicky navazovat objekty jeden na druhý a řídit je pomocí různých pravidel matematicky a algoritmicky definovaných.



obr. 23 strukturovaný parametrický návrh (3D Truss, Manuel Huerta, 2012)

Nevýhody této metody se projevují u velmi složitých struktur se složitými pravidly. Při přílišné složitosti už nejsme schopni domyslet, co může způsobit změna každého jednoho parametru. Může se stát, že změnou parametru na počátku hierarchie nastanou různé nechtěné změny na konci struktury. Vyladění všech parametrů, tak abychom dosáhli požadovaný výsledek, se stává příliš náročné.

Pro navrhování parametrických struktur byl před několika lety vyvinut jedinečný plug-in Grasshopper pod programem Rhinoceros. Má velmi příjemné

a pochopitelné uživatelské prostředí a díky tomu je většina současných parametrických projektů zpracovávána právě tímto softwarem.

Aplikace projektů vygenerovaných pomocí metody parametrického designu je umožněna díky novým výrobním možnostem CNC obráběcích strojů popsaných v kapitole 2.3.3 – výroba.

2.4.4 Architektura v pohybu

V této skupině metod je důležitým vstupem pohyb, tedy závislostí prostoru a času. Ve všech "pohybových" metodách je tedy v nějaké formě obsažena složka času.

2.4.4.1 Mechanický pohyb

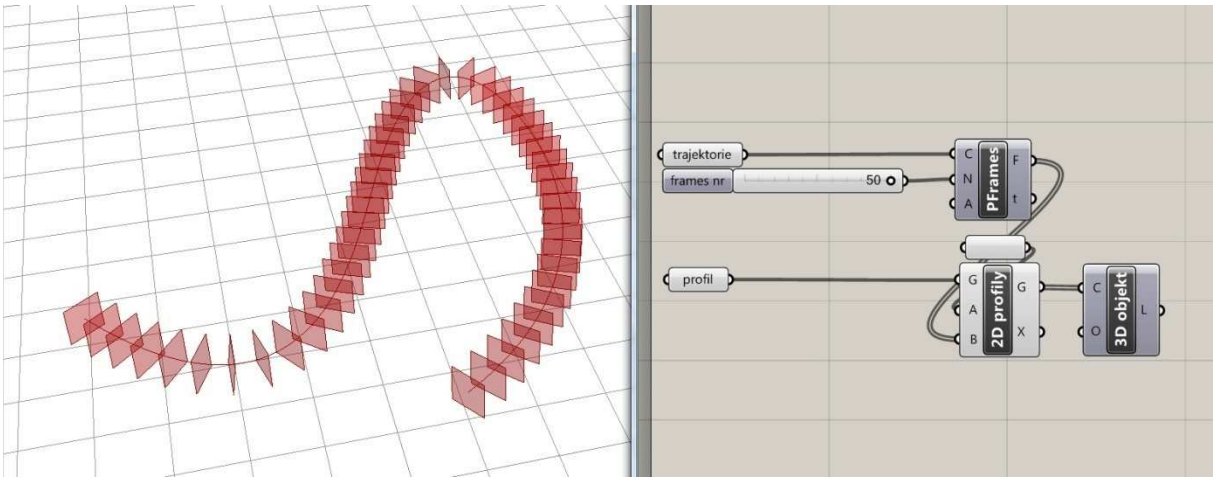
Klasické Newtonovské pojetí času a pohybu. Objekty, nebo jejich části se pohybují po trajektoriích v trojrozměrném prostoru v čase. Metody jsou založeny na snímání - sledování pohybu a na analýze pohybu.

Motion capture - zachycení pohybu, metoda je založena na způsobu získávání vstupních dat z reálného světa a jejich aplikaci v digitálním prostření. Systém byl vyvinut (nepočítám li armádu) pro animaci ve filmovém průmyslu. Snímače polohy umístěné na klouby pohybující se osoby předávají dostatečnou digitální informaci o pohybu člověka. Na základě těchto informací je možné takový pohyb zopakovat i v digitálním prostředí. Jacksonův videoklip s tančícími kostlivci je založen právě na této metodě.

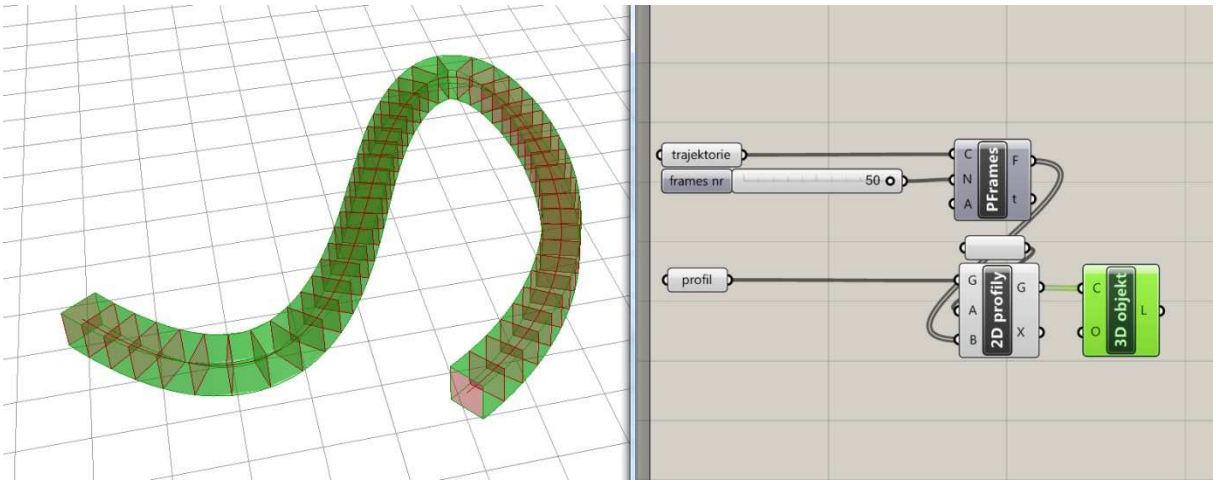
V architektuře se tento postup používá k prověřování chování lidí v prostoru, sledují se trasy jejich pohybu, prostor, který při pohybu potřebují, jejich interakce. Analýzou takových dat je možné prověřit vhodnosti různých prostorových řešení a ergonomicky je poté upravit. Pakliže někdo vysleduje v pohybech nějaká zásadní pravidla, je možné je aplikovat do pohybových simulací a ideální prostorové řešení hledat v digitálním prostření již dopředu – na základě teoretické pohybové analýzy navrhovat objekt.

2.4.4.2 Kinematická architektura

V této metodě není pohyb chápán jako pohyb hmotného reálného tělesa, pohyb je používán jako prostředek sloužící k návrhu objektů. Princip spočívá v rozpohybování dvourozměrných objektů po libovolné trajektorii v prostoru. Po určitých časových úsecích je poloha 2D objektu zapsána. V této fázi proces připomíná natáčení na kinofilm, kde je pohybující se objekt zachycen v různých polohách na jednotlivá políčka filmu. Výsledný 3D objekt vzniká propojením sekvence 2D příčných profilů do jednoho celku. Geometrický princip tohoto propojení bývá nazýván rybí kost. Časová složka je tak transformována do prostorových dimenzí. Výsledný objekt je statický, pohyb je ukryt v jeho genezi.



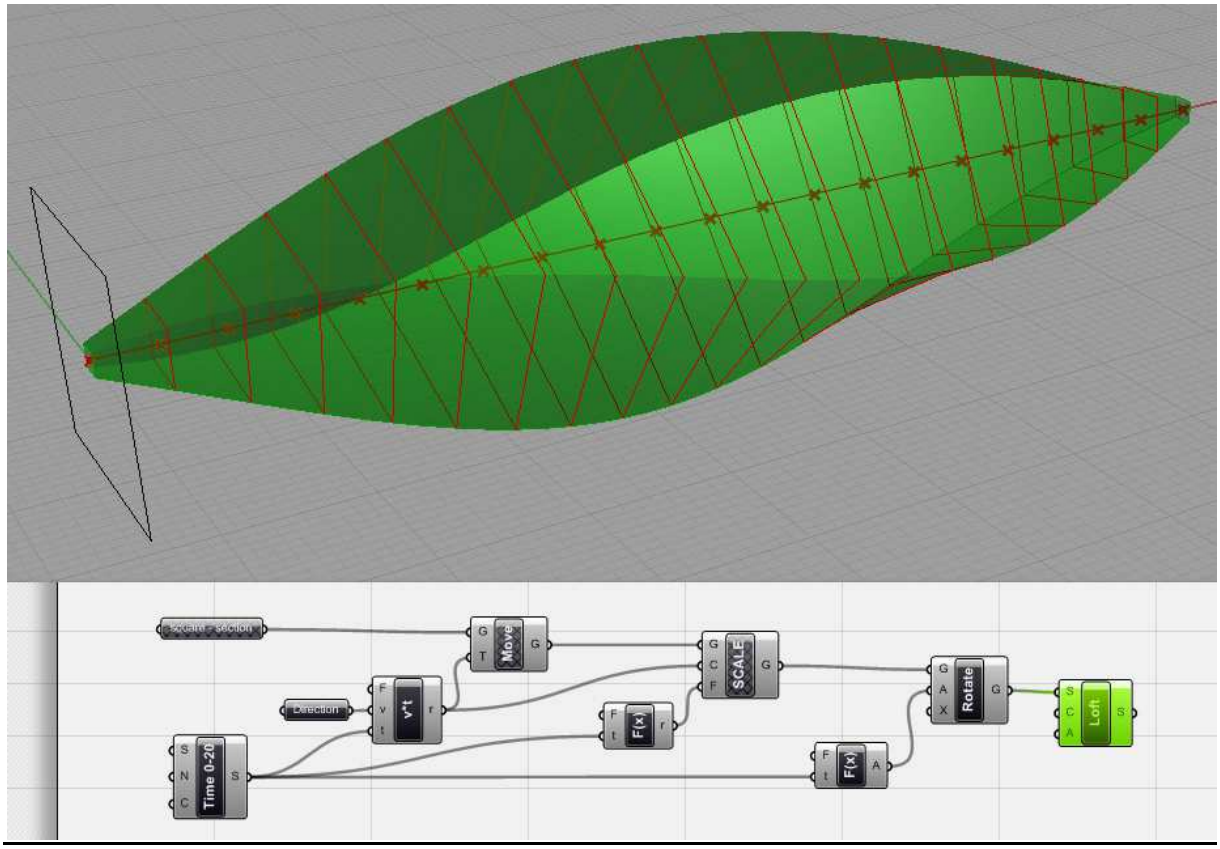
obr. 26 Sekvence 2D příčných řezů – rybí kost



obr. 27 Výsledný 3D objekt

Pohybující se objekt – profil - nemusí zůstat sám statický, může se v čase průběžně měnit.

Příklad: čtverec, pohyb po přímé trajektorii, rotace + zvětšení



obr. 28 průběžná změna profilu

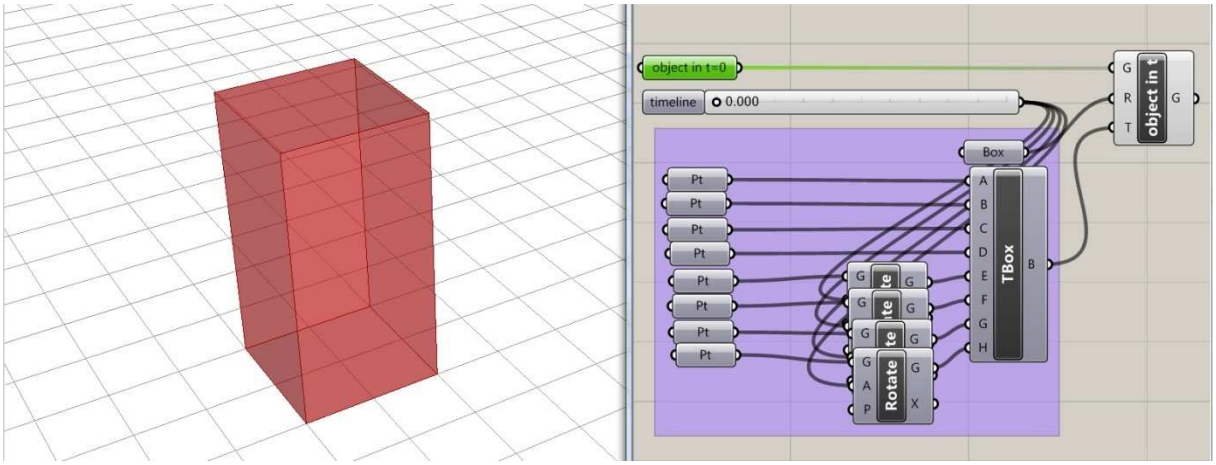
Příkladem realizace touto metodou je Hessing Cocpit poblíž Utrechtu od kanceláře ONL. Tvar budovy a protihlukové stěny u dálnice A2 je definován pomocí průběžné změny profilu budovy při průjezdu po dálnici.

obr. 29-30 Hessing Cockpit in Utrecht, NL, Kas Oosterhuis – ONL

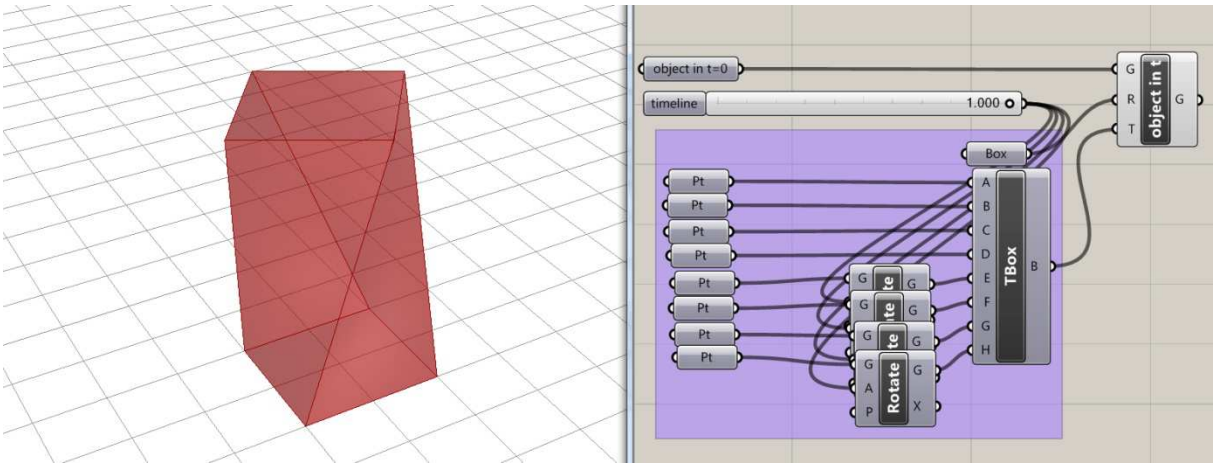
2.4.4.3 Metamorfická architektura

Stejně jako v předchozí metodě, je skryt pohyb v genezi objektu. Na rozdíl od kinematické architektury je zde deformován trojrozměrný objekt v závislosti na čase. Proces je nazýván "morphing". Spočívá v definování výchozího stavu, konečného stavu, případně klíčových mezistavů objektu v čase. Jednotlivé klíčové stavy jsou definovýny pomocí jednoduchých transformací počátečního objektu. Následně je vypočítán plynulý průběh transformace objektu od počátečního stavu přes předdefinované klíčové momenty až do stavu konečného. V tuto chvíli je objekt čtyřrozměrný, má podobu animace. Výsledný objekt je definován jako trojrozměrný řez čtyřrozměrného objektu – jednodušeji řečeno: stačí vybrat polohu na časové ose, ve které nám 3D objekt vyhovuje.

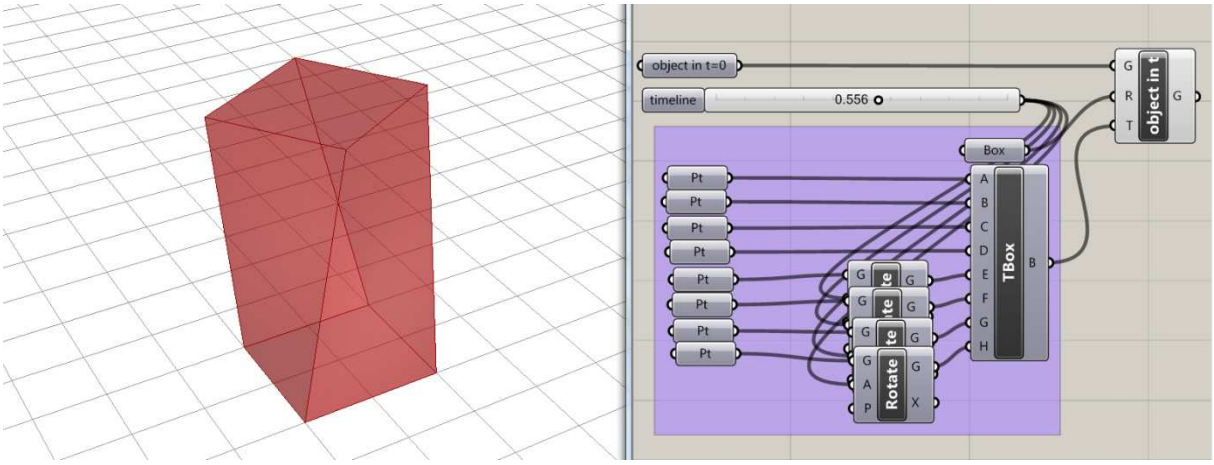
Příklad: *Jako jednoduchý příklad této metody předkládám deformaci kvádru. V počátečním stavu t=0 je kvádr nedeformovaný, poté je pomocí transformace uveden do stavu t=1 (transformace je ukryta v modrém poli algoritmu), následně je vypočítán plynulý průběh z t=0 do t=1. Na závěr jsem po prohlédnutí animace vybral stav v t=0.556.*



obr. 31 morphing – stav t = 0, počáteční stav



obr. 32 morphing – stav t = 1, definovaná transformace



obr. 33 morphing – stav t = 0.556, vybraný objekt z animace

Pro práci touto metodou doporučuji využití softwaru 3D studio MAX, který je primárně vyvinut pro vizualizace a animace. Algoritmy pro práci s klíčovými momenty (keyframes) a pro výpočet plynulých transformací jsou již v programu obsaženy.

2.4.4.4 Vnitřní pohyb

Poslední skupinou objektů, jejichž definice v sobě ukrývá pohyb a složku času, zde nazývám vnitřní pohyb. Tak jako v posledních dvou metodách byl čas ukryt v procesu tvorby, v této variantě bude ukryt ještě o něco hlouběji, v samotné podstatě objektů, v geometrických definicích. Nejde ani tak o metodu jako spíš o matematický či geometrický princip, který může celou práci ovlivnit a definovat.

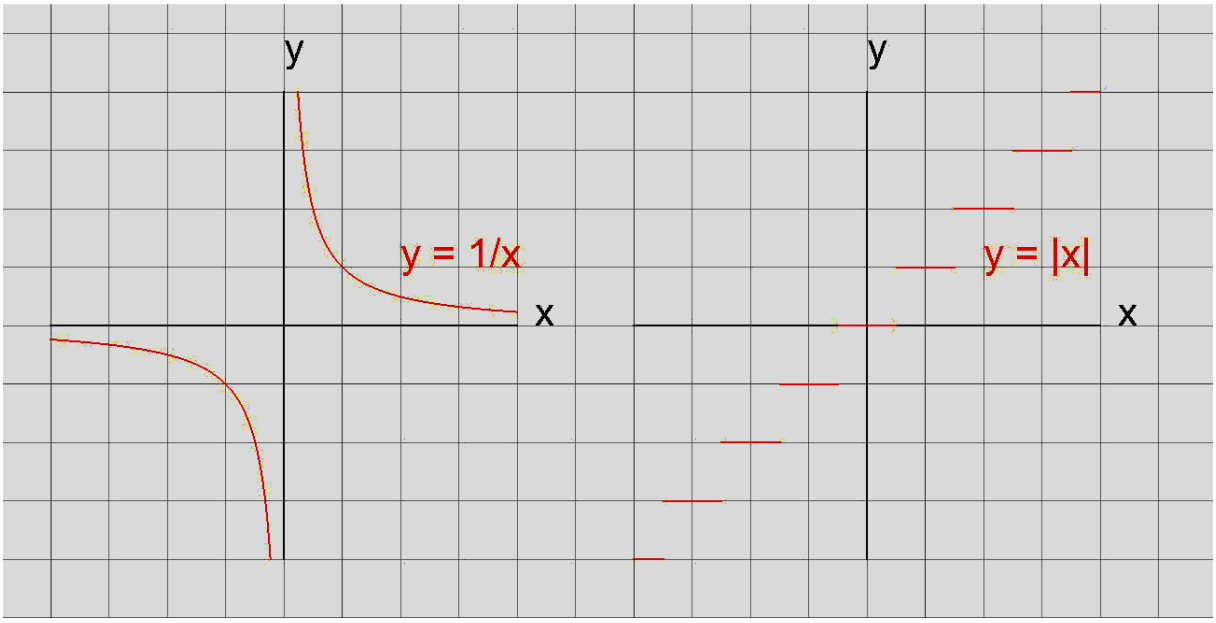
Pracuje se zde s geometrickými objekty na bázi infinitezimálního počtu. Některá základní geometrická primitiva jsou již popsána v kapitole 2.1.4 – 2.1.6

(parametrické objekty). Vždy jde o křivky a plochy, jejichž vlastností je kontinuita, u všech je vlastností tok v čase.

A) kontinuita křivek

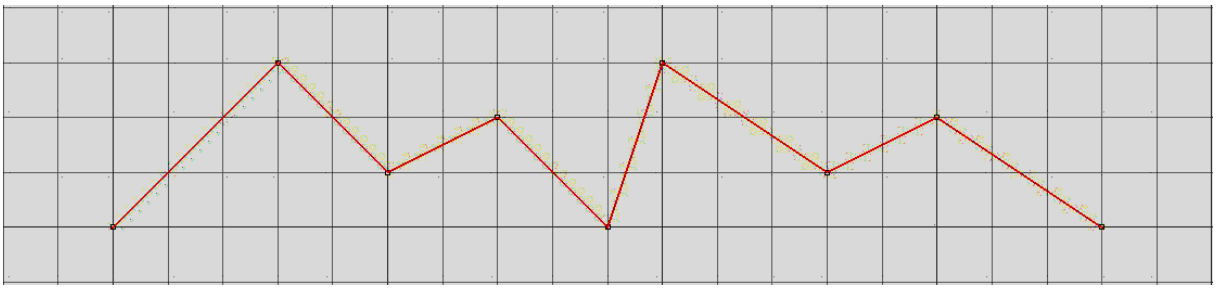
V této chvíli se bohužel musím vrátit k matematickému vysvětlení kontinuty u křivek, kterému jsem se v kapitole 2.1 raději vyhýbal.

U křivek vnímáme geometrickou kontinuitu (diskontinuitu) značenou "G" a kontinuitu parametrickou "C". Křivka s geometrickou kontinuitou G^0 je definována jako geometricky spojitá – nepřerušená. Geometrická diskontinuita G^0 znamená přerušení křivky, konce dvou segmentů křivky se tedy nedotýkají. Typickou ukázkou diskontinuity G0 jsou křivky dané rovnicí f: y=1/x nebo f: y ≈ (x). V prvním případě je diskontinuita v x=0, ve druhém je diskontinuit nekonečno.



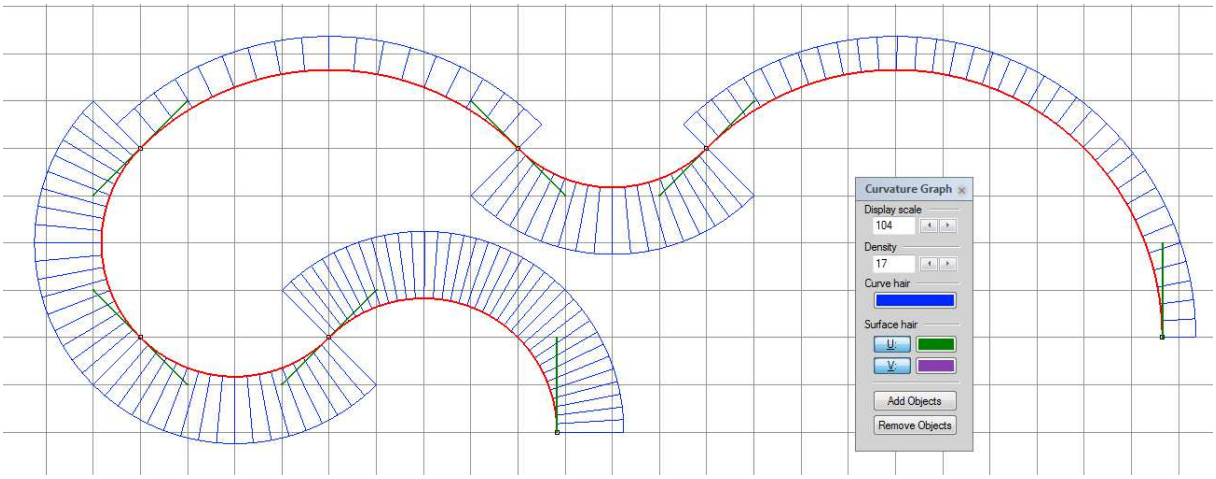
obr. 34 diskontinuita G^0

Pro spojení řady bodů v prostoru křivkou s G^0 nám postačuje soustava křivek 1° tedy lomená čára, jejíž segmenty jsou úsečky.



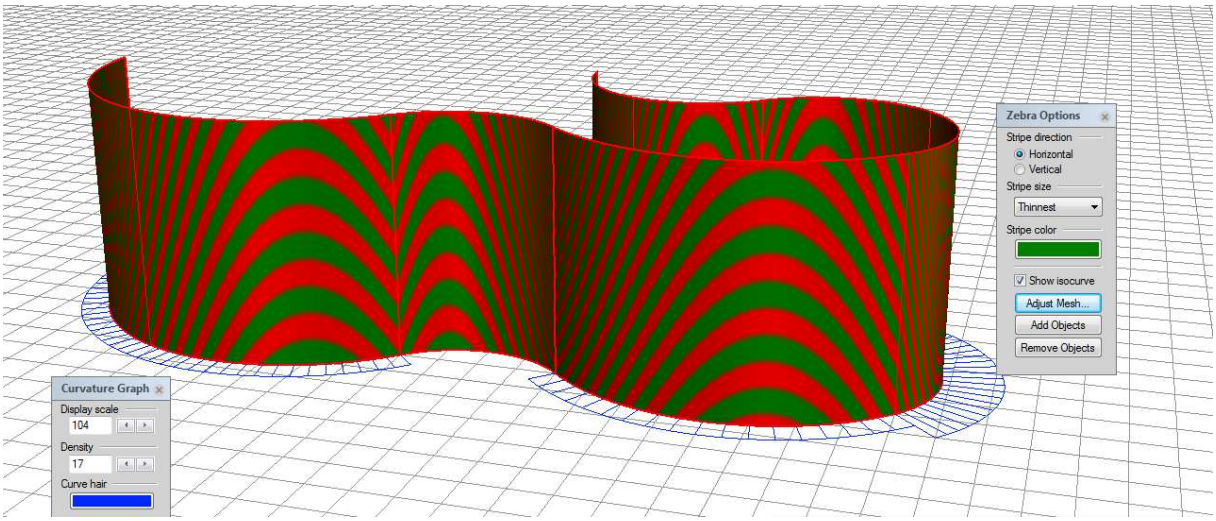
obr. 35 kontinuita G^0, křivka 1°

Geometrická kontinuita G^1 znamená tangenciální spojitost. Křivka G^1 nemá lomy, plochy nemají vnitřní hrany. Pro spojení řady bodů v prostoru křivkou s kontinuitou G^1 nám postačuje soustava křivek 2° (např. kruhových oblouků). Diskontinuita G^1 se projeví jako lom křivky, hrana u plochy.



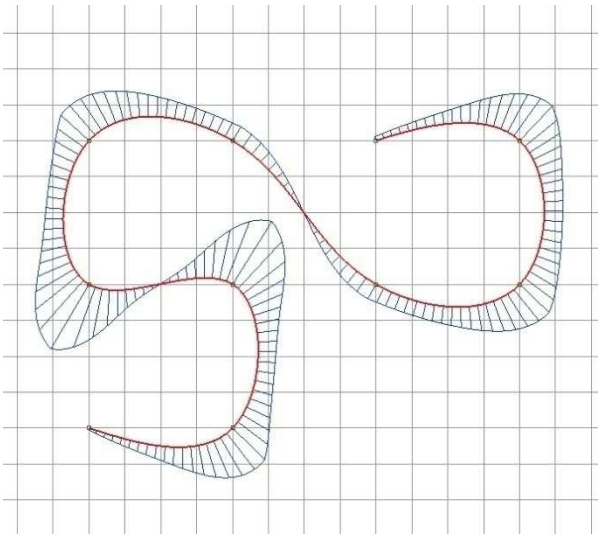
obr. 36 kontinuita G^1, křivka 2°

Kontinuita G^2 znamená kontinuitu křivosti, poloměr křivosti se mění plynule. Pro spojení řady bodů v prostoru křivkou s kontinuitou G^2 již potřebujeme křivky 3°, např. Beziérovu křivku, spline či NURBS. Diskontunita G^2 se projevuje skokovou změnou poloměru, nevzniká sice ostrá hrana, ale u zrcadlících objektů je diskontinuita patrná lomy odrazu.

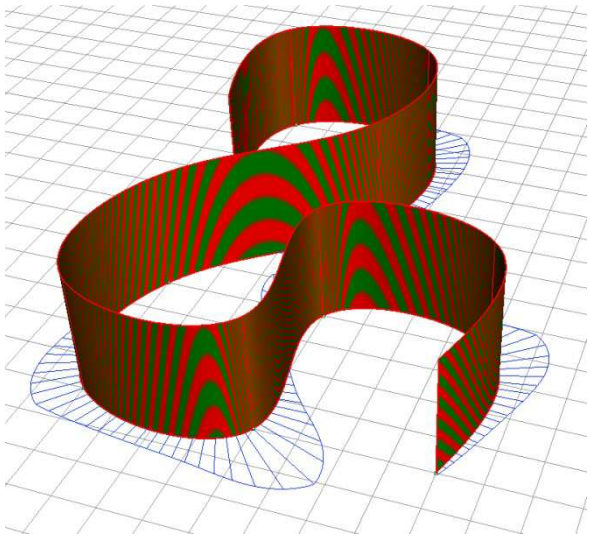


obr. 37 diskontinuita G^2, plocha 2°

U křivek s kontinuitou G^2 již můžeme sledovat plynulou změnu poloměru (porovnej analýzu poloměru křivky – modře – v obr. 36 a obr. 38) i plynulé návaznosti odrazu (porovnej odrazy zelených pruhů v obr. 37 a obr. 39)



obr. 38 kontinuita G^2, křivka 3°



obr. 39 kontinuita G^2, plocha 3°

Obecně platí, že čím vyšší geometrickou kontinuitu požadujeme, tím vyšší stupeň křivky k tomu potřebujeme. Potřebný stupeň křivky je o řád vyšší než stupeň kontinuity.

Křivky s vyšší kontinuitou než G^2 ale nemají žádné nové zásadní vlastnosti z hlediska tvaru ani viditelných vnějších projevů. Z toho důvodu se v největší míře používají křivky 3°. Křivky 4° a 5° s G^3 a G^4 se vyskytují např. v automobilovém a leteckém průmyslu, kde diskontinuity tvaru objektu řádu G^2 a G^3 ještě mají za

důsledek nežádoucí turbulence aerodynamického proudění. Křivky s vyšším řádem používají snad pouze ekonomové a politici pro zkreslení neblahých důsledků svých činů.

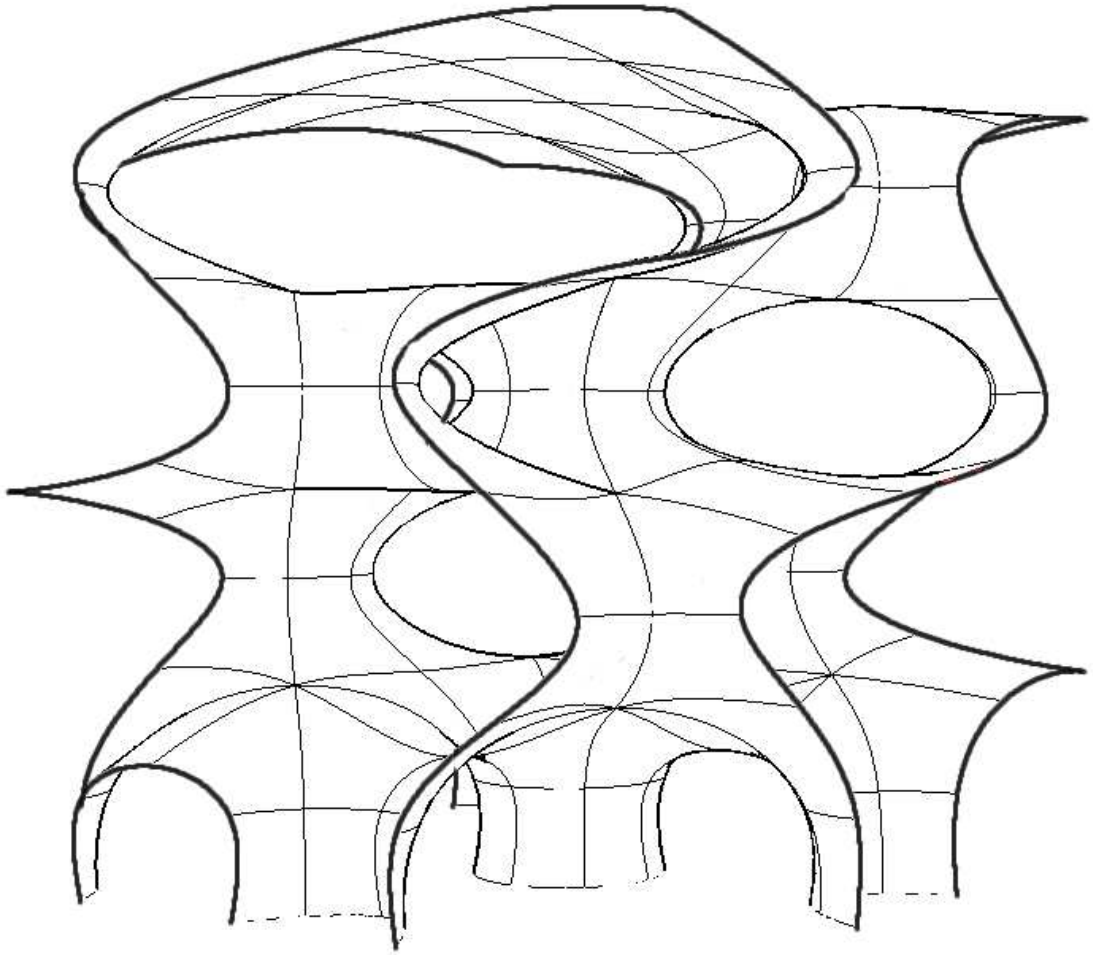
Rozdíl mezi geometrickou kontinuitou G^n a parametrickou kontinuitou C^n není ve většině případů znatelný. Zatímco geometrická kontinuita je geometrickou vlastností křivky, parametrická kontinuita je matematickou vlastností popisující vznik geometrie. V hraničních případech se G^n a C^n neshodují. Jako jednoduchý příklad uvedu řadu pěti bodů ležících na přímce. Propojím-li je křivkou 3° (např NURBS) nastane v bodech 2 a 4 automaticky parametrická diskontinuita C^3 . Vzhledem k tomu, že body leží na přímce, bude křivka i přes svou definici mít geometrické vlastnosti úsečky – tedy křivky 1° a žádnou geometrickou diskontinuitu vyšší než G^0 tedy mít nebude. V bodech 2 a 4 je křivka dokonale geometricky kontinuální. Parametrická diskontinuita je tak opticky nerozpoznatelná.

Existence skryté parametrické diskontinuity způsobuje nežádoucí projevy v dalším zacházení s křivkou. Posuneme-li například jeden z bodů křivky poblíž diskontinuity parametrické, projeví se i diskontinuita geometrická, kterou jsme předem neočekávali. Eliminace parametrických diskontinuit vyžaduje zvláštní algoritmy.

B) Kontinuální plochy

Nyní se můžu vrátit k využívaným teoretickým principům v architektuře. Objekty definované plochami, jejichž vlastnosti jsou výše popsané, mají jednu zásadní vlastnost – plynulost, kontinuitu. Objekty jsou navrhovány jako komplexní geometrické útvary, často definované jedinou plynule zakřivenou plochou.

Pro znázornění jsem jednu takovou strukturu definovanou jednou kontinuální plochou vyrobil (ortogonální řezy jsou definovány spline křivkami 3°, plochy NURBS). Objekt není vytvořen pomocí algoritmů, je definován pomocí jednotlivých ručně definovaných řezů budoucím objektem, podobnost s objekty vzniklými algoritmicky byla žádoucí.



obr. 40 kontinuální plocha

Celý objekt definován jako jedna parametrická plocha. Každý bod parametrické plochy je definován jako pohybující se po ploše - jeho parametrem je poloha v prostoru závislá na řídících časech (*u* a *v*). Můžeme tedy objekt chápat jako pohybující se, pohyb je však ukryt hluboko v definici.

C) Tekoucí prostor

Posuneme-li se v definici o řád výše a najdeme definici pro bod pohybující se v trojrozměrném objektu (nikoli pouze po ploše), budeme-li schopni takový objekt navrhnout, pak jej můžeme nazývat tekoucím prostorem. Pohyb bodu v objektu bude definován ve třech řídících směrech (nikoli totožných s naším souřadným systémem) v závislosti na třech různých časech. Výsledkem je komplikovaný volumetrický objekt se složitými vnitřními geometrickými pravidly.

Pro představu – když vybereme pouze jeden okamžik na jedné časové ose, podle níž je objekt definován, a zobrazíme jej, dostaneme kontinuální plochu popisovanou v předchozí kapitole. Tekoucí prostor lze tedy definovat jako stopu pohybující se kontinuální plochy.

Obdoba tekoucího prostoru bývá používána pro zobrazování fyzikálních hypotéz, např. zakřiveného prostoru dle Einsteinovy obecné teorie relativity.

2.4.5 Matematicky definovaná architektura, matabally

Metoda je z pohledu matematiky přehledná a jednoduchá. Na základě jednoduchých matematických rovnic a jejich kombinací jsou definovány komplexní geometrické tvary. Z hlediska geometrie je metoda trochu složitější, musíme být totiž schopni si uvědomit, co která matematická rovnice znamená, a jak s ní zacházet. Z pohledu architektury je pro nás metoda výzvou, protože nám přináší nepřeberné množství nových tvarů. Schopnost pracovat s nimi je však bohužel většinou podmíněna hlubokými znalostmi matematiky a geometrie.

2.4.5.1 Matematické vyjádření

Na rozdíl ode všech předchozích metod, kde byla geometrická primitiva definována výhradně v parametrickém tvaru, v této metodě pracujeme s rovnicemi ve tvaru libovolném.

Např. přímka:

Geometrie:

Přímka procházející bodem: A (x_0,y_0,z_0) se směrovým vektorem v (v_x,v_y,v_z)

Rovnice v parametickém tvaru:

$$x = x_0 + t * v_x$$
$$y = y_0 + t * v_y$$
$$z = z_0 + t * v_z$$

kde je (x_0, y_0, z_0) je libovolný bod, kterým přímka prochází, (v_x,v_y,v_z) jsou konstanty (vektor v) určující směrnici přímky a t je parametr

Soustava rovnice přímky v obecném tvaru:

$p: (a_1x + b_1y + c_1z + d_1 = 0) \& (a_2x + b_2y + c_2z + d_2 = 0)$

rovnice je dána průsečíkem dvou rovin – konjunkcí rovnic dvou rovin

Usnadnění je viditelné např. u vyjádření kružnice v ploše

Parametrické vyjádření v závislosti na úhlu:

$x = x_0 + r * (\cos \varphi)$
$y = y_0 + r * (\sin \varphi)$

při implementaci v parametrickém trvaru je ale kružnice rozdělena a vyjádřena soustavou tří nebo čtyř beziérových křivek 2° odpovídajících kruhovým obloukům, což už tak snadné není.

vyjádření na základě vlastnosti (vzdálenost od středu je rovna poloměru)

$r = \sqrt{((x_p-x_s)^2 + (y_p-y_s)^2)}$,

kde *r* je poloměr, *s* střed a *p* bod na kružnici

pseudokód pro implementaci je pak ještě jednoduší:

if dist(s, p) = r then P leží na kružnici

Čím bude geometrie složitější, tím bude výhodnější vyhýbat se parametrickému vyjádření. U některých složitých rovnic je dokonce převod na parametrickou formu nemožný.

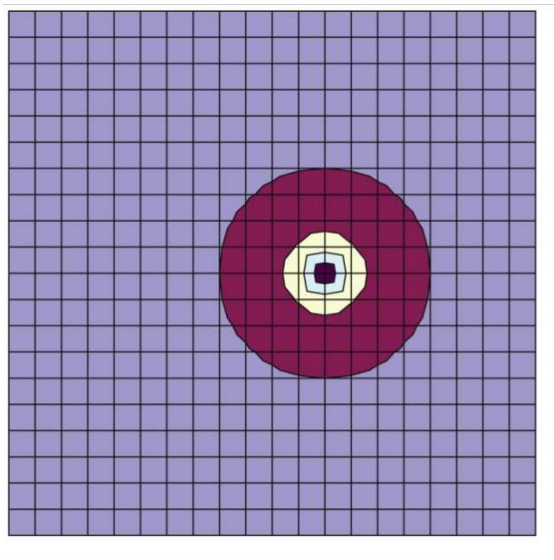
2.4.5.2 Metaball

Příkladem je tzv. metaball, prostorový objekt definovaný na základě součtu intenzit od několika zdrojů.

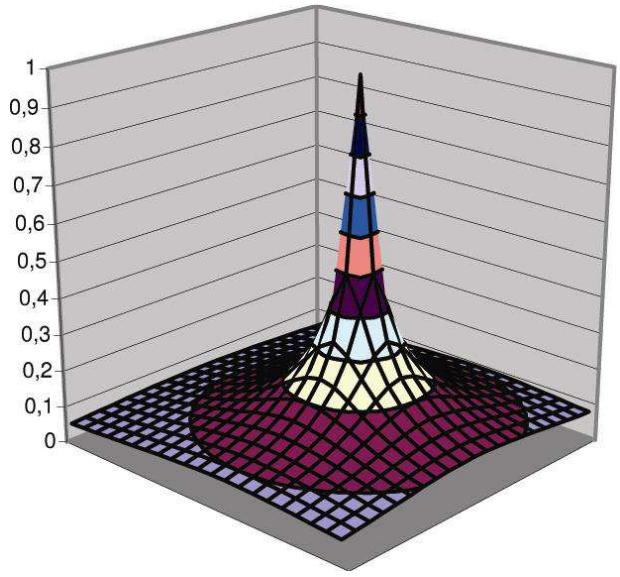
A) Objekt definovaný jediným zdrojem

bod G (x_g, y_g, z_g) generuje pole intenzity do okolí
intenzita v libovolném bodě P (x_p,y_p,z_p) je nepřímo úměrná vzdálenosti od bodu G
i_g je počáteční intenzita definovaná v bodě G
i_p je srovnávací intenzita, v níž je výsledný objekt limitován, ořezáván

prostorový objekt je dán nerovnicí:
i_g / (1 + dist(P, G)) < i_p
povrchová plocha je dána rovnicí:
i_g / (1 + dist(P, G)) = i_p



obr. 41 2D řez objektem mataball(1)



obr. 42 diagram intenzity v ploše

B) Objekt definovaný více zdroji

Body G_1 (x_{g1}, y_{g1}, z_{g1}) a G_2 (x_{g2}, y_{g2}, z_{g2}) generuji pole intenzit do okolí
i_{g1} a i_{g2} jsou počáteční intenzity definované v bodech G_1 a G_2

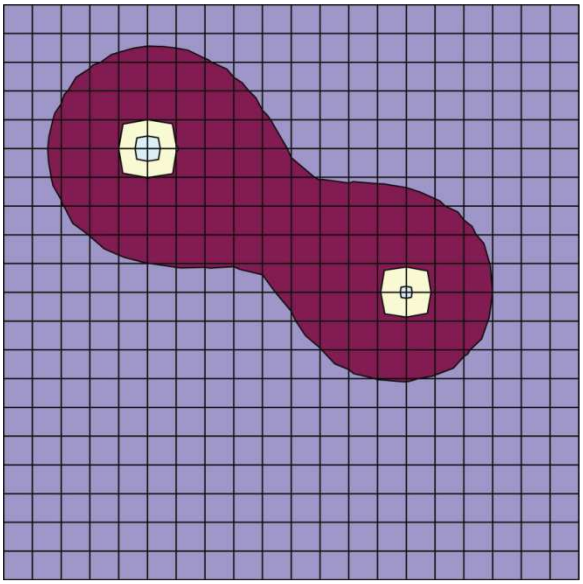
intenzity $i_{p,g1}$ a $i_{p,g2}$ v libovolném bodě P (x_p,y_p,z_p) jsou nepřímo úměrné vzdálenostem od bodů G_1, G_2

Výsledná intenzita v prostoru je dána součtem jednotlivých intenzit generovaných body G_1 a G_2

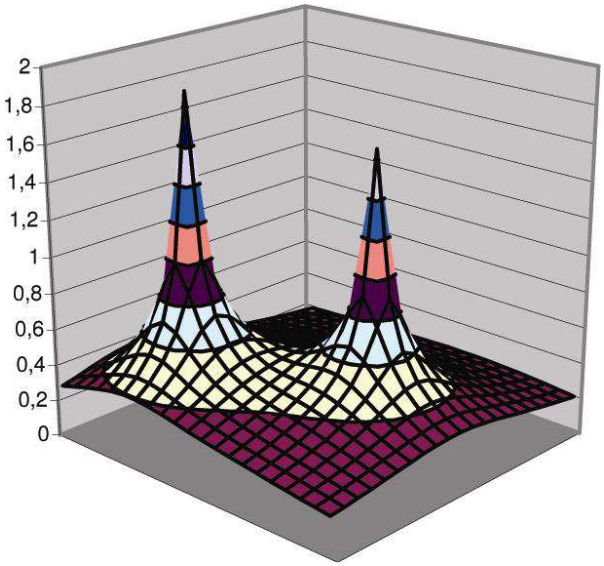
Objekt je dán nerovnicí:

$$i_p > i_{g1} / (1 + dist(P, G_1)) + i_{g2} / (1 + dist(P, G_2))$$

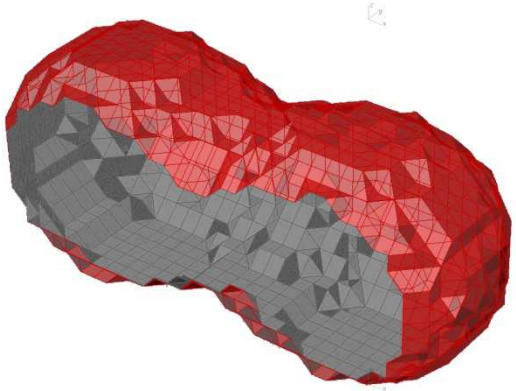
která říká: je-li intenzita v bodě P menší než intenzita srovnávací i_p, pak bod P náleží objektu



obr. 43 2D řez objektem mataball(2)



obr. 44 diagram intenzity v ploše



obr. 45 Mataball 3D – povrch

2.4.5.3 Grafická interpretace

Přestože metoda přináší nové možnosti a výhody, má i značné nevýhody. Tou největší je způsob zobrazování, grafické interpretace. Problém spočívá v nemožnosti parametrického vyjádření objektů, nemáme tedy možnost vykreslit objekt klasickými metodami požívanými v CAD systémech.

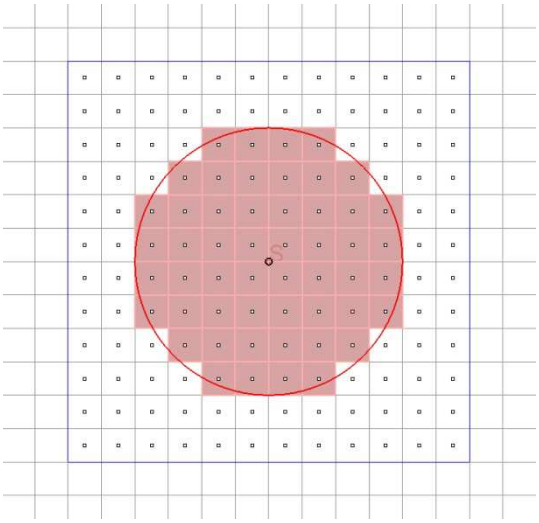
Z podstaty řešení neparametrických rovnic jsme schopni získat pouze binární informaci o bodě v prostoru a jeho vztahu k objektu. Touto informací je pouze: patří do objektu / nepatří do objektu. Z toho vyplývá zásadní problém: nejsme dopředu schopni určit body, jimiž je objekt vyjádřen. Musíme tak prověřit všechny body v prostoru a provést jejich selekci dle rovnic.

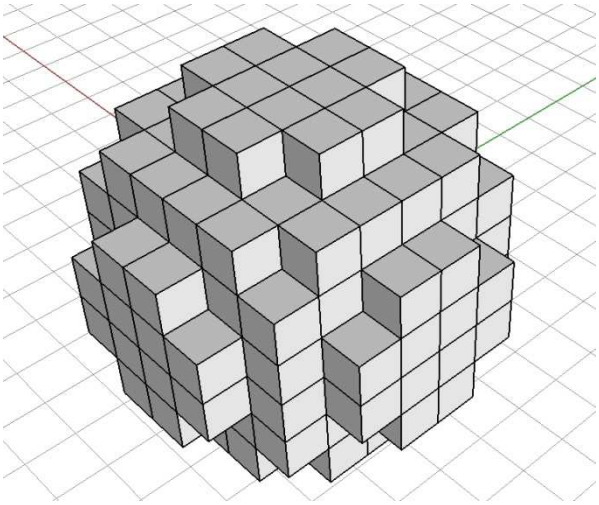
A) Voxelová / pixelová interpretace

Tato metoda již byla popsána v kapitole 2.1.3, zde uvedu příklad postupu pro vyjádření kruhu.

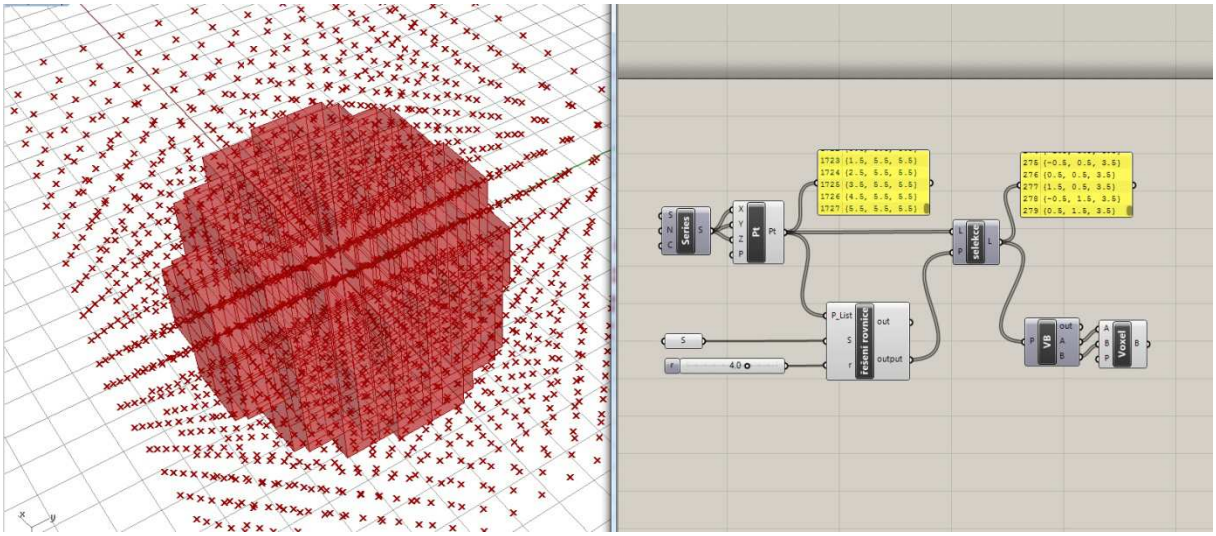
if dist(s, p) < r then P je bod kruhu

- definujeme přesnost (rozlišení) – množství pixelů, pro něž bude výpočet řešen (modře)
- pro každý bod vyřešíme rovnici
- vybereme body, pixely, které rovnici vyhovují (růžová plocha)





obr. 46 pixelová interpretace *obr. 48 voxelová interpretace koule*



obr. 47 voxelová interpretace koule - proces

Příklad implementace výpočtu rovnice:

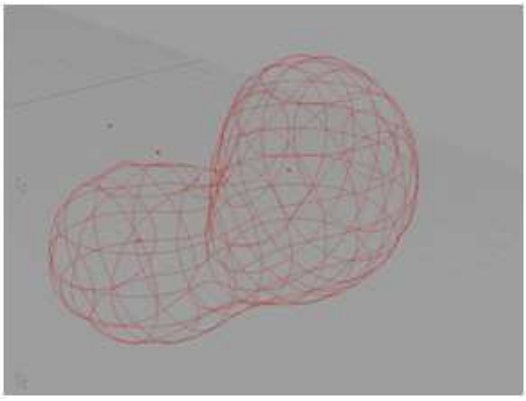
```
Private Sub RunScript(ByVal P_List As List(Of Point3d), ByVal S As Point3d, ByVal r As Double, ByRef
output As Object)

   Dim i As Integer
   Dim Bool_List As New list(Of Boolean)   '(seznam výsledků rovnice)
   For i = 0 To P_list.count - 1           '(pro každý bod)
      Dim P As New point3D(P_List(i))      '(jeden řešený bod)
      Dim Bool As Boolean                  '(binární informace o řešení rovnice)
      If (P.DistanceTo(S)) < r Then        '(nerovnice definující kouli)
         Bool = True                       '(bod vyhovuje rovnici, výsledek = true)
      Else
         Bool = False                      '(bod nevyhovuje rovnici, výsledek = false)
      End If
      Bool_List.add(Bool)                  '(zápis výsledku do seznamu)
   Next 'i
   Output = Bool_List                      '(přiřazení - export dat)

End Sub
```

B) Teselace

Předchozí způsob je jednoduchý, pochopitelný a snadný na programování, jeho nevýhodou je určitá hrubost, to lze částečně napravit posunem hraničních bodů (bodů na povrchu) do pozice, která více odpovídá rovnici povrchu. Dosáhneme toho pomocí iterace, aproximace.

C) Zobrazení řezů objektem

Jednoduchá a rychlá reprezentace komplikovaných matematických objektů. V každé řezové rovině definujeme plochu, jejíž každý bod posuneme po normále do vzdálenosti intenzity v bodě. Řezem takové plochy ve výšce srovnávací intenzity je křivka, která je řezem povrchu objektu.



obr. 49 interpretace řezem

obr. 50 interpretace řezem

2.4.5.4 Nevýhody metody

Jakákoli chyba v zápisu rovnice má za důsledek kolaps systému nebo nesmyslný výsledek. Taková chyba se těžko dohledává a opravuje.

Objekty jsou ovladatelné pouze pomocí rovnic, přesné vyladění výsledného tvaru je tak obtížné.

Neexistuje zpětná vazba. Změníme – li cokoli na výsledném objektu jinak než pomocí rovnic, rovnici to nezmění, případná další úprava rovnic znehodnotí předchozí úpravy.

Náročnost na dobu výpočtu. Špatný odhad nebo neznalost podstaty metody může způsobit nevhodné nastavení rovnic či rozlišení nastavení. To poté znamená exponenciální nárůst požadavků na množství výpočtu, zpomalení nebo dokonce přetížení paměti.

2.4.6 Algoritmické metody

Metoda je založena na automatickém zpracování vstupních dat pomocí algoritmů do výsledné podoby. Pomocí algoritmů můžeme definovat chování jednotlivých zpracovávaných objektů, jejich vzájemné vztahy, transformace nebo např. výskyt.

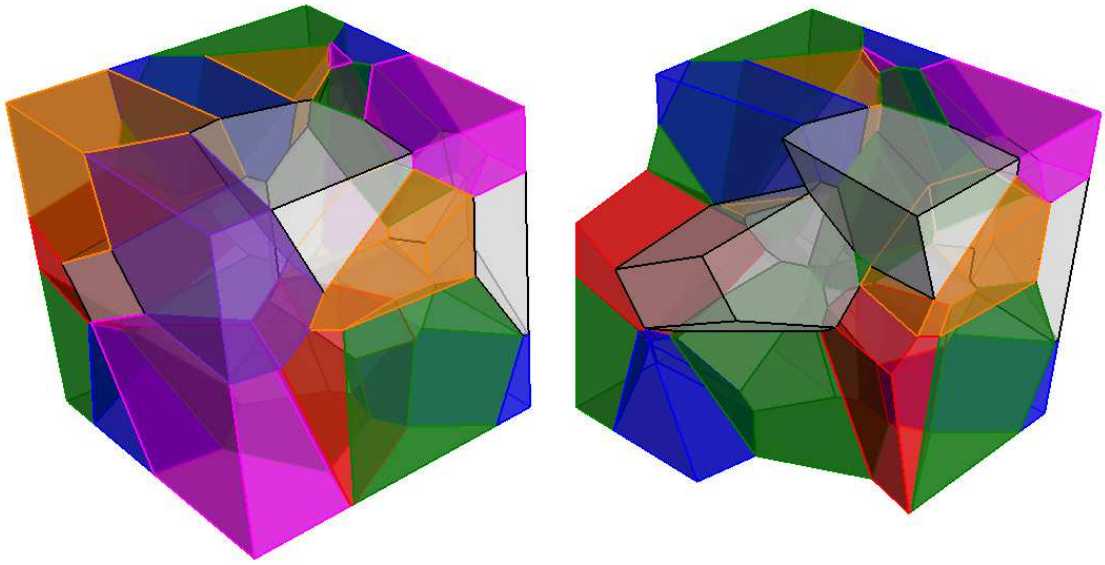
Základem algoritmů jsou jednoduché rovnice, z kterých je algoritmus poskládán do logické struktury. Většina algoritmů je větvených, na základě vstupních dat se rozhoduje, kterou cestou ve struktuře se proces výpočtu ubírá.

Architektura vytvořená pomocí algoritmické metody vzniká naprogramováním struktury, která řeší požadované úkoly. Čím je algoritmus obecnější, tím větší jsou jeho možnosti, na druhou stranu je i složitější

Výhodou metody je fakt, že na základě stejných vstupů vždy dostaneme stejný výsledek. Výsledek je generován automaticky.

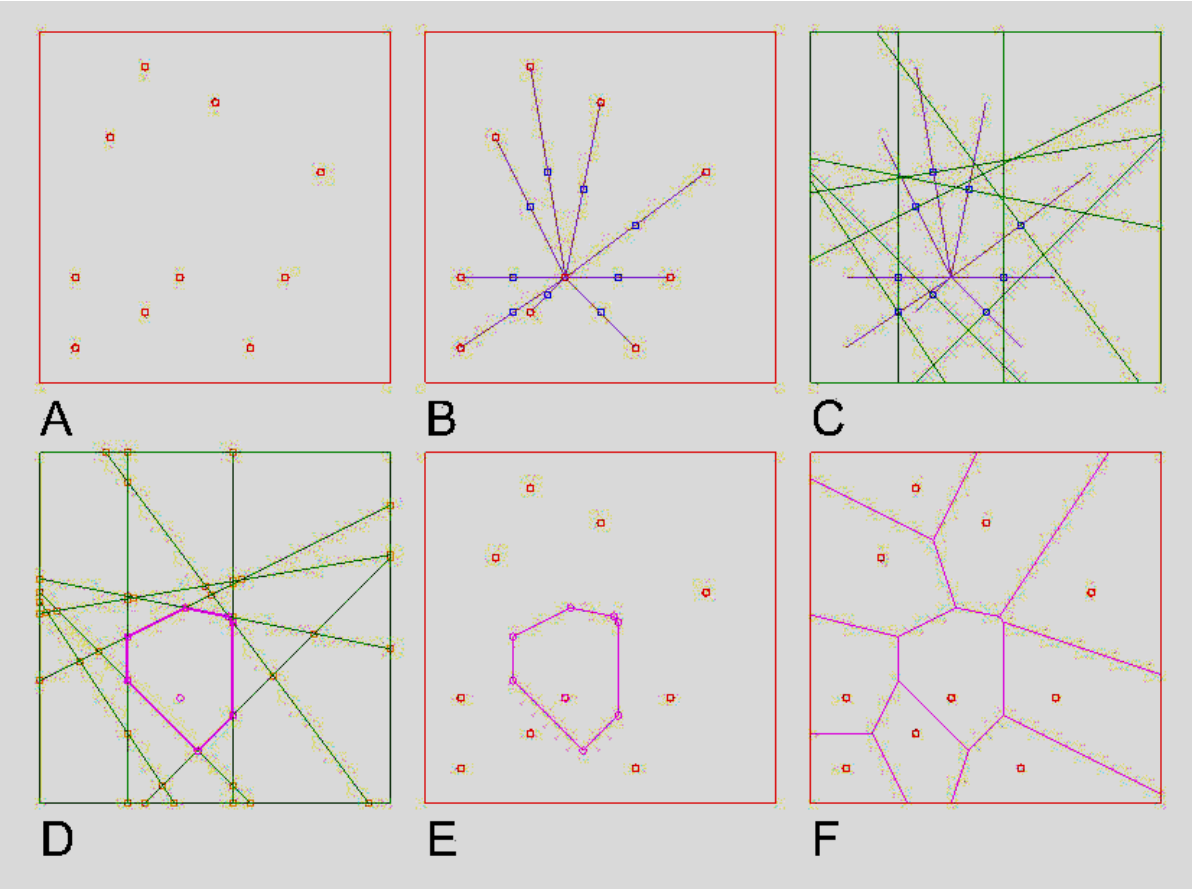
Nevýhodou metody je nutnost znalosti alespoň základních principů programování.

Pro příklad uvedu jeden zajímavý algoritmus, který má v architektuře spoustu možných využití – jde o dělení plochy (prostoru) na buňky v okolí definičních bodů a nazývá se Voronoi diagram.



obr. 52 Voronoi 3D

na obr. 52 je ukázka Voronoi diagramu ve 3D, Důležitou vlastností každé jedné buňky je neexistence nekonvexních úhlů



obr. 51 Voronoi 2D

Na schématech je vidět postup algoritmu – A) výběr deseti bodů v ohraničené ploše, B) spojnice od bodu 1 k ostatním, poloviny vzdáleností, C) kolmice na všechny spojnice v polovině vzdálenosti, D) kolnice jsou nasekány dle křížení na malé úsečky, výběr vnitřních segmentů okolo bodu, E) buňka pro první bod, F) opakujeme pro všechny body.

Vyvinutých algoritmů je nepřeberné množství, jejich rozboru je vyčleněna samostatná kapitola 2.5 - Algoritmy

2.4.7 Evoluční architektura

Evoluční architektura je založena na genetických (evolučních algoritmech). Proces návrhu je obdobou Darwinovy teorie o vývoji druhů, ale je aplikován na vývoj architektonických objektů.

Vývoj druhu (v případě architektury objektu) začíná od tzv. nulté populace, kterou ručně definujeme. V populaci máme jedince – objekty, u nichž si všímáme některých jejich vlastností. Poté definujeme základní transformace, které mohou tyto vlastnosti měnit. První generace obdobně jako v přírodě vzniká reprodukcí a křížením, populace se rozroste o nové jedince, jejichž vlastnosti jsou buď shodné s rodiči, nebo kombinací vlastností rodičů. U některých jedinců jsou vlastnosti náhodně lehce pozměněny – jedinci jsou mutováni. Na základě hodnocení vlastností (fitness) jsou vybráni jedinci, kteří nejvíce vyhovují danému účelu. Vybraní jedinci jsou určeni pro rozmnožování a stávají se rodiči další generace. Vývoj je ukončen buď dosažením požadovaného cíle, nebo je zastaven v n-té generaci a je vybrán finální nejvýhodnější jedinec.

Aplikace v architektuře je možná. Nastavení vlastností, mutačních transformací a hlavně hodnotícího systému je v případě složitých objektů s funkčními a mechanickými vlastnostmi velmi náročné.

Genetické algoritmy budou detailněji rozvedeny v kapitole 2.5.4

Problémy nastávají s názvoslovím, evoluční architekturou také často bývají nazývány objekty inspirované přírodou – lépe bionická architektura. Nemluvě o používání pojmu "evoluční architektura" v jiných oborech.

2.4.8 Genetické programování

Zcela zvláštní metodou je genetické programování. Genetické algoritmy nejsou aplikovány na samotné objekty, ale na jiné algoritmy. Znamená to, že je pomocí genetického algoritmu vyvíjen optimální algoritmus pro daný účel. Jinak řečeno jeden program sám napíše jiný program a ten potom můžeme opakovaně používat. Metoda je velmi blízká vývoji umělé inteligence, je používána pro autodidaktické programy.

V architektuře využít lze, zatím spíše na teoretické úrovni.

2.5 Algoritmy

Algoritmus je schematický postup pro řešení určitého druhu problémů, který je prováděn pomocí konečného množství přesně definovaných kroků.

Slovo algoritmus je odvozeno od perského matematika, žijícího v Bagdádu v 9.století الخوارزمي أبو عبد الله محمد بن موسى , v anglosaské transkripci /Abū ‘Abd Allāh Muhammad ibn Mūsā al-Chwārizmī/. (zdroj wikipedia, ve zdroji je arabské jméno napsáno pozpátku (أبو عبد الله محمد بن موسى الخوارزمي), v článku již upraveno dle arabské konvence psaní zprava doleva), který pomocí prvních algoritmů popsal základní pravidla algebry.

Obecných definic algoritmu je spousta, většinou poněkud nesrozumitelných, nebo prakticky nepoužitelných.

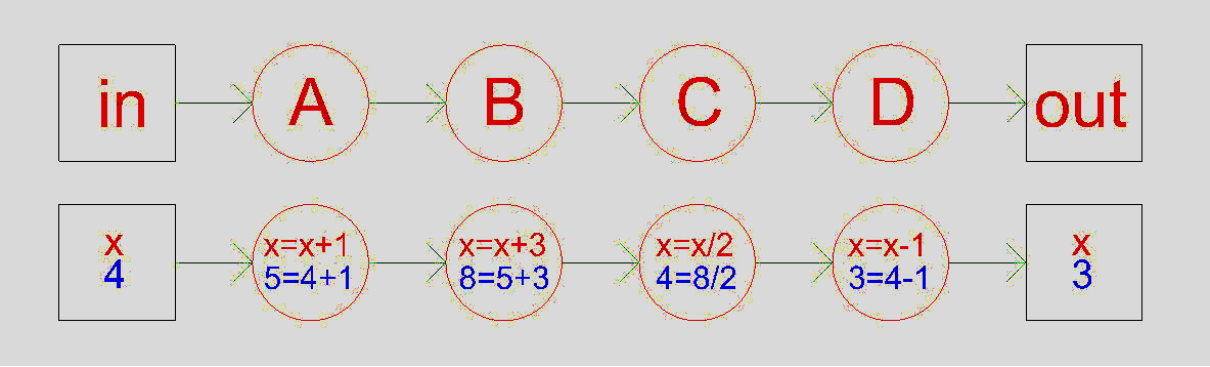
Osobně chápu algoritmus jako strukturu složenou z matematických operací, která na základě vstupních dat generuje požadované výsledky.

Pro zápisy algoritmů používáme programovací jazyky, algoritmus se tak stává jednoduchým programem. Při programování máme dvě verze zápisu – čistý – finální kód, kterým je program definován a funguje. Druhou variantou je pseudokód – zkrácená, přehledná verze, popis struktury algoritmu.

Typologické členění algoritmů je podřízeno účelu třídění, není tedy obecné, nýbrž poněkud subjektivní.

2.5.1 Jednoduché, deterministické algoritmy

Deterministický algoritmus je definován počtem řešení v každém kroku programu. Omezen je právě jedním jediným řešením. Schematicky jde tedy o přímou návaznost jedné funkce za druhou



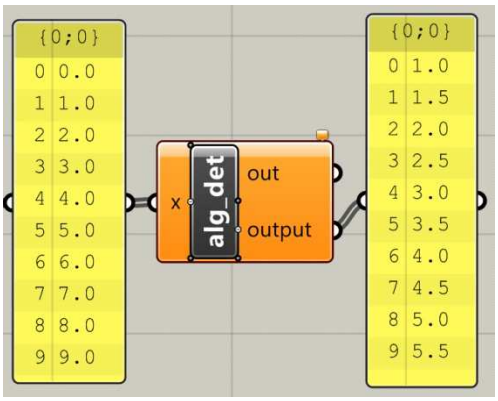
obr. 53 Deterministický algoritmus

Zápis v pseudokódu:

```
Pro x:
        x = fA(x)
        x = fB(x)
        x = fC(x)
        x = fD(x)
return x
```

Zápis v programovacím jazyce (VBscript)

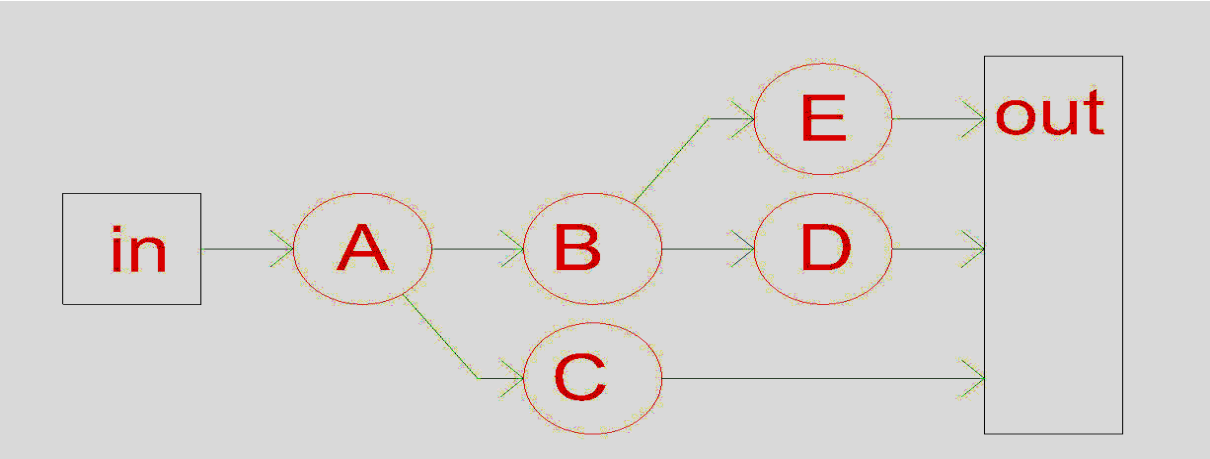
```
Function Det_alg(ByRef x As Double)        'definujeme funkce pro x
   x = x + 1                               'krok A
   x = x + 3                               'krok B
   x = x / 2                               'krok C
   x = x – 1                               'krok D
End Function

det_alg(x)                                 'použijeme funkci pro x
output = x                                 'přiřazení výsledku
```



obr. 54 aplikace deterministického algoritmu

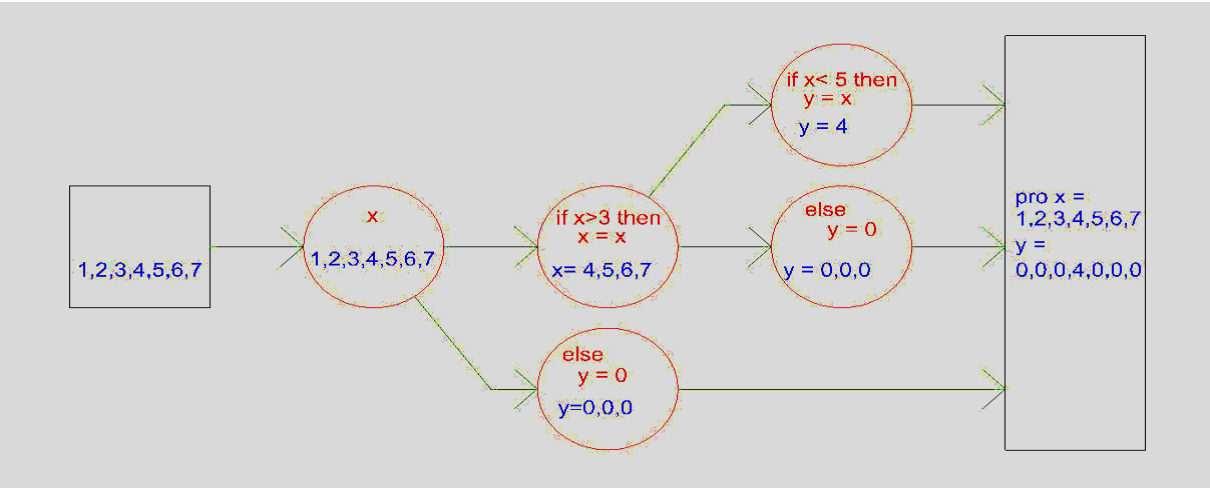
2.5.2 Větvené algoritmy

S jednoduchými algoritmy si většinou nevystačíme, prvním důvodem jsou výjimečné stavy v algebraických rovnicích (např. ošetření dělení nulou), dalšími důvody může být náš požadavek na použití rozdílných postupů pro různá zadání. Algoritmus pak má v určitých krocích různé typy výsledků a struktura se začíná komplikovat.



obr. 55 schéma větveného algoritmu

Nejjednodušším způsobem jak algoritmus rozvětvit je použití podmínek (if-then-else). Jestliže data A vyhovují podmínce, pak s nimi bude provedena operace B, v ostatních případech s nimi bude provedena operace C.



obr. 56 schéma algoritmu s průběhem hodnot

zápis kódu pro algoritmus na obr. 56

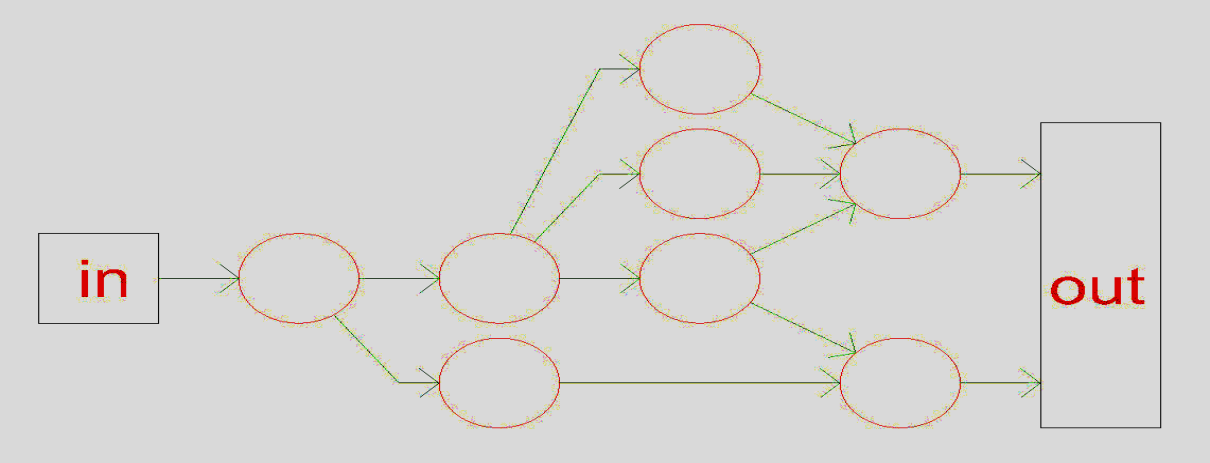
```
if x > 3 then
  if x < 5 then
    y = x
  else
    y = 0
  endif
else
  y  = 0
endif
```

Větvení je založeno na rozhodování o pravdivosti výroku (např. je li výrok x<3 pravdivý pak…). Větvení celé struktury pouze na základě takto jednoduchých výroků by bylo možné, struktura by ale byla zbytečně komplikovaná. Proto výroky skládáme dle pravidel výrokové logiky a Booleovské matematiky. Zavádíme jednoduché operace pro práci s pravdivostními hodnotami (true-false / 0-1). Základními operacemi jsou konjunkce, disjunkce a negace (and, or, not).

Nechci zde zabíhat do detailů logiky ani booleovské matematiky, uvedu pouze příklad zjednodušení předchozího algoritmu.

```
if (x > 3) and (x < 5) then        'operátor AND: platí li výrok (x>3) a zároveň platí výrok (x<5) pak…
  y = x
else
   y = 0
endif
```

Data, jejichž zpracovávání probíhá odděleně v jednom kroku, mohou být v dalším kroku zpracovávaná společně. Struktura se tak nejen větví ale i spojuje.



obr. 57 struktura větveného algoritmu

2.5.3 Rekurzivní algoritmy

Tento druh algoritmu je založen na vícenásobném opakování jedné procedury pro stejná data. Hodnoty jsou tak průběžně měněny, přepisovány. Používáme funkce, kde je přesně určen počet opakování, nebo funkci kde je opakování zastaveno až při dosažení stanovaného cíle.

Při definovaném počtu opakování může vypadat zápis následovně:

```
For i = 1 to 10
  x = x * 2
next
```

Pro x = 1 budou kroky vypadat následovně: x = 2, 4, 8, 16, 32, 64, 128 … 1024 po kroku 10.

Pro neurčitý počet opakování použijeme smyčku a ukončovací podmínku.

```
Do
  x = x * 2
Loop Until x > 1000
```

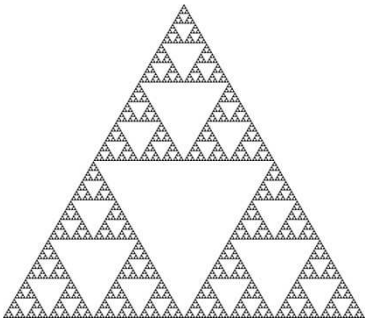
Pro x = 1 budou kroky stejné jako v předchozím případě, opakování ale bude zastaveno ve chvíli kdy x přesáhne hodnotu 1000, výsledek bude stejný: x = 1024.

U této varianty je třeba dát velký pozor na ošetření ukončovací podmínky. Např. chybná podmínka:

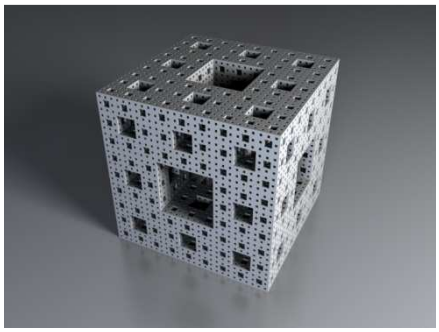
```
Do
  x = x * 2
Loop Until x = 1000
```

Má za následek opakování x = 2,4,8…512,1024, zde jsme ukončovací podmínku minuli, 2048, 4086…. Ze smyčky už se nedostaneme, program "zatuhne"

Rekurzivní algoritmy se používají zejména pro iterace a aproximaci. Zvláštním využitím je fraktálová geometrie, založená na postupném dělení, či rozvoji geometrických struktur.



obr. 58 Sierpinski triangle



obr. 59 Menger sponge

2.5.4 Genetické algoritmy

Genetické algoritmy napodobují systém vývoje biologických druhů. Používají k tomu stejných základních principů jako příroda, jen je matematicky vyjadřují a aplikují v jiném oboru.

Základem pro vývoj genetickým algoritmem je skupina jedinců – prvků. U každého jedince definujeme jeho vlastnost – geny. Prvotní skupina se nazývá generace 0 a je vytvořena ručně.

Jako příklad zde máme skupinu jedinců, jejichž osm vlastností je definováno binárně (nulami a jedničkami)

P1 (0,1,0,1,0,1,0,0)
P2 (1,0,0,1,0,1,0,0)
P3 (0,1,0,0,0,0,0,1)
P4 (0,0,1,0,0,1,0,0)

Aby dával algoritmus nějaký smysl, potřebujeme vědět čeho dosáhnout. V tomto příkladě jsou např. ideálním jedincem ten, který má největší množství vlastností rovno 1. U každého jedince budeme tedy hodnotit, do jaké míry vyhovuje. Budeme ho hodnotit tzv. fitness funkcí.

P0,1 (0,1,0,1,0,1,0,0) fitness: 3
P0,2 (1,0,0,1,0,1,0,0) fitness: 3
P0,3 (0,1,0,0,0,0,0,1) fitness: 2
P0,4 (0,0,1,0,0,1,0,0) fitness: 2

Nyní již jsme schopni prvního evolučního kroku – selekce. Prostě vybereme několik jedinců, kteří přežijí, tentokrát to budou ti s největší hodnotou. Stanovení správné fitness funkce je nejdůležitější, ale také nejobtížnější část algoritmu.

P0,1 (0,1,0,1,0,1,0,0) fitness: 3
P0,2 (1,0,0,1,0,1,0,0) fitness: 3

Vybraní jedinci nám budou sloužit jako rodiče pro další generaci. Zde je více způsobů jak se mohou geny z rodičů na potomky předávat

P1,1 (0,1,0,1,0,1,0,0) reprodukce (kopie jedince P0,1)
P1,2 (0,0,0,1,0,1,0,0) náhodný výběr z genů rodičů (křížení P0,1 x P0,2)
můžeme vytvořit i všechny kombinace, postačí několik
P1,3 (1,0,0,1,0,1,0,0)
P1,4 (1,1,0,1,0,1,0,0)

Nyní jsme stvořili první generaci, proces můžeme opakovat.
Po několika generacích skončíme na skupině jedinců:

P3,1 (1,1,0,1,0,1,0,0)
P3,2 (1,1,0,1,0,1,0,0)
P3,3 (1,1,0,1,0,1,0,0)
P3,4 (1,1,0,1,0,1,0,0)

Zde by se vývoj zastavil při fitness = 4, tušíme určité rezervy.
Abychom zabránili vývojové stagnaci, zavedeme do vývoje vedle reprodukce a křížení třetí proces – mutaci. U některých jedinců náhodně občas nějaký gen změníme.

P3,1 (1,1,0,1,1,1,0,0)
P3,2 (1,0,0,1,0,1,0,0)
P3,3 (1,1,0,1,0,1,0,0)
P3,4 (1,1,0,1,0,1,0,1)

To nám zajistí přístup i ke genům, které původní populace vůbec neměla. V další generaci se tyto nové geny projeví, a budou-li schopny přežití, pak celu populaci obohatí.

Na závěr po několika dalších generacích dojdeme k jedinci, který zamoří celou populaci (mutace jen zhorší vlastnosti, křížení je již nezlepšuje).

Pn,1 (1,1,1,1,1,1,1,1) fitness 8
Pn,1 (1,1,1,1,1,1,1,1) fitness 8
Pn,1 (1,1,1,1,1,1,1,1) fitness 8
Pn,1 (1,1,1,0,1,1,1,1) fitness 7 mutant (má horší vlastnosti, nebude vybrán)

My zde na jednoduchém případě vidíme, že to je jedinec zcela ideální. U složitějších příkladů takového hodnocení schopni nejsme (kdybychom znali řešení, tak nepotřebujeme algoritmy), konec vývoje buď stanovíme sami omezeným počtem generací, nebo ukončovacím algoritmem, který hodnotí, zdali ještě dochází v populaci k vývoji.

Rychlost genetického algoritmu je dána velikostí populace – nastavením selekce, sílou – množstvím mutací a hlavně správnou definicí fitness funkce.

2.6 Datové struktury a vazby

Pro práci s algoritmy potřebujeme uchovávat data, se kterými pracujeme v organizovaných strukturách. Data při práci s počítačovou grafikou se většinou skládají ze složky polohové (souřadnice v prostoru x, y, z), struktury objektu a vazeb mezi objekty.

Nejjednodušší je bod, u něhož nám stačí souřadnice (x, y, z). U dalších objektů, například u úsečky definované dvěma body, potřebujeme mít definovanou datovou strukturu úsečky: Line (Pt1, Pt2), a souřadnice dvou bodů: Pt1 (x,y,z) a Pt2 (x,y,z). U složitých objektů může být struktura velmi rozsáhlá. Systém pak funguje tak, že se složitější objekty odkazují na jednodušší – úsečka se odkazuje na bod. Při modifikaci objektu je zase potřebný odkaz opačný – pohneme li bodem, bod se odkáže na úsečku a ta se změní. Takto mohou být objekty strukturovány hierarchicky.

Existují ale i vazby mezi objekty navzájem, potřebujeme tedy datové struktury i pro tyto vazby.

Základním problémem při návrhu softwaru je vymyslet datové struktury tak, aby s nimi šlo co nejlépe zacházet jak z hlediska uživatelského (rychlost, komplexita), tak z hlediska programátorského – při psaní programu (přehlednost, jednoduchost). Oba požadavky se často vzájemně popírají. Některé programy jsou sice "blbé"

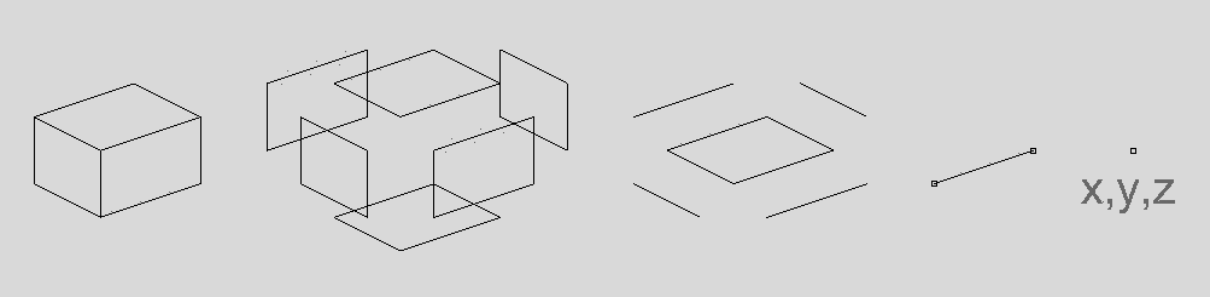
a občas nefungují, ale byly napsány rychle a levně, jiné programy jsou zase rychlé a všehoschopné, pak ale ještě nebudou dokončené.

Pro účely dalšího vývoje nastíním několik struktur vazeb mezi objekty a datových úložišť pro práci s nimi.

2.6.1 Hierarchické vztahy

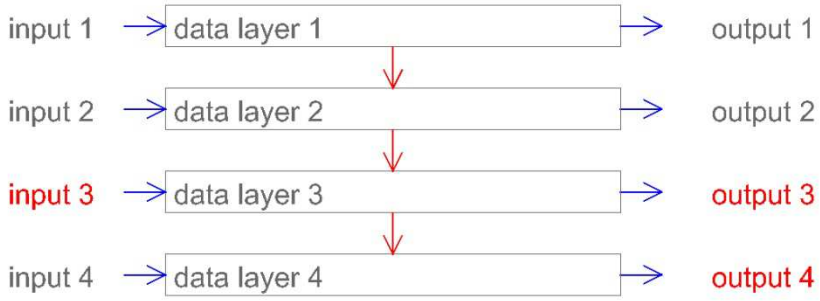
2.6.1.1. Vazby shora dolů, kreativní

Základními vazbami ve struktuře jsou vazby objektů nadřazených – složitějších na objekty podřízené, jednodušší. Jako příklad dejme objekt kvádr. Při jeho vzniku je generována datová struktura a její naplnění daty. Kvádr je tvořen šesti plochami, každá plocha čtyřmi hranami (úsečkami), každá úsečka dvěma body, každý bod je definován třemi souřadnicemi v prostoru. Vytvoříme li tedy objekt "kvádr", tak to znamená vytvořit strukturu o pěti úrovních naplněnou (1 + 6 + 6*4 + 6*4*2 + 6*4*2*3) informacemi a vazbami.



obr. 60 struktura kvádru

V diagramu pak vypadá struktura asi takto:



obr. 61 vazby shora dolů

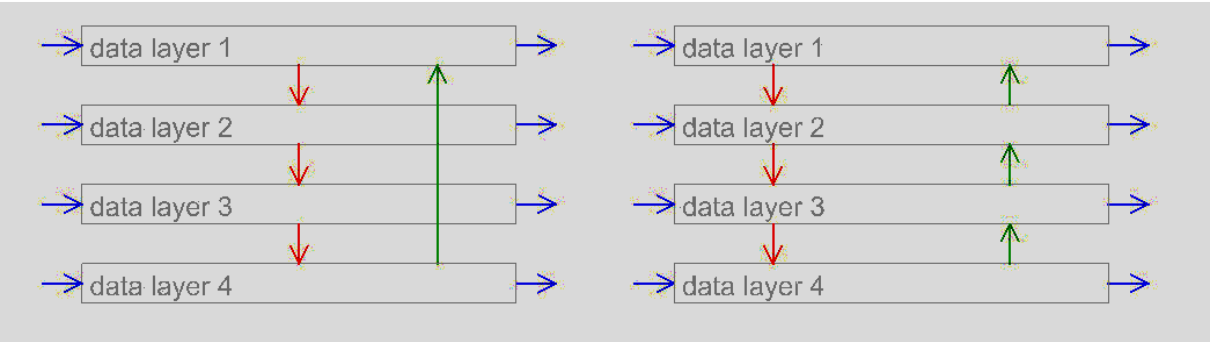
S takto vytvořeným kvádrem mohu manipulovat jako s celkem, změním li její základní parametry /input 1/ – např. místo osazení do prostoru, tak s ním pohnu. Transformace se projeví shora dolů na všechny prvky. Mohu provádět základní geometrické operace (posun, rotaci, zvětšení…). Datová struktura může zůstat zachována, informace v ní se nahradí novými, transformovanými.

Problém nastává, změním-li data na nižší úrovni, například posunu stěnou1. V diagramu (obr. 61) červený vstup. Změní se pouze data na nižší úrovni a to jen ve větvi pod stěnou1. Výsledek vypadá tak, jako bychom kvádr otevřeli. Při další manipulaci s kvádrem záleží na tom, jak je zapsán algoritmus pro transformace. Buď je transformován každý objekt podle stejné transformace jako kvádr, pak třeba otevřeným kvádrem posuneme a nic zásadního se nestane, nebo jsou data nižších struktur vymazána a nahrazena novými po transformaci aplikované na první úrovni, pak se nám posunutý kvádr zase zavře. Práci s otevíráním můžeme považovat za nenávratně ztracenou. Pomíjím to, že otevřená kvádr ztrácí vlastnosti uzavřených objektů.

2.6.1.2. Vazby zdola nahoru, zpětné, modifikační

Na základě problému, nastíněném v předchozí kapitole, vzniká řešení v podobě nových vazeb.

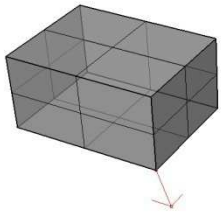
Struktura dat je doplněna o zpětné vazby od jednodušších objektů na objekty složitější, na vyšší úrovni. A to buď částečně, skokem nebo systematicky, postupně.

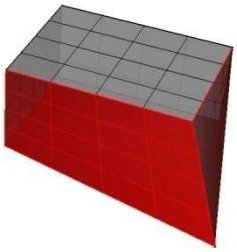


obr. 62 vazby zpětné

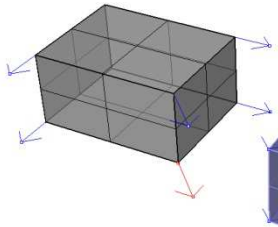
Díky těmto vazbám můžeme změnit objekt na libovolné úrovni a celý objekt se naší změně přizpůsobí. Posunu-li tedy nyní jednu stěnu kvádru, celý kvádr se o tento posun protáhne. Základní vlastnosti zůstanou zachovány.

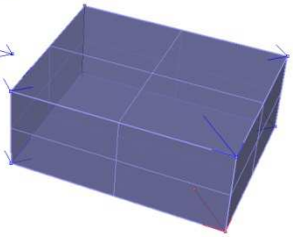
Nic není tak jednoduché, jak na první pohled vypadá. Posunu li jeden vrchol kvádru, tak už to nebude kvádr – tři stěny budou zborcenými plochami, stěny nebudou tvořit obdélníky…, ale jiný objekt. Ve druhém případě se posun zpětnou vazbou projeví přímo do základní struktury kvádru. Na základě složitějšího algoritmu se posune se více bodů, hran, stěn v různých směrech tak, aby byla zachována definice kvádru (např. pravoúhlost). Tento přístup vyžaduje, aby součástí definice objektu byly i jeho lokální transformace – reakce na modifikace na nižších úrovních. U složitých objektů se může zdánlivě jednoduchá operace (např. posun jednoho bodu) projevit markantními nechtěnými změnami v celé geometrii.





obr. 63 modifikace postupná

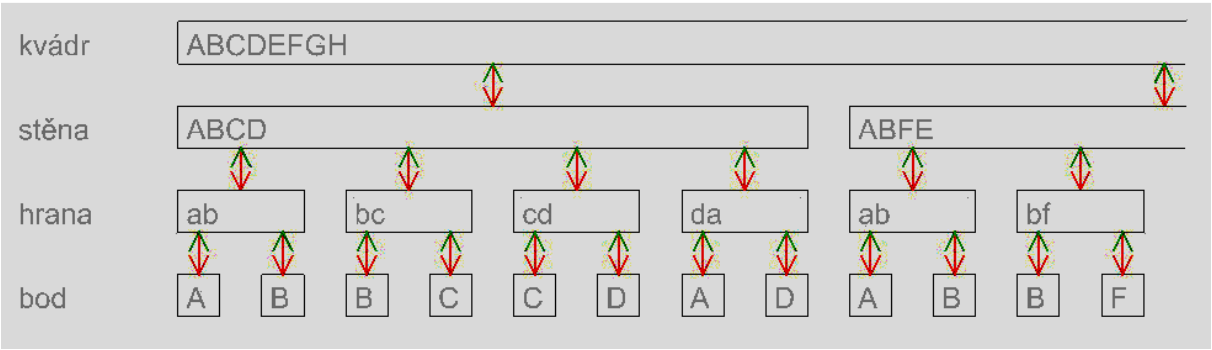




obr. 64 modifikace celku

2.6.1.3. Vazby diagonální

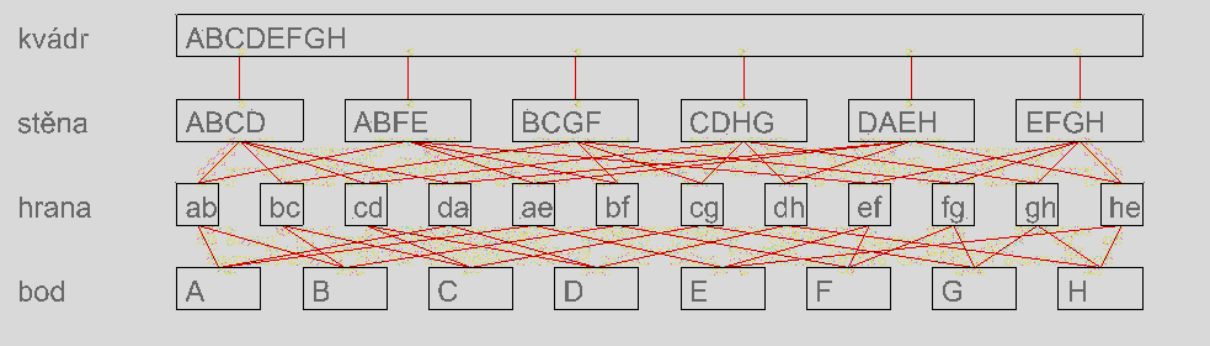
Na předchozích schématech byly uvedeny příklady vazeb čistě vertikálních, data jsou ale ve většině případů, jak už bylo zřejmé z příkladu kvádru, strukturována i horizontálně. Vedle sebe na stejné úrovni stojí šest stěn kvádru. Schéma se nám díky tomu větví do stromové struktury.



obr. 65 stromová struktura vazeb

Tato struktura je velmi přehledná, není ale vhodná ze dvou důvodů. V první řadě je nesmyslně velká, ve druhé zde máme jeden prvek zastoupen v mnoha vydáních. Struktura zbytečně zabírá datový prostor, manipulace s objekty bude pomalejší.

Z toho důvodu můžeme definovat vztahy křížem mezi prvky. Hierarchie zůstává zachována, struktura je optimalizovaná na minimum.



obr. 66 diagonální struktura vazeb

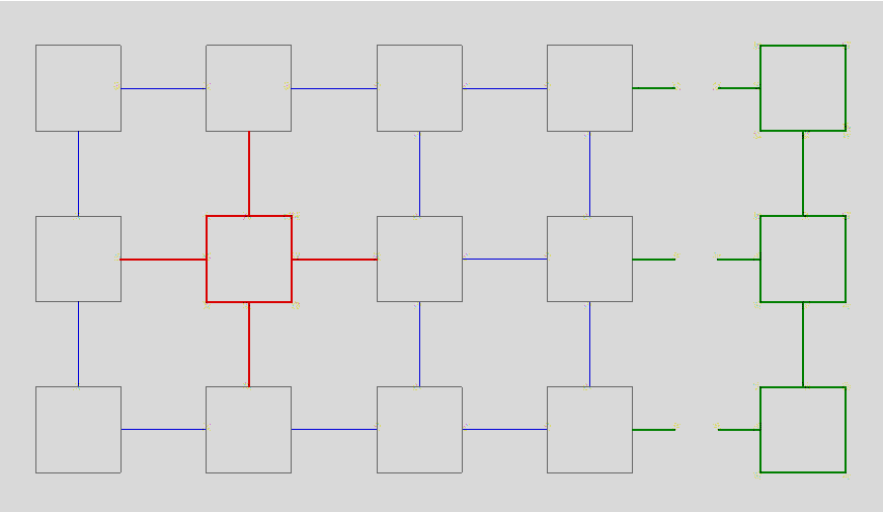
Na rozdíl od obr. 65, kde je pouze čtvrtina celé struktury kvádru, na obr. 66 je struktura stejného objektu celá, optimalizovaná. Počet prvků ve struktuře je výrazně nižší, počet vazeb zůstává stejný.

2.6.2 Struktury s horizontálními vztahy, sousednost

Existují objekty a algoritmy, u nichž nám systém hierarchických – vertikálních vazeb nestačí, řeší se zde vztahy mezi prvky na stejné úrovni – tzv. sousednosti.

2.6.2.1 Pevné vazby

Nejjednodušší systém sousednosti je vázaný, pevný. Používá se při primární genezi objektu, tam kde je dopředu zřejmá geometrie a struktura objektu. Typickým příkladem je rovnoměrná síť složená z trojúhelníků nebo čtverců. Ve chvíli, kdy ji tvoříme, máme jasnou představu o tom který čtverec (nebo trojúhelník) bude ležet vedle druhého. Vazby sousednosti je tedy možné zapsat do jednoznačné, pevné struktury.



obr. 67 sousednost – pevné vazby

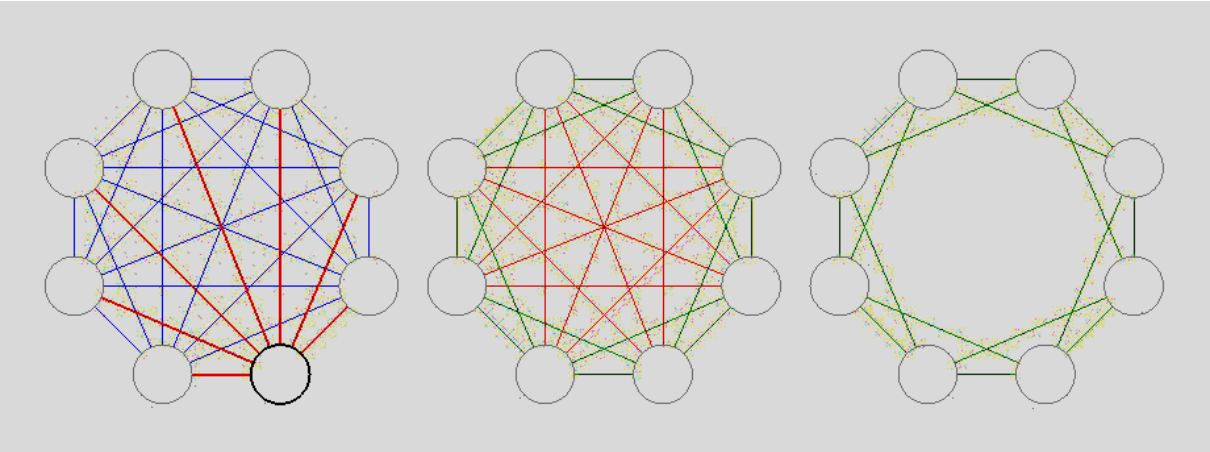
Na obr. 67 jsou vyznačeny vazby sousednosti mezi prvky v ortogonální struktuře. Výhodou tohoto systému je malé množství vazeb (červeně vyznačeny vazby pro jeden objekt, modře všechny /ostatní/ vazby ve struktuře).

Zásadní problém nastává ve chvíli, kdy chceme do takové struktury zasahovat. Zeleně jsou vyznačeny prvky, které by měly být do struktury přidány. Potřebujeme k tomu složitý vyhledávací algoritmus, který najde k přidávaným prvkům prvky z původní struktury v sousedském vztahu, poté vytvoří nové pozice ve struktuře, nové vazby. Problém je ještě složitější, pokusíme-li se spojit dvě již existující struktury do jedné. Požadavky na změny uvnitř struktury jsou obtížně řešitelné.

2.6.2.2 Volné vazby

V případě, že nejsme schopni dopředu definovat vztahy sousednosti, například protože ještě neznáme geometrii objektu, nebo je objekt již vytvořený bez vazeb a my chceme vazby doplnit, pak použijeme vazby volné.

Vazby jsou definovány na základě vyhledávacího a hodnotícího algoritmu. Pro danou skupinu prvků vytvoříme všechny možné vazby, každý prvek propojíme se všemi ostatními. Takto vyhledané vazby zhodnotíme podle definice vazby a nevyhovující vyřadíme.



obr. 68 sousednost – volné vazby

Na obr. 68 vidíme vlevo počet všech vyhledaných vazeb mezi osmi prvky struktury. Počet vazeb je (n * (n-1)), v tomto případě 56, s rostoucím počtem prvků se počet vazeb exponenciálně zvyšuje! Pro jeden prvek jsou vazby vyznačeny červeně.

Na prostředním diagramu vidíme hodnocení sousedských vztahů (zde podle vzdálenosti). Červeně jsou vyznačeny vztahy vyloučené, zeleně ponechané.

Na pravém obrázku výsledek.

2.6.2.3 Selektivní vazby

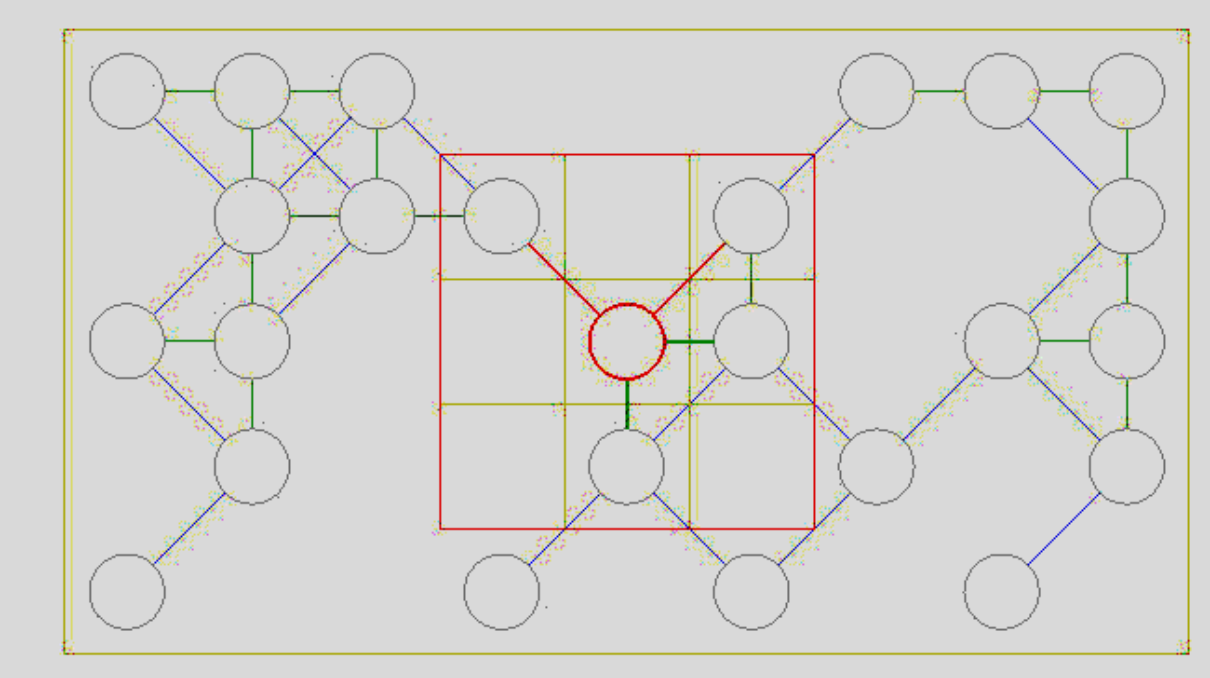
Účinnost systému volných vazeb je ztrácí s přibývajícím počtem řešených prvků. U komplikovaných struktur, kde není neobvyklé řešit najednou až stovky tisíc prvků, narůstá množství řešených vztahů do miliard a výše. V případě, že je algoritmus založený na volných vazbách volán opakovaně, dochází ke zpomalení výpočtu, přetížení paměti.

Tento problém je možné řešit pomocí roztřídění prvků do menších skupin a vztahy pak hodnotit uvnitř těchto skupin, nikoli v rámci celku. Tím omezíme exponenciální nárůst množství vazeb s počtem prvků, nárůst bude víceméně lineární.

Třídění do menších skupin je závislé na definici hodnocení – selekce vazeb. Uvedu jednoduchý příklad.

- Vazby jsou řešeny mezi skupinou bodů náhodně rozmístěných v ploše.
- Vazba je vyhodnocena jako relevantní, když je vzdálenost mezi dvěma body menší než …

1) Hodnocení vazby je na základě vzájemné polohy, body tedy budou tříděny na základě polohy v ploše.
2) Body musí být rozděleny do skupin tak, aby už velikost skupiny nevyloučila případnou relevantní vazbu.



obr. 69 sousednost – selektivní vazby

3) Na obr. 69 vidíme příklad, kde je v každé skupině – buňce – jeden bod (může jich být více). Velikost Buňky je stanovena v závislosti na maximální velikosti vazby.
4) Pro každý řešený bod najdeme buňku, do níž náleží a buňky nejblíže sousedící, tím nám vznikne omezená skupina bodů, u nichž řešíme vazby. Na obr. 69 je řešený bod vyznačen červeně, skupina devíti buněk vyznačena červeným čtvercem.
5) Řešíme vztahy ke čtyřem bodům, dvě vazby jsou vyloučeny (body jsou příliš vzdálené - červeně), dvě vazby jsou relevantní (zeleně).
6) Opakujeme pro každý bod postup 4-5. Na obrázku jsou znázorněny všechny řešené vazby, modře pak vyloučené, zeleně relevantní.

Zhodnocení selektivního algoritmu pro uvedený příklad:

Při použití volných vazeb bez třídění musíme pro 25 bodů vyhodnotit 600 potenciálních vazeb.

Po zavedení třídění hodnotíme vazeb pouze 78.

Bohužel není tento algoritmus vždy takto účinný. Je li definice relevance vazby příliš široká, třídění nepomůže. Je-li relevantnost vazby definována pomocí příliš složitého vzorce, nemusíme být ani schopni prvky roztřídit.

2.6.3 Datové kontejnery

2.6.3.1 Definice

Veškeré algoritmy pracují se skupinami dat, které je potřeba mít přehledně uložené, roztříděné a dostupné. Proto zavádíme různé druhy datových kontejnerů.

Nejjednodušším kontejnerem je prostá definice jedné hodnoty. V programu můžeme definovat například číslo, pojmenovat ho a pak se na něj pomocí jeho jména odkazovat.

Příklad (VBscript):

```
Dim cisloX As integer        'vytvoření datového kontejneru pro jedno číslo, název cisloX
cisloX = 10                  'přiřazení hodnoty

Dim cilsoY As integer
cisloY = cilsoX + 2          'následný odkaz na data v kontejneru cisloX
```

2.6.3.2 Seznam

Jednoduchým kontejnerem pro skupinu dat je pole, prostý seznam. Nadefinujeme seznam položek, naplníme jej daty, posléze z něj data můžeme čerpat, upravovat je atd.

```
Dim SEZNAM  As New List (Of Integer)        'vytvoření seznamu dat(čísel)

For i = 1 to 5                              'naplnění seznamu čísly 1-5
  SEZNAM.add(i)
Next

cisloX = SEZNAM(3)                          'vyhledání čtvrté položky v seznamu
SEZNAM(0) = 10                              'změna hodnoty na prvním místě v seznamu
```

Data v seznamu pak vypadají následovně:

SEZNAM:	
0:	10
1:	2
2:	3
3:	4
4:	5

2.6.3.3 Pole vícerozměrné

V mnoha případech nám nestačí jednoduchý seznam – jednorozměrné pole. Často je nutné mít u jedné položky více hodnot. Například seznam polohy bodů, kde potřebuje pro každý bod hodnoty x,y,z – tedy dvourozměrné pole.

POLE:			
	X	Y	Z
0	2	1	6
1	6	2	8
2	3	3	3
3	5	6	9

Z tahového pole poté čerpáme informace obdobně, jako z jednoduchého seznamu

```
SOURADNICE_Y_BODU2 = POLE(2)(1)
```

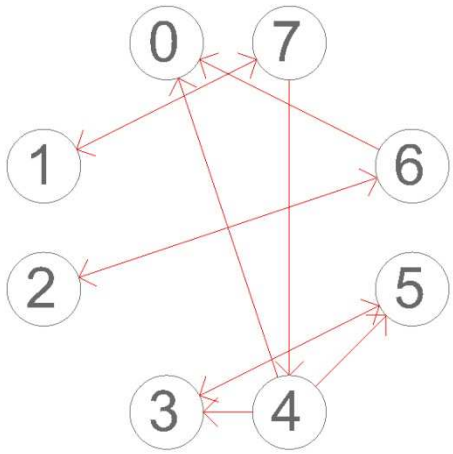
Množství rozměrů pole není omezené, omezené jsou spíše lidské schopnosti, jak se v mnohorozměrných polích vyznat.

2.6.3.4 Asociativní paměť

Pracujeme li s neurčitým počtem prvků nebo s množstvím nekoherentních dat a potřebujeme mezi nimi uchovávat vzájemné vztahy, průběžně je měnit a upravovat, máme možnost využít schéma asociativní paměti.

Data jsou uchovávána v jediném seznamu, třídění není nijak důležité, každá položka je v seznamu vedena pod svým číslem (indexem) a je tak snadno dohledatelná. Nové položky jsou přiřazeny na konec seznamu, indexy se tak nemusí měnit.

Existuje více způsobů jak vztahy mezi položkami zapisovat.



obr. 70 asociativní paměť - příklad

A) Seznam vztahů

Je vytvořen seznam, ve kterém jsou u každého prvku uvedeny vazby na ostatní.

0: ()
1: ()
2: (6)
3: (5)
4: (0, 3, 5)
5: (3)
6: (0,2)
7: (4)

Seznam je jednoduchý, nenáročný na prostor. Je trochu složitější do seznamu vztahy přidávat (kontrola duplikací) a vztahy odebírat (vztah, který má být odebrán, musí být nejdříve vyhledán, seznam nahrazen novým bez vyhledaného vztahu). Výhodou seznamů je možnost uchování pořadí vazeb

B) Binární pole

Je vytvořeno dvourozměrné pole (n*n prvků), v tomto poli jsou zachyceny všechny potenciální možnosti vazeb mezi prvky. Existence vazby je pak definována binárně (0/1 – je/není).

	0,1,2,3,4,5,6,7
0:	(0,0,0,0,0,0,0,0)
1:	(0,0,0,0,0,0,0,0)
2:	(0,0,0,0,0,0,1,0)
3:	(0,0,0,0,0,1,0,0)
4:	(1,0,0,1,0,1,0,0)
5:	(0,0,0,1,0,0,0,0)
6:	(1,0,1,0,0,0,0,0)
7:	(0,0,0,0,1,0,0,0)

Orientace v poli je nesmírně přehledná, přidání a odebrání prvku, přidání vazby i zrušení vazby jsou velmi jednoduché operace. Při přidání prvku jednoduše připíšeme jeden řádek a jeden sloupec. Přidání vazby znamená změnit v poli hodnotu na 1, odebrání na 0. Duplicity nehrozí.

Nevýhodou je velikost pole, které exponenciálně narůstá s počtem prvků. Dále není ve struktuře zachyceno pořadí vazeb.

C) Vícevrstvé pole

V případě, že nám předchozí systémy nevyhovují, použijeme složitější systém třídění a uchovávání vazeb.

Binární informaci můžeme nahradit informací složitější (např. typem vazby).

	0,1,2,3,4,5,6,7
0:	(0,0,0,0,0,0,0,0)
1:	(0,0,0,0,0,0,0,0)
2:	(0,0,0,0,0,0,A,0)
3:	(0,0,0,0,0,A,0,0)
4:	(B,0,0,C,0,A,0,0)
5:	(0,0,0,A,0,0,0,0)
6:	(C,0,B,0,0,0,0,0)
7:	(0,0,0,0,A,0,0,0)

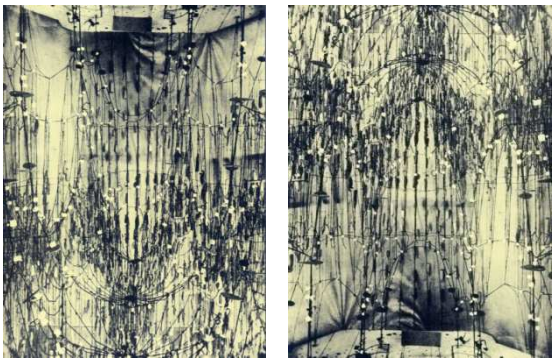
Zvýší se tak efektivita pole, výrazně také narostou požadavky na paměť a rychlost.

Může nastat situace, kdy potřebujeme definovat mezi dvěma prvky více vazeb a nechceme je mít odděleně, pak můžeme dvourozměrné pole rozšířit na trojrozměrné a metodu ve článku A) i B) zkombinovat.

Systémů lze vymyslet neomezené množství, nikdy není snadné dopředu zvolit ten správný pro daný úkol.

3 Analýza projektů

3.1 Antoni Gaudí

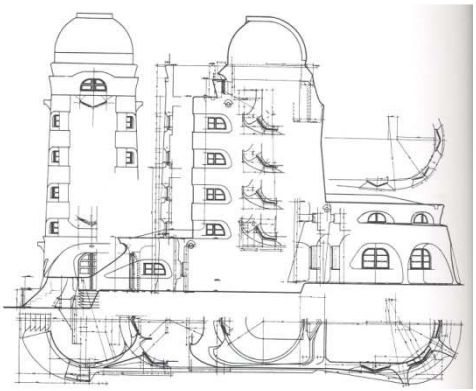


obr. 71 Gaudí – statické schéma

Jako první příklad uvádím dílo Antoni Gaudího z přelomu 19. A 20. století. Z hlediska metodiky je průkopníkem, který definoval své návrhy na základě výpočtů budoucí geometrie. Své výpočty prováděl mechanicky, nikoli číselně – viz obrázek 71. Trojrozměrná zavěšená statická schémata jsou předobrazem budoucích staveb. Principy, které používal, se v pozměněné formě objevily až po dalším půl století s nástupem digitálních technologií.

3.2 Einsteinova věž, 1920-21

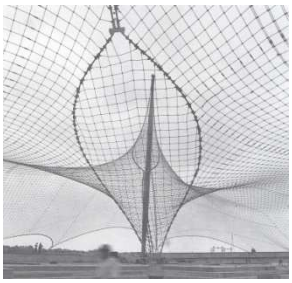
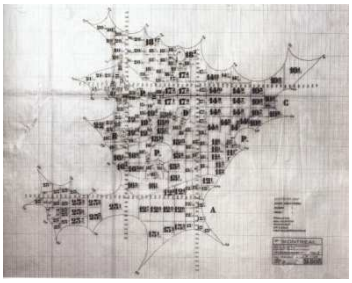
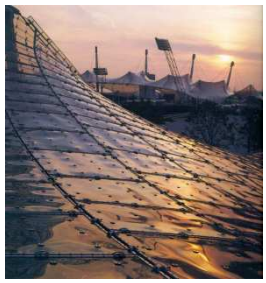
obr. 72 Einstenova věž



obr. 73 Einstenova věž - výkresy

Einsteinovu věž od Ericha Mendelsohna v Potsdami je vhodné zmínit jako jakési maximum geometrické uvolněnosti, které bylo před nástupem digitálních technologií dosažitelné a ještě přesně definovatelné. Na výkrese (obr. 73) vidíme, že zdánlivě volné křivky mají přesnou geometrickou a matematickou definici.

3.3 Frei Otto – stanové konstrukce


obr. 74,75 Německý pavilon, Expo '67, Montreal *obr. 76,77 Olympijský stan, Mnichov*

Jeden z prvních zásadních projevů nástupu digitálních technologií do procesu architektonického navrhování. Lanová konstrukce navržená Frei Ottem a Rolfem Gutbrodem pro Expo '67 v Montrealu a následně podobný princip realizovaný v Mnichově v letech 1968-72 v kooperaci Günter Behnish & Partner a Frei Otto.

Oba návrhy vycházejí z geometrie " *four-point-tents* " /cit. Architecture in Twentieth Century, Tashen, str.319/ , tedy parabolických hyperboloidů. Statické výpočty definující tvar konstrukce již byly prováděny digitálně, výkresová dokumentace je provedena ručně.

3.4 Nádraží Lyon – Satolas, 1989-94



obr. 78,79,80 Calatrava – Lyon-Satolas

Fragment stavby – betonový pilíř je zde prezentován jako příklad kombinace několika metod. Santiago Calatrava ve svých projektech jedinečným způsobem spojuje statické principy, konstrukci a uměleckou formu v nedílný celek. V případě pilíře na obr. 78-80 vidíme betonovou hmotu, která "obepíná" průběhy sil v pilíři.

U Calatravy je velmi obtížné zjistit, v které fázi procesu jsou využity digitální technologie. Ze skic je zřejmé, že představu o tvaru a průběhu sil v konstrukci má od

samého začátku, výsledné zpřesnění je již dílem výpočtu. Složitý tvar pilíře je vygenerován na základě průběhu sil, umělecky dopracován a znovu přesně definován křivkami 3°. Pilíř je monolitický, litý do plechových forem na místě.

3.5 Guggenheimovo muzeum, Bilbao, 1997



obr. 81,82,83 Gehry, Guggenheimovo muzeum, Bilbao

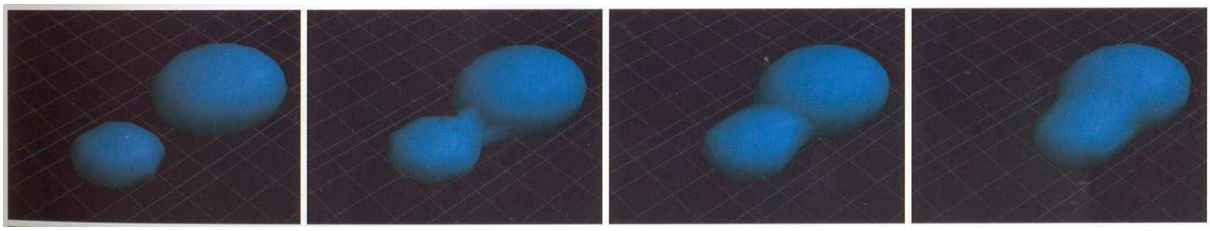
Frank Gehry bývá velmi často zmiňován jako průkopník digitálních forem architektury. Svébytným způsobem ve svých dílech propojil přístup sochařský a přístup digitální. Zakřivené objekty vznikají současně v reálném provedení ve formě hliněných (a jiných) modelů i ve 3D digitální formě. Nelze určit, co bylo první, přesná geometrická definice zakřivených ploch je ale důkazem, že finální forma návrhu je čistě digitální.

V procesu návrhu se tak objevuje hned několik metod, které dříve nebyly používány. Z reálných modelů jsou informace snímáním přenášeny do digitální formy, modely jsou vytvářeny na základě digitálních dat. Prostorově složité konstrukce jsou vyráběny na míru na základě 3D digitálních dat, montáž prvků na místě určována s pomocí GPS.

3.6 Výstavní pavilon BMW, Frankfurt, 1998 – 99

obr. 84 Bernhard Franken, Výstavní pavilon BMW, Frankfurt

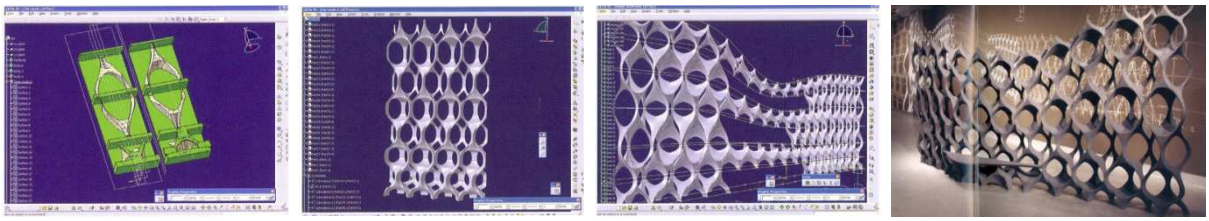
Tvar pavilonu je definován čistě matematicky, inspirací jsou dvě vodní kapky, které se působením gravitace spojují v jednu. Na simulaci je použit podobný algoritmus, jako byl popsán v kapitolách 2.4.5 Matematicky definovaná architektura, matabally, 2.4.5.2 Metaball.



obr. 85 Výstavní pavilon BMW – vývoj

Výsledný tvar je nařezán ortogonální sítí. Síť řezových křivek definuje nosnou konstrukci i dělení jednotlivých panelů. Kovové profily jsou řezány CNC obráběcími stroji, skla jsou lita do trojrozměrných vyfrézovaných forem.

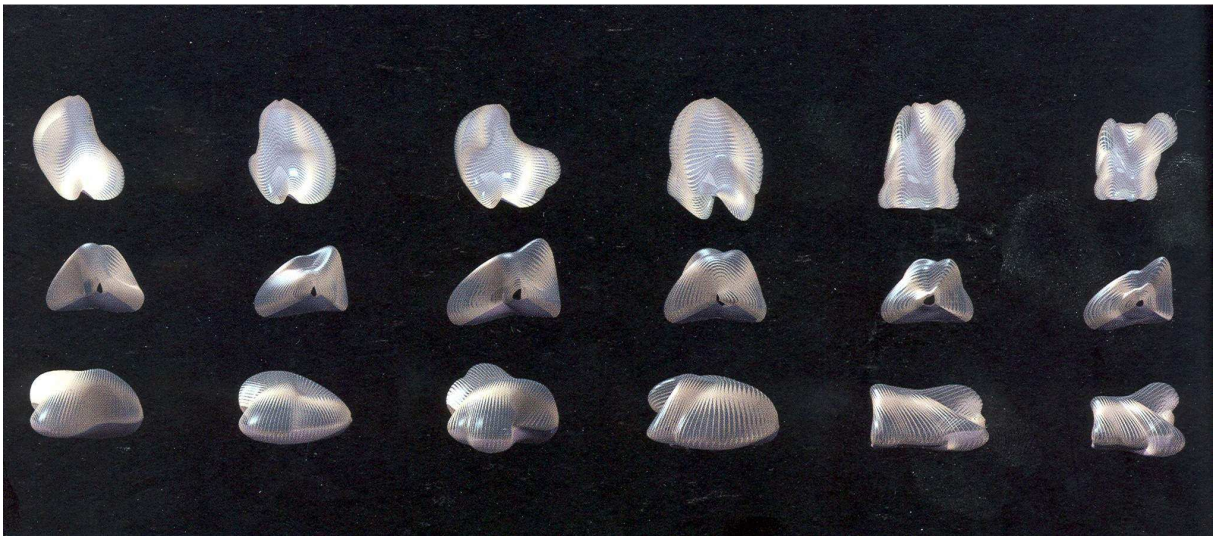
3.7 Bone Wall, 1999



obr. 86-89 Urban A & O, Bone Wall

Prostorová skulptura – kostěná zeď je vymodelována v programech Catia a Rhinoceros. Struktura se skládá z jednotlivých parametrických prvků (obr 88). Na obr. 89 vidíme kompozici prvků lineárně modifikovaných, dále pak model výsledného objektu, ve kterém je zřetelné, jak se každý prvek přizpůsobuje řídící síti. Na závěr instalace výsledného díla. Jednotlivé prvky byly vyřezány pětiosými CNC frézami a sestaveny na místě.

3.8 Embryological House, 1999

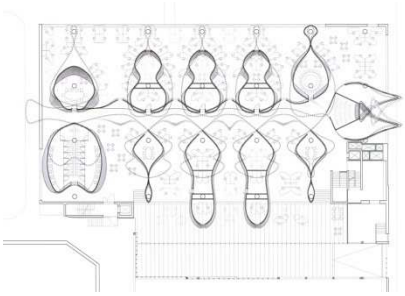


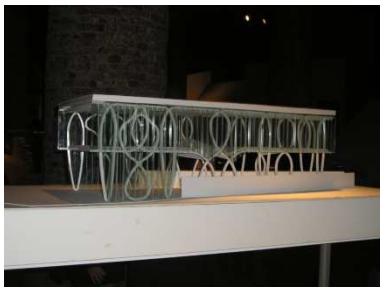
obr. 90 Greg Lynn, Embryological House

Greg Lynn v tomto pokusném projektu definoval budovu podobně jako živočicha. V počátečním stádiu je objekt definován jako struktura obalující jedinou funkci, podle potřeb se zárodek vyvíjí, pro jednotlivé funkce vznikají potřebné různorodé prostory a obalová struktura se jim přizpůsobuje. Na obr. 90 vidíme různě komplikované výsledky vývoje.

Projekt je založen na parametrickém designu, inspirace přírodou jej řadí mezi bionickou architekturu.

3.9 DADAGROWS, 2001





obr. 91 Greg Lynn FORM, DADAGROWS, Florencie

obr. 92,93 Biennale Venezia 2004

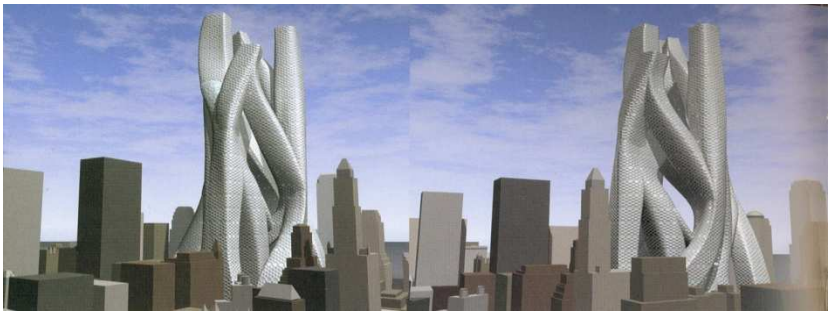
Projekt Dadagrows je založen na rozdělení kancelářského prostoru na pracovní a odpočinkové zóny. Oddělení je dosaženo zakřivenou zasmyčkovanou plochou.

Složitá plocha definující vnitřní prostory objektu vychází z teoretické práce – výzkumu chování křivek a zakřivených ploch. Tvoříme li křivku iterativně na základě

jejích požadovaných vlastností, pak se při požadavku minimálního poloměru vytvoří na křivce při určité konstelaci smyčka. Na obr. 91 vidíme výsledný půdorys kanceláře, jehož dominantním prvkem jsou právě tyto smyčky (řezy plochou). Na obr. 92,93 vidíme podobné přístupy prezentované na bienále v Benátkách v roce 2004.

Geometrií a automatickou genezí křivek se Greg Lynn dříve výrazně zabýval a výstupy prezentoval ve svých přenáškách. Jestli byly tvary v tomto projektu generovány pomocí algoritmů, nebo byl tvar plochy pouze inspirován výstupy teoretických prací nevím. Aktuální stav projektu není nikde detailně prezentován, zřejmě bylo od jeho realizace upuštěno.

3.10 Oblique WTC, 2001



obr. 94 Lars Spuybroek - NOX, oblique WTC, New York

obr. 95 Oblique WTC - vývoj

Projekt vznikl jako okamžitá reakce na 11. září. Geometrie objektu navazuje na práce Frei Otta, Paula Virilia a Clauda Parenta z šedesátých let. U výškových budov hledá vedle stávající vertikality i diagonalitu a její výhody.

Geometrie objektu je založena na chování tížných řetězovek. Na obr. 95 je zachycen vývoj, na počátku jednoduchý model z provázků, poté převedení do digitální formy. Chování provázků (pevnost, pružnost) je naprogramováno tak, aby odpovídalo realitě. Okolo provázků – os budovy- je následně vygenerován objem a plášť. Metoda není příliš odlišná od práce A. Gaudího o sto let dříve.

3.11 Autobusová zastávka "Amazing Whale Jaw", 2003

obr. 96 Amazing Whale Jaw, Maurice Nio, Hoofsddorp

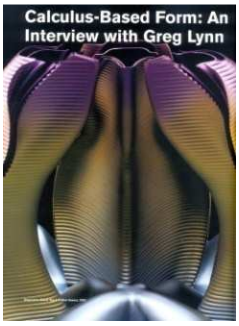
Maurice Nio vždy hledá nové možnosti a formy architektonického vyjádření. Před nemocnicí Spaarne vytvořil padesátimetrovou sochu, která slouží jako autobusová zastávka. Geometrie projektu není nijak digitálně generovaná, je čistě intuitivní. Zpracování digitální je. Zajímavá je technologie výroby. Objekt je po částech opracován z polystyrenových bloků CNC frézami na základě vložených digitálních dat. Povrch potažen polyesterem a bohužel po několika letech provozu nabarven.

Autor svůj projekt definuje takto:

"People often wonder about the building's shape and what it represents, and there are a number of possible answers. A correct answer in architectural terms is that it can be viewed as a large boulder that has been worn away by footsteps and sight lines. A correct answer in philosophical terms is that it can be regarded as a form that has not been pored over but which simply allowed itself to be discovered. A correct answer in terms of the designing process is that it can also be explained as a product of this process, which in this case was somewhat intuitive because of the unfamiliar technical terrain in which everyone had to operate. All these answers are simultaneously correct and irrelevant. Like the white face of a geisha, every opinion and image can be projected onto the building and it has no answers of its own." /zdroj: www.nio.nl/

… z hlediska procesu navrhování může být také vysvětlen objekt jako výstup procesu, který byl poněkud intuitivní díky tomu, že jsme pracovali v neprozkoumaném neznámém technickém prostředí…

3.12 Alessi Tea & Coffee Towers, 2003

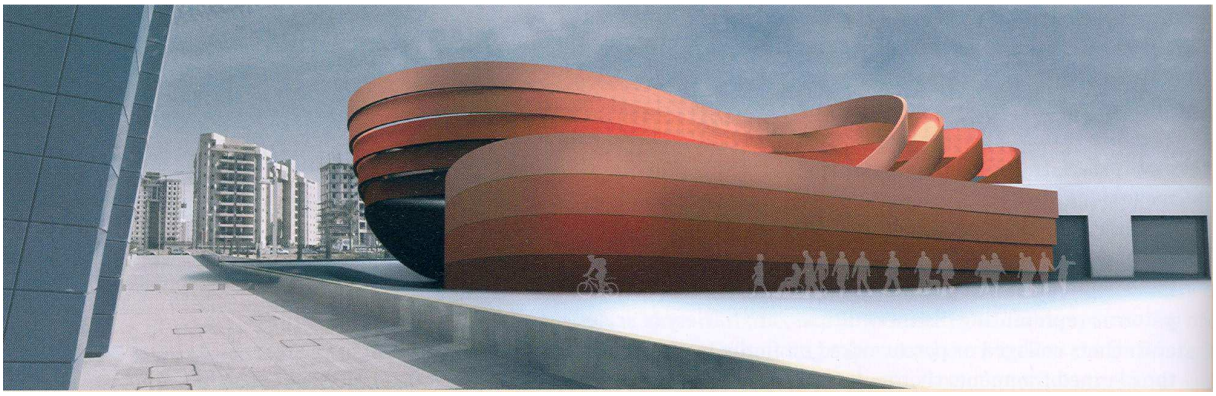


obr. 97 GregLynn, Alessi Tea & Coffee towers

Sestava kávového a čajového servisu, atypický design v kolekci 99 kusů pro firmu Alessi.

Zakřivené plochy jsou definovány parametricky včetně barevnosti. Plochy vylisovány z titanu, Každý set je unikátní, ale princip výroby pro všechny kusy stejný. Greg Lynn na tomto příkladu vysvětluje vývoj a vliv matematiky na formu v designu a architektuře. Přednáška je aktuálně ke zhlédnutí na:
http://www.ted.com/talks/greg_lynn_on_organic_design.html

3.13 Holon Design Museum, 2003

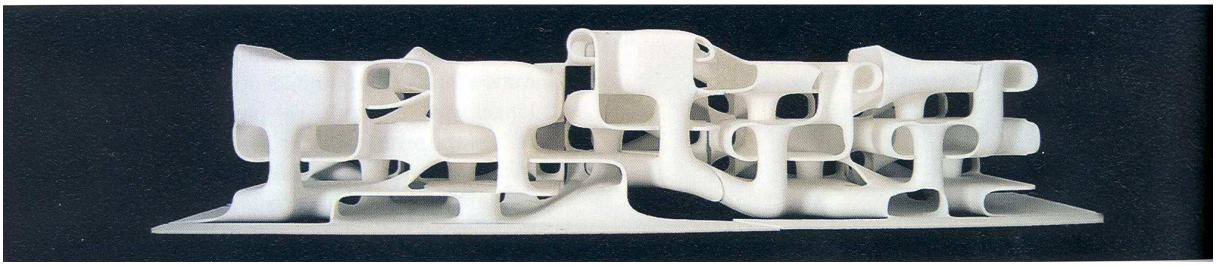


obr. 98 Ron Arad Associates, Holon Design Museum, Israel

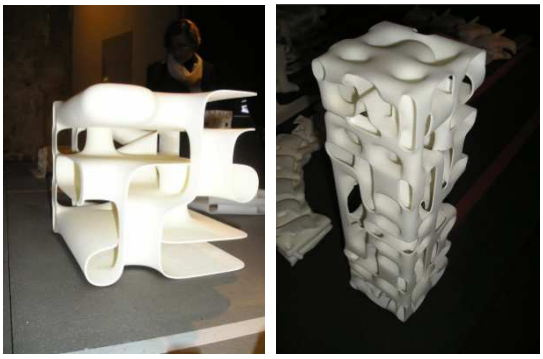
Soustava pruhů – křivek je definována parametricky, deformovány jsou tak, aby bylo dosaženo požadovaných vizuálních efektů z různých směrů pohledu. Tvar je reakcí na okolní zástavbu.

Křivka je definována pomocí několika řídících bodů. Návrh se skládá z několika podobných křivek nad sebou, řídící body jsou u každé křivky o kousek posunuty. K podobnému posunu dochází i u barvy.

3.14 Domin(f)o House, 2003

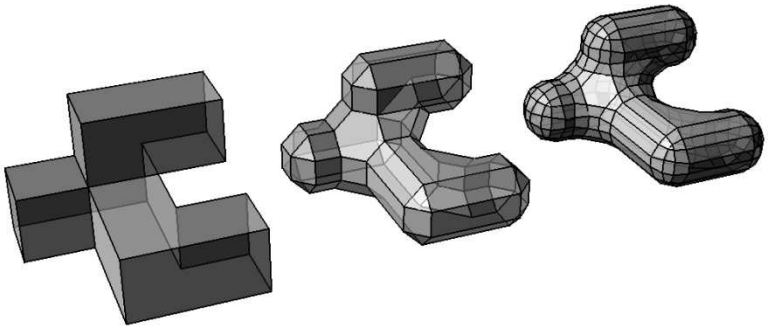


obr. 99 DR_D_Lab (Art Academy Stuttgart) 2002-2003, DR_D Studio (Berlin)



obr. 100,101 Bienale Venezia 2004

Prostorové struktury vycházejí z definic kontinuální plochy (detailně kapitola 2.4.4.4). V příkladech uvedených na obrázcích 99-101 nejsou plochy definovány pomocí vodících křivek, ale naopak. Původní struktura je definována jako pravoúhlá z rovných ploch. Pomocí několika iterací algoritmu, který ořezává ostré hrany, dojdeme k výsledku s hranami zaoblenými, ortogonální struktura je i ve finální podobě jasně zřetelná.



obr. 102 iterace zaoblení

Na obr. 99 je školní projekt ze Stuttgartské univerzity, dopracován v berlínském atelieru. Na obr. 100 a 101 jsou obdobné struktury prezentované na bienále v Benátkách. Na obr. 102 jsem naznačil postup prvních kroků algoritmu.

Modely jsou vytvořeny technologií 3D tisku, spékáním polyamidového prášku.

3.15 D-tower, 2004

obr. 103-106 Lars Spuybroek – NOX a Q.S.Serafijn, D-Tower, Doetinchem

Další z odkazů na práci Frei Otta a Antoni Gaudího. Tvar objektu vychází z chování vznášejícího se balónu upnutého na lanech (Otto Frei – Pneus in Nature and Technics, 1977). Návrh byl generován v obrácené poloze (obdobný princip jako u A. Gaudí), připomínal tak nákupní tašku. Tvar je výsledkem fyzikálního modelování – vztahu nafukované konstrukce a tažných lan. Původní tvar byl plně symetrický, v průběhu byly symetrie uvolňovány, tvar byl průběžně iterativně dopočítáván. Výsledný objekt je prověřen metodou konečných prvků.

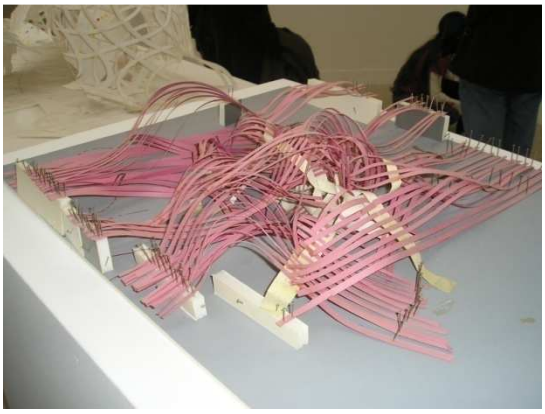
Jednotlivé laminátové díly jsou vyrobeny na formách vyfrézovaných z polystyrenové pěny CNC frézami.

Objekt je doplněn Led diodami, které v noci svítí různými barvami. Barvy se interaktivně mění na základě dat z internetové stránky, která vyhodnocuje nálady obyvatel ve městě.

3.16 Son-O-House, 2004

Architektonická kancelář NOX, Lars Spuybroek tentokrát ve spolupráci se skladatelem Edwinem van der Heide vytvořili další interaktivní projekt. Východisky pro ně bylo snímání pohybu, vytváření tzv. kinetogramů – záznamů pohybu. Na základě pohybu člověka v různých situacích vytvořili lineární struktury, jakýsi spletenec vazeb, s nímž následně pracovali podobně jako v případě D-tower.


obr. 107-108 Lars Spuybroek – NOX a Edwin van der Heide, Son-O-House, Son en Breugel

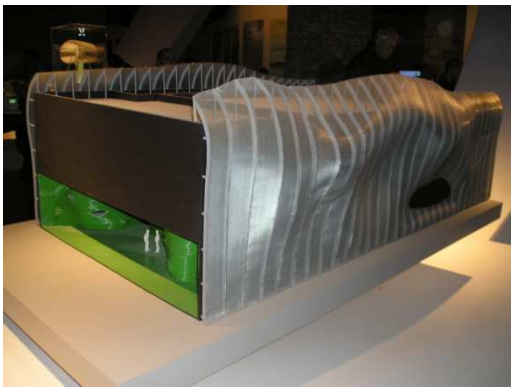



obr. 109-110 Son-O-House – modely

Do struktury vzniklé pohybem byl "nafouknut" balón, čímž byla definována fyzikálně vyjasněná forma. Nosná konstrukce vzniká diagonálními řezy nafouknuté plochy. Postup výroby je podobný jako u D-tower, tentokrát provedený v nerez-oceli.

Do struktury jsou osazeny snímače pohybu a reproduktory, do nichž je, na základě pohybů lidí uvnitř, pouštěna hudba složená pro tento objekt. V různých místech struktury pak zní různé tóny. Hudba interferuje a přizpůsobuje se každému, kdo dovnitř vstoupí.

3.17 Maison Folie, 2004

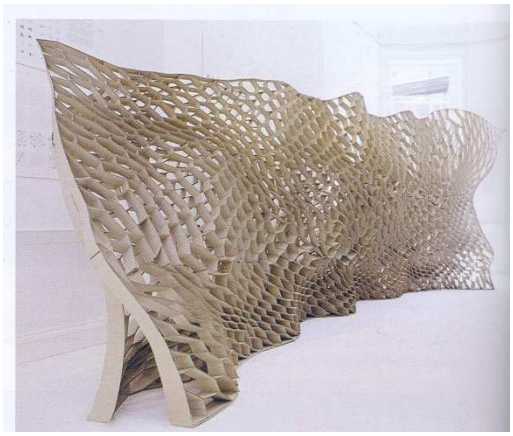
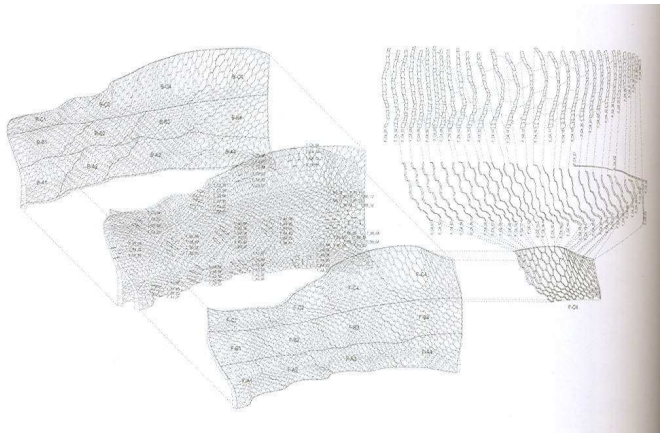


obr. 111 Maison Folie - model


obr. 112-113 Lars Spuybroek - NOX, Maison Folie, Lille

Rekonverze továrního objektu na výrobu textilií na víceúčelovou halu určenou pro umění. Nová funkce objektu byla zvýrazněna fasádami a interiérem tvořeným volnými křivkami a plochami. Fasády jsou modelovány podle chování textilie ve větru. Struktura je definována svislými NURBS křivkami.

3.18 Manifold Honeycomb Morfologies, 2004

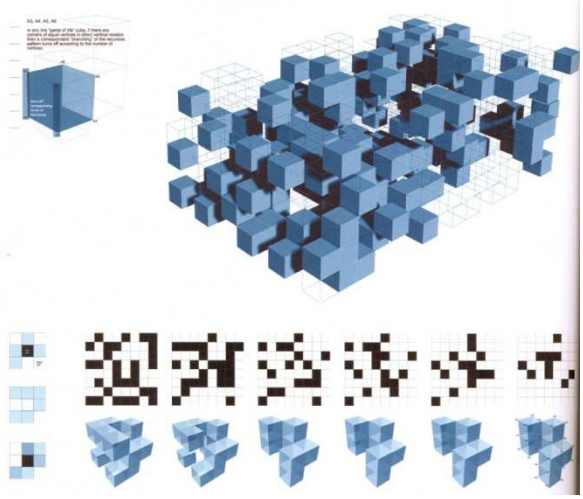



obr. 114-115 Andrew Kudless, Manifold Honeycomb Morfologies, London

Andrew Kudless, "Manifold Honeycomb Morfologies", dizertační projekt pod vedením Michaela Hensela, Michaela Weinstocka a Achima Mengese na AA Graduate School of Architecture, Londýn 2004.

Parametricky definovaná struktura, jednotlivé prvky struktury reagují na chování celku. Skvělý příklad studentské práce dotažené do realizace – instalace.

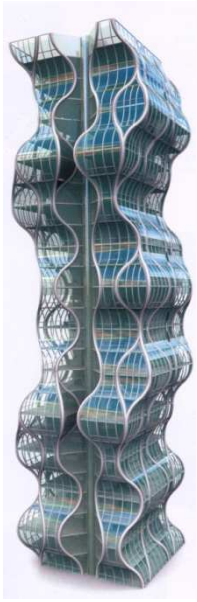
3.19 3D Game of Life, 2004



obr. 116 Brandon Williams / Studio Rocker, 3D Game of Life

Ukázka genetického algoritmu. Struktura složená z krychlí se průběžně vyvíjí na základě vztahů sousednosti mezi jednotlivými krychlemi. Pro každou krychli je vyhodnocován vztah k okolním. Na základě variací a selekce je po několika generacích dosaženo zvolené zadání.

3.20 Project X, 2004



obr. 117 Stanley Perelman, Project X, Mannhatan

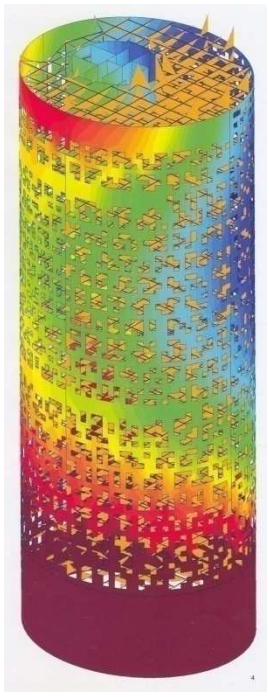
Fasáda objektu je definována pomocí matematických výrazů – interference goniometrických funkcí. Na plochu fasády je použita interference v jednom směru, na sloupy ve dvou směrech.

příklad podobné interference:
stěny: (y=sin1z+sin2z+sin3z, x=1), *(x=sin1z-sin2z+sin3z, y=1)*
sloupy: (y=sin1z+sin2z+sin3z, x=sin1z-sin2z+sin3z)

Výsledkem jsou stěny zakřivené pouze v jednom směru, sloupy jsou tvořeny prostorovými křivkami.

Návrh vypadá velmi komplikovaně, při využití matematiky je ale velmi jednoduchý a nenáročný.

3.21 Torre Agbar, 2005



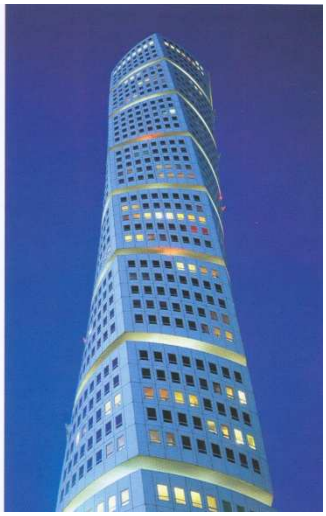
obr. 118 Jean Nouvel, Torre Agbar, Barcelona
obr. 119 Torre Agbar - schéma

Půdorys mrakodrapu se plynule mění z tvaru eliptického na tvar oválný. (viz kapitola 2.4.4.2 - Kinematická architektura)

Pro rozmístění oken byl vyvinut speciální algoritmus, jehož jedním parametrem byla poloha vůči světovým stranám, dalším zpětná vazba od výpočtu statického namáhání.

Jako vedlejší produkt algoritmu vzniklo barevné řešení fasády.

3.22 Turning Torso 2005



obr. 120,121 Satniago Calatrava, Turning Torso, Malmö

Pohyb zespodu nahoru, rotace půdorysu, postupné ztenčování stěn. (viz kapitola 2.4.4.2 - Kinematická architektura) Plochy zakřivené ve dvou směrech.

I když je možné s digitálními metodami paralelu najít, myslím si, že v případě Calatravy jde o čistě umělecké vyjádření a elegantní konstrukci.

Na obr. 117 vidíme jeden z mnoha koncepčních modelů, které samotné budově předcházely.

3.23 Veletrh Miláno 2005


obr. 122,123 Massimiliano Fuksas, New Milan Fair Trade, Milano

Zřejmě největší realizace parametrické architektury, 1,5 km dlouhá zakřivená plocha tvořící osu mezi výstavními pavilony milánského výstaviště.

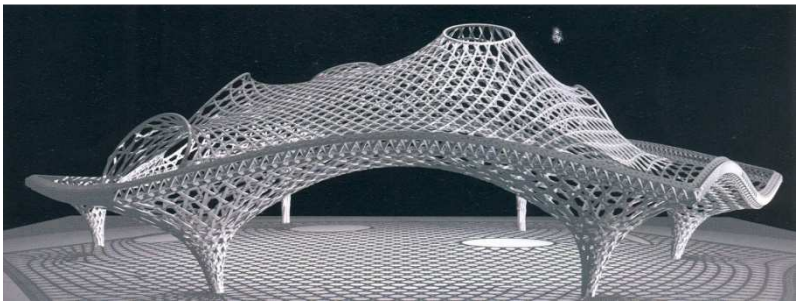
Plocha je definovaná algebraickými výrazy do podoby vln a kráterů. Na ploše je vygenerovaná trojúhelníková síť. Každý segment konstrukce má jiné rozměry, ale všechny jsou typologicky a parametricky shodné.



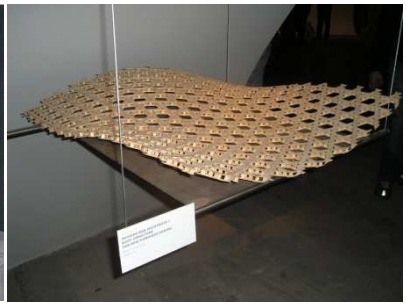
obr. 124-126 Massimiliano Fuksas, New Milan Fair Trade, Milano

Návrh a výroba tohoto objektu jsou ukázkou nových možností parametrického navrhování. Řádově stovky tisíc skleněných ploch a ocelových nosníků, každý s jinými rozměry bylo navrhováno jako jediný celek. Všechny podobné prvky byly vyrobeny stejnou technologií. Každý prvek je osazen na své unikátní místo.

3.24 Pompidou Centre, 2005



obr. 127 Shigeru Ban, Arup AGU, Pompidou centre, Metz



obr. 128 Pompidou centre - model

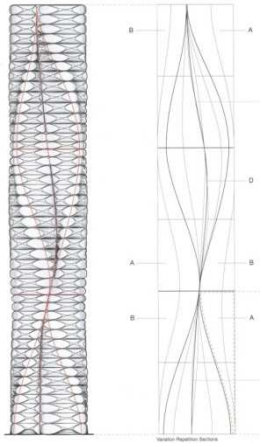
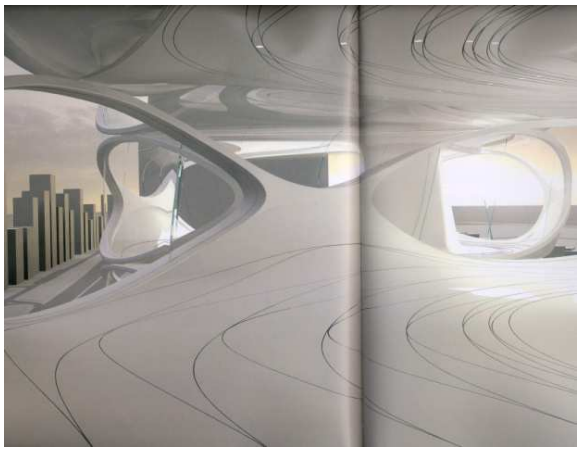
Principy z hlediska digitálního navrhování jsou stejné jako u předchozího příkladu. Struktura vychází z japonské tradiční architektury. Na rozdíl od předchozího návrhu, kde mohla být trojúhelníková síť libovolně optimalizována, zde bylo nutné dodržet průběžné linie. Struktura tak není amorfní ale vázaná. Viz kapitola 2.6.2.

3.25 Phaeno Science Centre, 2005

obr. 129 Zaha Hadit Architects, Phaeno Science Centre, Wolfsburg

Zaha Hadit vytvořila jedinečný kolektiv architektů, programátorů a inženýrů, který je schopen vyvíjet a realizovat nové formy v architektuře. Klíčovou osobou je vedle Zahy Hadit Patrik Schumacher, který je vůdčí osobou teoretického zázemí v atelieru. Na základě architektonických požadavků programátoři vyvíjí algoritmy a systémy, které jsou následně aplikovány v procesu návrhu a při realizaci. Tato symbióza je velmi výhodná z obou pohledů. Architekti mají dokonalé teoretické zázemí, programátoři mají okamžitou zpětnou vazbu.

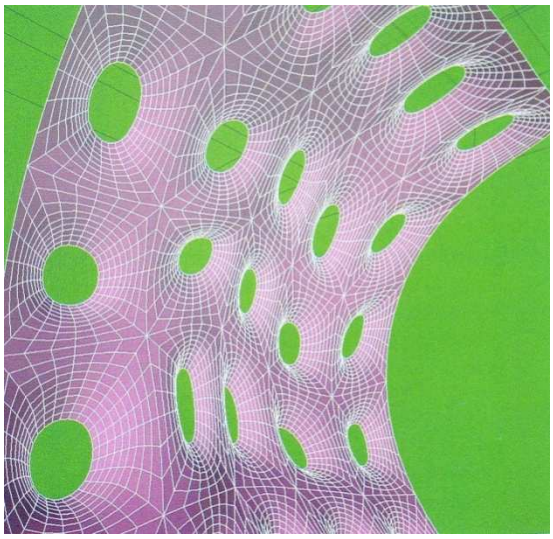
3.26 Residental tower, 2005

obr. 130-132 Contemporary Architecture Practise – Ali Rahim, Hina Jamelle, Dubai

Celistvý parametrický návrh mrakodrapu. Jednotlivé menší části jsou deformovány na základě definice větší struktury, kontinuita ploch je důsledně zachována.

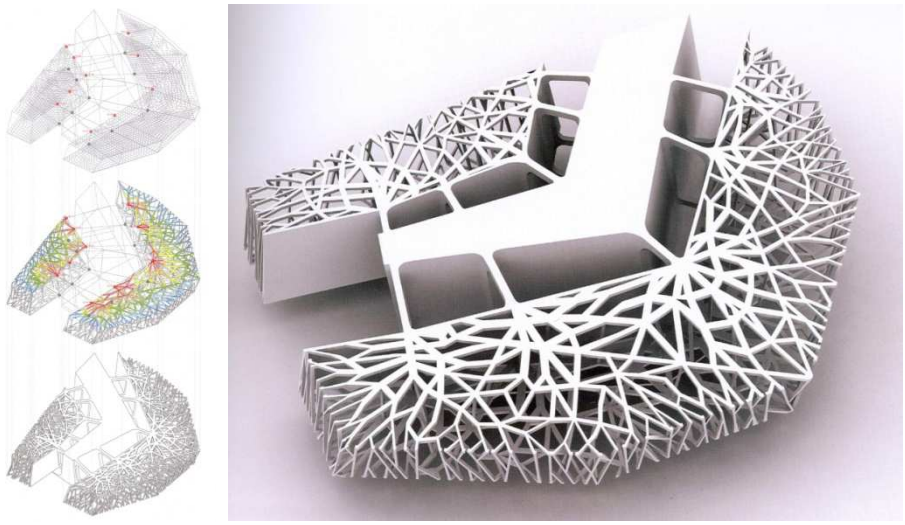
3.27 West Queen West, 2006

obr. 133-134 Alsop Architects , Toronto

Složitá fasáda je vytvořena rekurzivním algoritmem. Struktura fasády byla na počátku pravidelná, soustavou požadavků, definovaných průběhem sil ve fasádě, na jednotlivé otvory se struktura deformuje.

3.28 Národní knihovna, 2006



obr. 135-136 OCEAN a Scheffer + Partner, Praha

Jeden z mnoha soutěžních návrhů na národní knihovnu v Praze. Nosná konstrukce je definovaná fraktálovou strukturou rozrůstající se po fasádě objektu. Jednotlivým prvkům je přisouzena tloušťka nikoli v závislosti na statice, nýbrž na logice vzniku. Nejsilnější jsou počáteční prvky, další postupně slábnou, tak jako větvící se kořeny stromu.

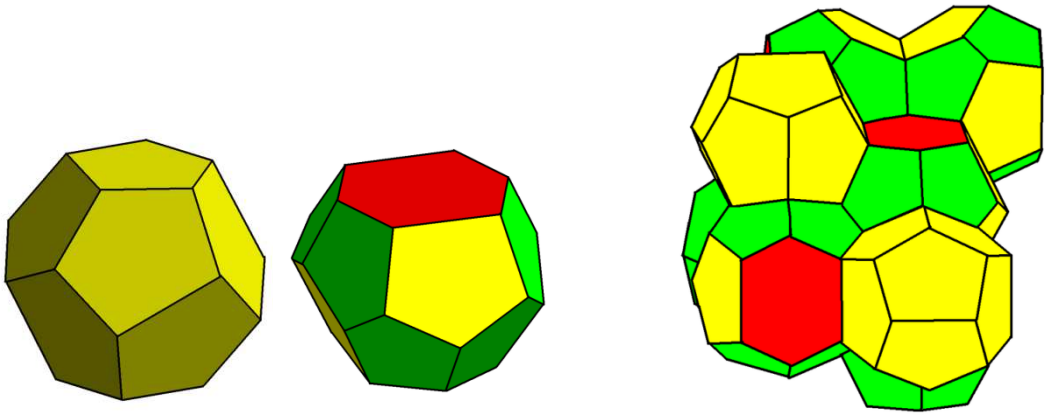
3.29 National Aquatics Center – Watter Cube, 2007

obr. 137 PTW Architects, CSCEC, CCDI, Arup, Peking

Plavecký stadion je na první pohled nesmírně složitou a nepravidelnou strukturou bez jakékoli symetrie a řádu. Asi by tomu tak i bylo, kdyby v roce 1993 fyzikové Weaire a Phelan nezkoušeli vylepšit Kelvinovo řešení vyplnění prostoru stejně velkými objekty při minimálních styčných plochách. Počítačovou analýzou

s pomocí Voronoi – algoritmu (viz. 2.4.6 - Algoritmické metody) došli k řešení, které se nazývá Weaire–Phelanova struktura.

Prostor je vyplněn polopravidelnými dvanáctistěny a čtrnáctistěny kde dvanáctistěn je tvořen pětiúhelníky a čtrnáctistěn dvěma šestiúhelníky a dvanácti pětiúhelníky. Oba objekty mají přitom stejný objem.

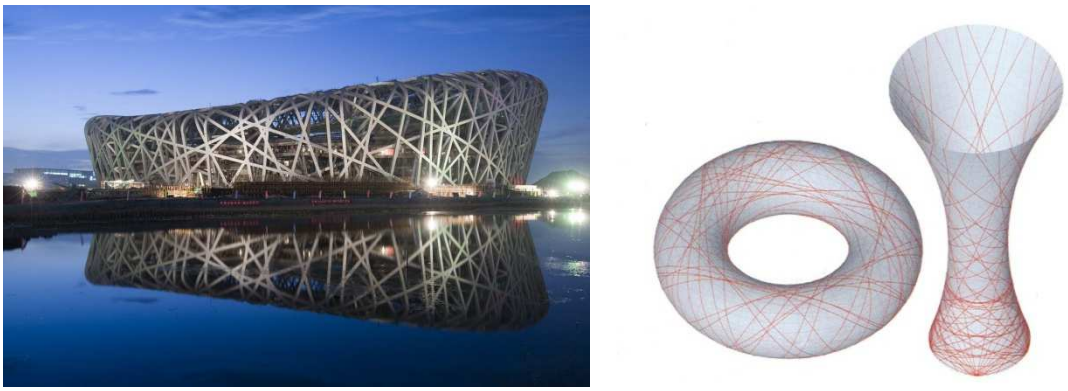


obr. 138-140 Weaire–Phelanova struktura

Výsledná kompozice se opakuje v ortogonální struktuře. Při aplikaci této struktury na plavecký stadion byla struktura vzhledem ke kostce otočena tak, aby symetrie nebyly na první pohled patrné a vznikl dojem nahodilé komplikovanosti.

Struktura byla oříznuta plochami kvádru. Podobně jako u mýdlových bublin byl fyzikálně nasimulován tlak na stěny jednotlivých buněk, což vytvořilo výsledné vyboulení fasády.

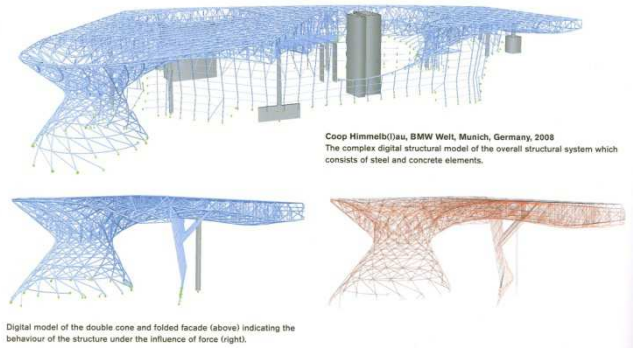
3.30 Národní čínský stadion "Ptačí hnízdo", 2008



obr. 141 Herzog & de Meuron, ArupSport, Bird's Nest, Peking obr. 142 Arup AGU, 2005

Podobný případ jako u předchozího objektu, zdánlivá nahodilost je dosažena pouze posunem os symetrií. Není náhodou, že se na obou projektech podílí ARUP.

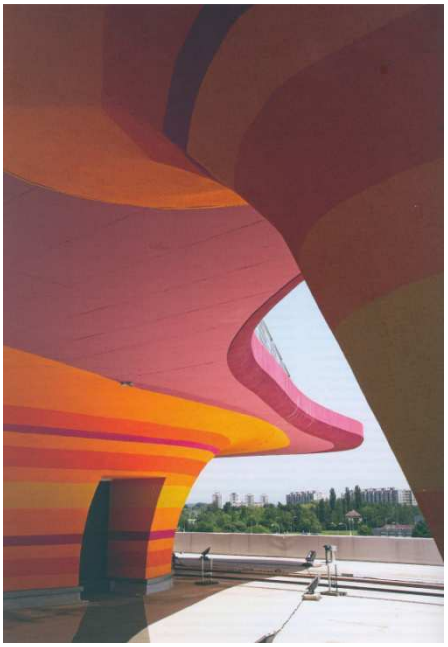
3.31 BMW Welt, 2008

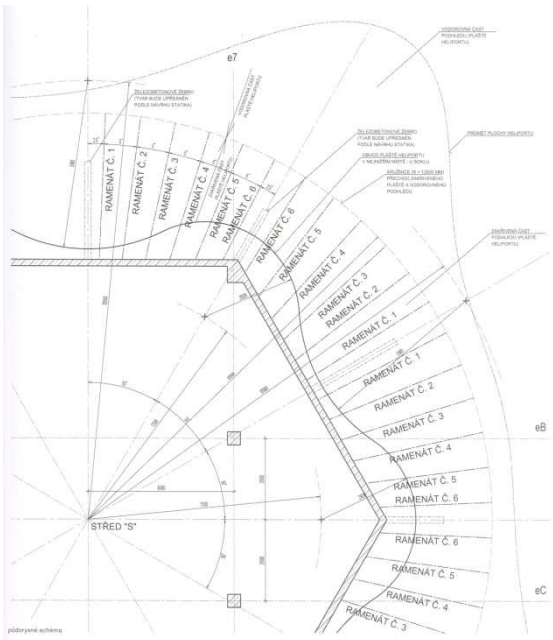


obr. 143,144 Coop Himmelb(l)au, BMW Welt, Mnichov

Další realizace parametrického návrhu, princip je velmi podobný jako v případě Milánského výstaviště od Masimiliana Fuksase.

3.32 Heliport, Pohotovost FN v Hradci Králové, 2008





obr. 145 Juha, Topinka – DOMY, Heliport pohotovosti FN, Hradec Králové

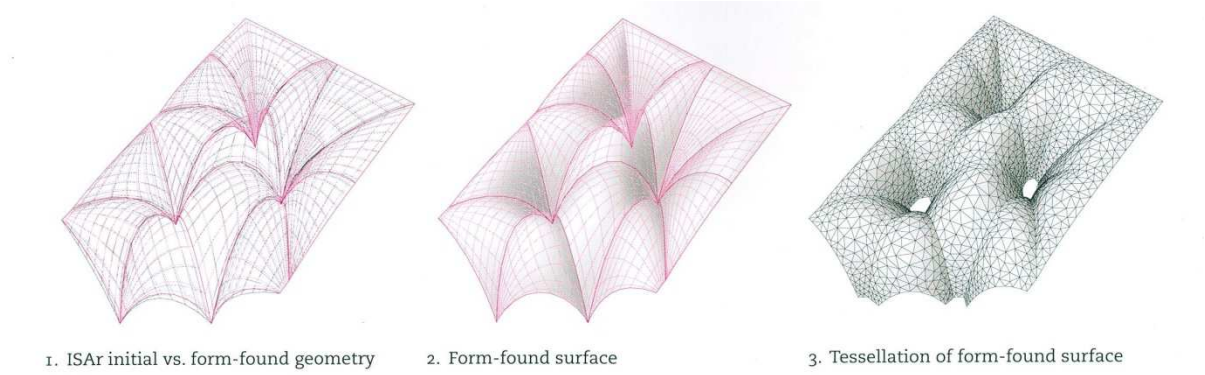
Nástavba na střeše nemocničního objektu je navržena jako zakřivený trojrozměrný objekt. Jeho geometrie je založena na dvou volně definovaných vodících křivkách. Plocha je mezi nimi dotvarována pomocí příčných řezů, ramenátů.

Není třeba zde hledat žádné speciální výpočty, algoritmy. Jde o prostorově volně zakřivenou fasádu navěšenou na geometricky pravidelnou konstrukci.

3.33 Voussoir Cloud, 2008

obr. 146 IwamotoScott Architecture a Buro Happold, Voussoir Cloud, LA

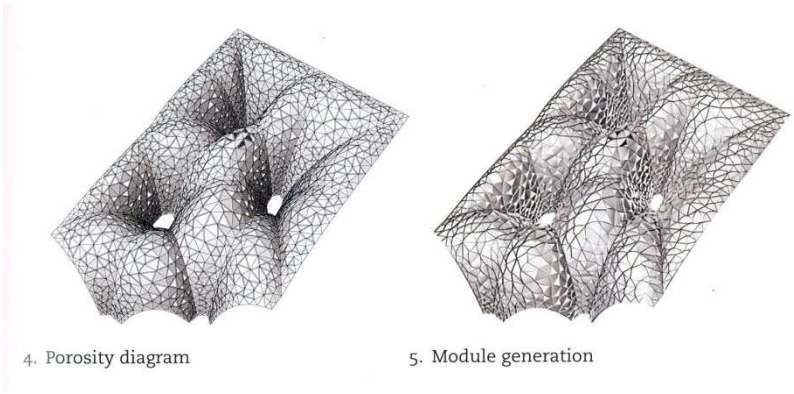
Další návrat ke klenbám od Antoniho Gaudí. Tentokrát v ryze digitální podobě. Architektonická kancelář IwamotoScott a statik Buro Happold vytvořili lehkou samonosnou konstrukci – klenbu. Scripty, pomocí nichž byla klenba definována, vytvořili Chris Chalmers a John Kim pod dohledem Andrewa Kudlesse(viz 3.18).



obr. 147-149 Voussoir Cloud, proces1-3

Na počátku byla předběžná představa o výsledku díla definovaná pomocí NURBS křivek a ploch, poté byl iterativně dopočítán reálný tvar kleneb. V obrácené poloze se klenba vytvarovala vlastní vahou do tvaru řetězovek, čímž se v namáhání konstrukce téměř odstranily momenty. Na výsledné ploše byla pak dalším algoritmem

vyhledána trojúhelníková struktura upravená tak, aby menší trojúhelníky byly v místech většího namáhání, větší na vrcholech kleneb.

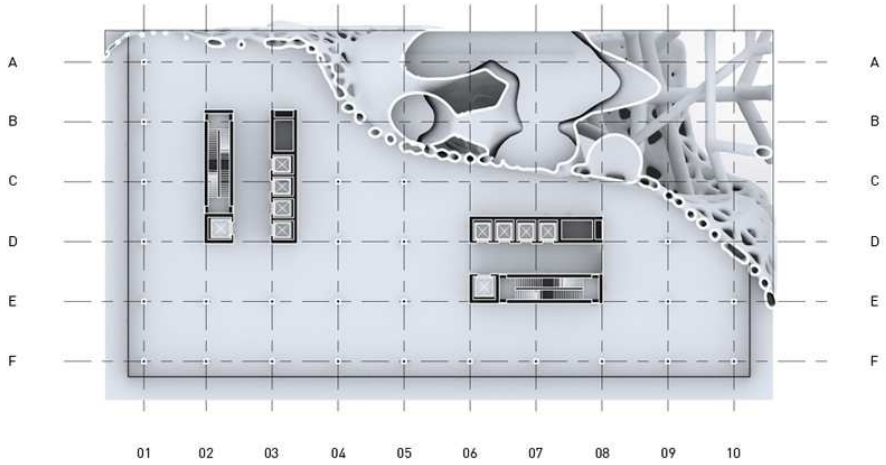


obr. 150-151 Voussoir Cloud, proces 4-5

Dalším algoritmem byla definována porozita kleneb. V místě vypuštěných trojúhelníků byly ohyby definovány obloukem tak, aby došlo i k zakřivení jednotlivých ploch. V tomto bodě má konstrukce jisté rezervy. Měkká konstrukce se často ohýbá pod jinými úhly a křivost pak neodpovídá původnímu výpočtu, dochází k borcení klenby. Dále pak zvolená porozita umožňuje zakřivení pouze některých ploch, navíc nikdy ve všech směrech. Výsledná struktura je tedy kombinací rovných, a různě zakřivených segmentů. Výsledek je z hlediska křivosti poněkud zmatený.

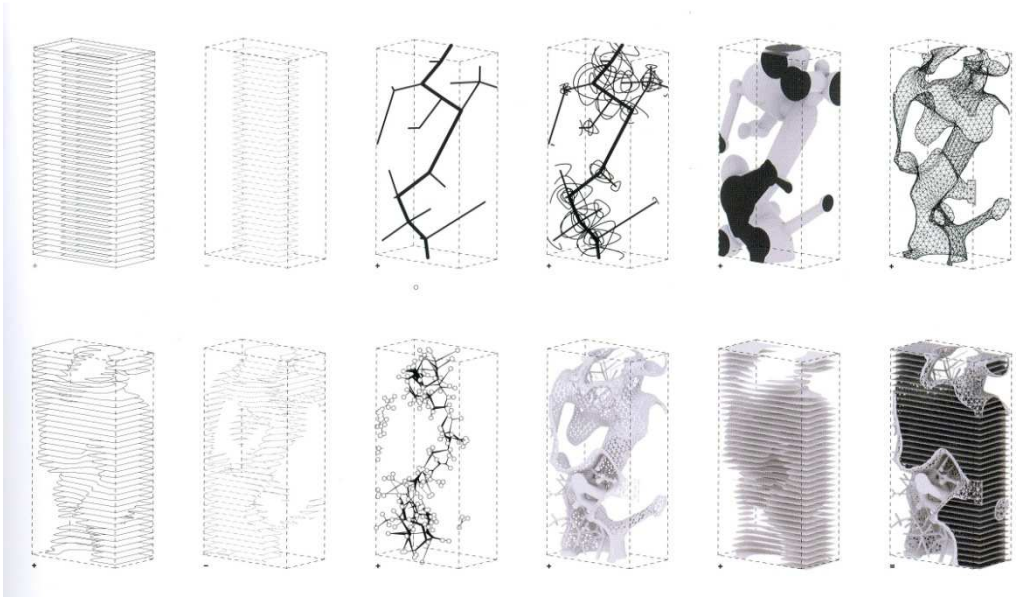
Posledním bodem je výroba. Jednotlivé segmenty byly vyřezány z tenké dřevěné dýhy (wood laminate) a pozohýbány do krabicového tvaru, který víceméně drží tvar.

3.34 Proto-Synthesis & Trabeculae, 2008



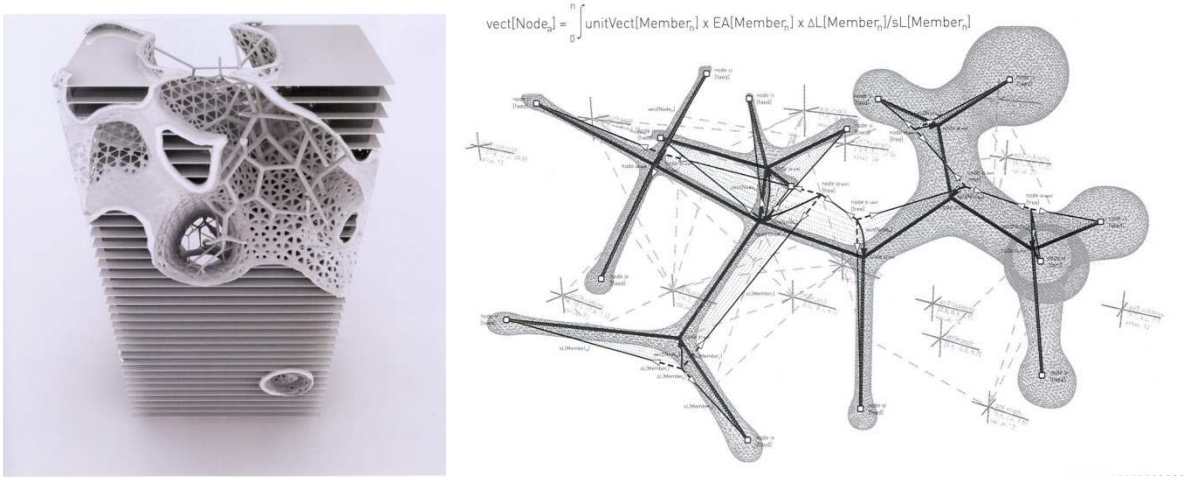
obr. 152 Supermanoeuvre, Trabeculae - půdorys

V tomto projektu jsou prezentovány přístupy kanceláře Supermanouvre. Pomocí genetických algoritmů jsou hledány ideální vazby uvnitř objektu, vyhodnocovány jsou světelné podmínky okolo vytvářených atrií.



obr. 153 Proto-Synthesis & Trabeculae – vývojový diagram

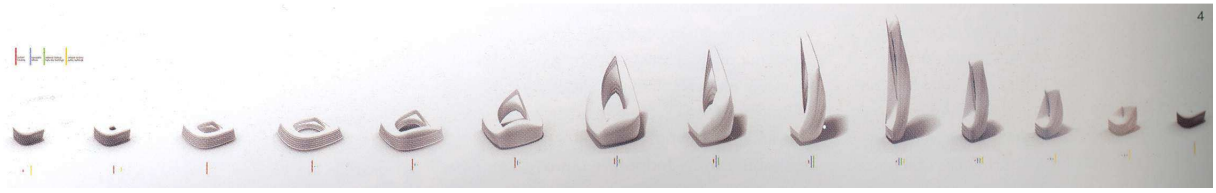
Původně svislé atrium s přístupem světla pouze shora přináší nerovnoměrné světlo do objektu, Lineární struktura (diagram 3) vyhledaná genetickým algoritmem svislé atrim modifikuje tak, aby do něj světlo přicházelo i bočními stěnami (diagram 3-5). Okolo lineární struktury je vygenerován prostorový objekt v závislosti na požadovaných velikostech prostoru pomocí matematických definic (srovnej kapitolu 2.4.5.2 - Metaball a obr. 155), který je odečten od původní konstrukce. Do vzniklého prázdného prostoru atria je vygenerována nová nosná konstrukce nahrazující původní odebrané prvky. Model je vytvořen metodou 3D tisku.



obr. 154,155 Trabeculae - model, metaball

3.35 Velké pražské Benátky, 2008

obr. 156 Tomáš Legner, Velké pražské Benátky, Praha

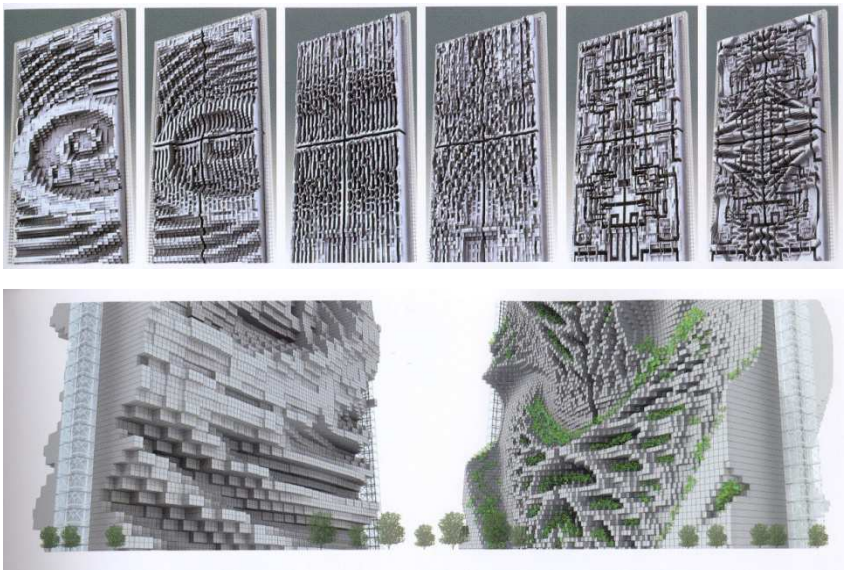


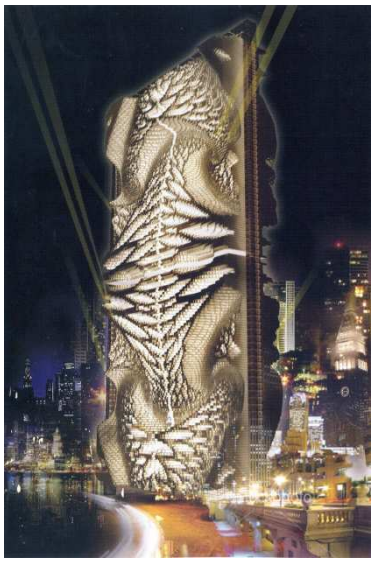
obr. 157 Velké pražské Benátky - schéma

Diplomová práce na VŠUP řeší návrh urbanistické struktury tvořené parametrickou blokovou zástavbou.

Parametry jednotlivých bloků definují velikost vnitrobloku, podlažnost a výškový posun jednotlivých rohů bloku. Na různých půdorysech definovaných nepravidelnými polygony vznikají jediným algoritmem různé, ale koherentní objekty, které lze velmi jednoduše modifikovat (viz obr. 157) do různých tvarů.

3.36 Geno-Matrix, 2009



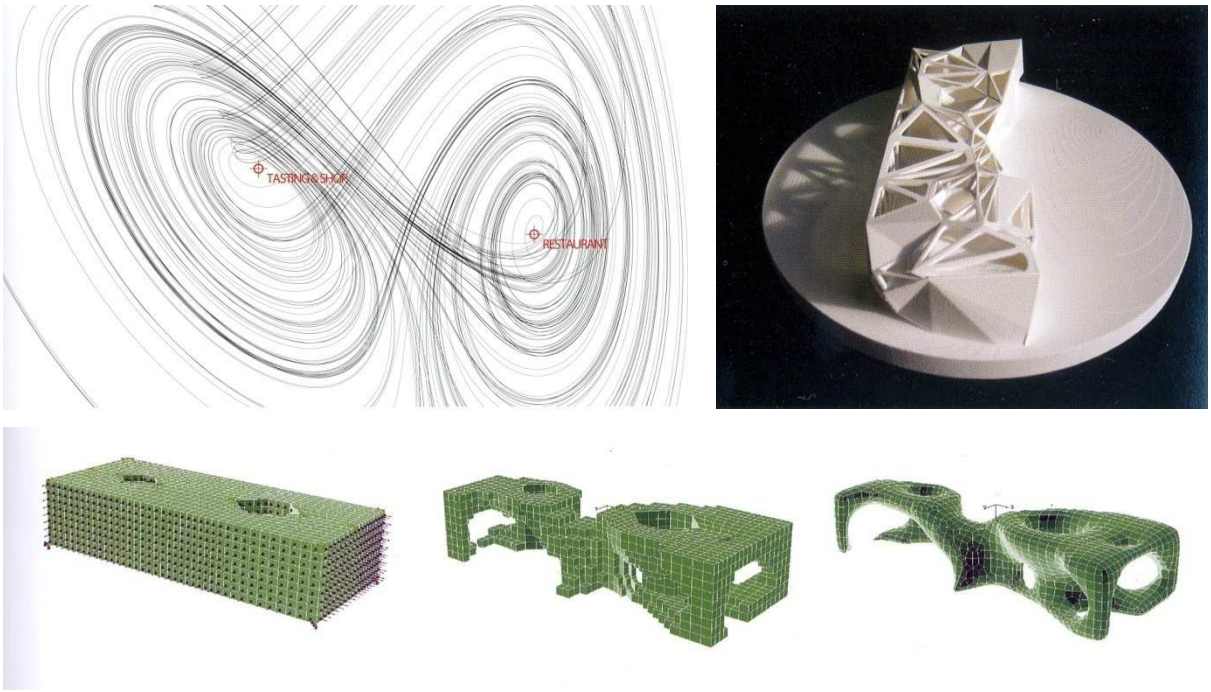


obr. 158-160 Tang & Yang Architects, Geno-Matrix

Mrakodrap je navržen z tisíců jednotlivých buněk, které se mohou vysouvat podél vlastní osy. Tím je vytvořena struktura umožňující nekonečné množství variant vzhledu objektu.

Jednotlivé buňky jsou vysouvány do různých poloh na základě vlivů okolního prostředí, vztahů sousednosti či genetických algoritmů. Na obrázcích 158-160 jsou různé varianty chování domu definované různými požadavky.

3.37 Strange Attractor, 2009



obr. 158-160 MESNE, Strange Atractor, South Eastern Victoria

V roce 1963 Edward Lorenz definoval základní diferenciální rovnice – tzv. Lorenzovy transformace pro pohyb v silovém poli v trojrozměrném prostoru.

dx/dt = s(y-x)

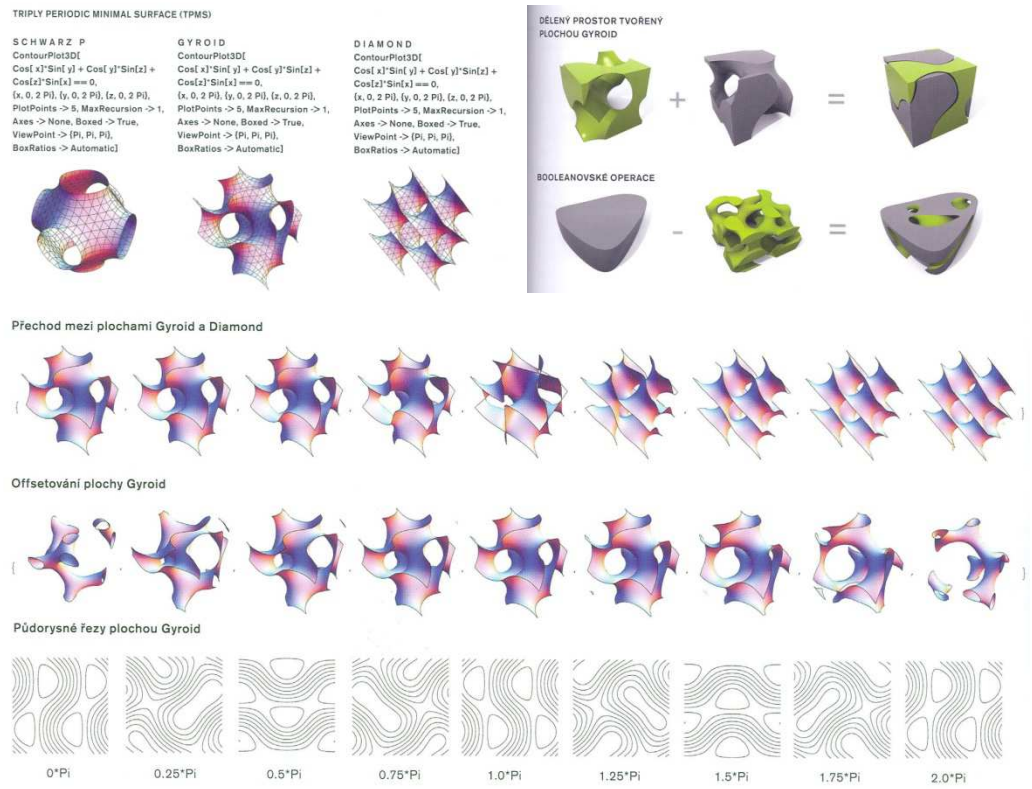
dy/dt = rx-y-xz

dz/dt = xy – bz

Rozvojem těchto rovnic je možné definovat pohyb bodu v prostoru v okolí tzv. attractorů – generátorů silového pole. Na obr. 158 vidíme trajektorie pohybu okolo dvou center, Pomocí těchto trajektorií byl ořezán kubický objekt. Algoritmus pracuje na principu voxelizace (viz kapitola 2.4.5.3.A – voxelová interpretace), (obr.160).

Výsledný objekt je doplněn o konstrukci zasklení vyřezaných prostor. Model na obr. 159 vyroben technologií 3D tisku.

3.38 Liquid Blocks, 2010

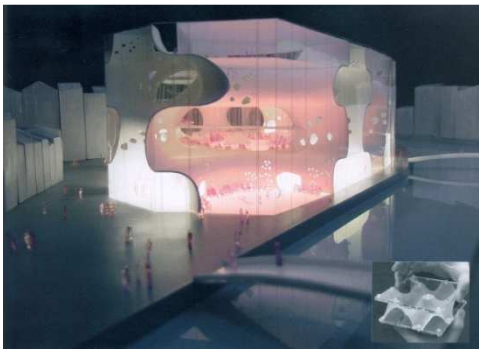


obr. 161-163 Michal Bednář / FLO(W), Liquid Blocks

Projekt vychází z matematického vyjádření minimální plochy, zabývá se úzkou skupinou těchto ploch, které jsou matematicky popsány pomocí algebraických rovnic – tzv. TPMS (Triply Periodic Minimal Surfaces) TPMS jsou definovány rovnicí o třech neznámých, kde argumenty neznámých tvoří periodické výrazy (např. goniometrické funkce).

Pomocí těchto rovnic je definován "tekoucí prostor" rozdělený "kontinuální plochou" na dva. (viz kapitola 2.4.4.4 Vnitřní pohyb)

Obdobným příkladem je dřívější návrh slavnějšího kolegy Toya Ita v Gentu



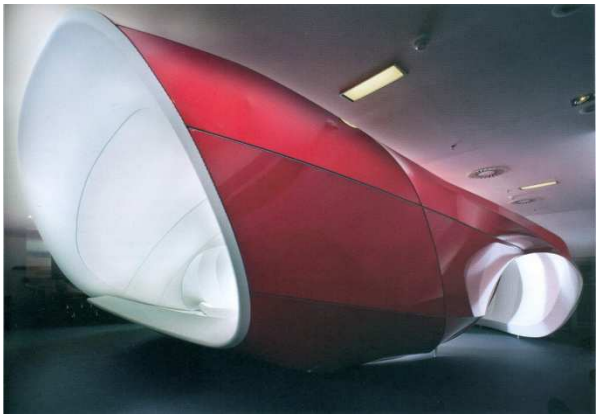
obr. 164, Toyo Ito, Andrea Branzi, Forum for Music, Dance and Visual Culture, Gent, 2004

3.39 Gallery Building, 2010

obr. 165 SPARCH a Alsop Design, Gallery Building, Shanghai

Objekt je definovaný jako polygonální struktura reprezentace trojrozměrného zakřiveného objektu. Viz obr. 02 – interpolace zakřivené plochy v kapitole 2.1 – Geometrie. Barevné členění ploch je umělecký vyjádřením, kde tuším inspiraci v obrazech Pieta Mondriana.

3.40 Zákaznické centrum Vodafone, 2011



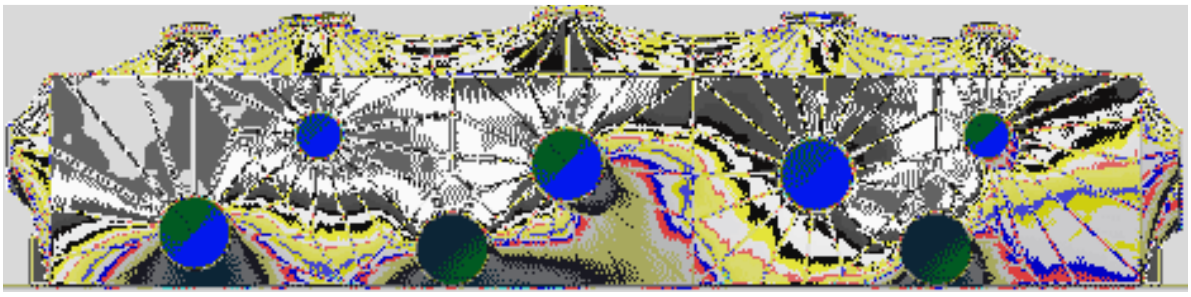
obr. 166-167 Luka křížek, IO Studio, Vodafone, Praha

Objekt je navržen na základě vlastností použitého materiálu. Jde o soustavu membrán – minimálních ploch tvořených barrisolovou folií – napnutých mezi nosné konstrukce. Nosná konstrukce je chytře navržena tak, aby se částečně opticky eliminovaly vlastnosti dané technologií – protisměrná křivost membrán, tangenciální diskontinuita sousedních ploch – a celek působil jednotně.

Membrány jsou doplněny Corianovým nábytkem zatepla ohýbaným.

3.41 Olympic Shooting Venue, 2012





obr. 168-170 Magma Architecture, Olympic Shooting Venue, Londýn

Montované dočasné haly pro střelecké disciplíny pro olympijské hry v Londýně jsou opláštěny průsvitnou membránou. Membrána tvoří minimální plochu definovanou okrajovými podmínkami – ortogonální hrany a kruhové "průstřely"

Plošné rozvržení jednotlivých dílců folie odpovídá dělení plochy pomocí voronoi algoritmu (viz obr. 51 – Voronoi 2D v kapitole 2.4.6 – Algoritmické metody).

4 Návrh systému

Na vývoji systému se v letech 2007 – 2009 podílelo volné sdružení tří lidí: Adam Sirotek, Rudolf Müller, Petr Soviš. V této sestavě jsme se věnovali myšlence plné digitalizace procesu návrhu architektury. Od roku 2009 není sdružení aktivní, na práci pokračuji sám. V této kapitole se pokusím prezentovat ideje, které během spolupráce vznikly.

4.1 Paralela

Architektura je dnes obecně chápána jako synonymum pro stavitelství. V průběhu historie se vyvíjely formy a slohy, které odpovídaly stavu společnosti a dodnes poukazují na statut společnosti, závislý na politické, ekonomické, náboženské a sociální úrovni. Vždy je ale forma výsledkem ega jednotlivce nebo určité skupiny.

Tvrdíme, že architektura v základě pokrývá základní lidské potřeby na životní prostor. Pokud je uspokojíme, přichází nadstavba v podobě kulturního, duchovního, estetického vývoje společnosti.

Zmíněné potřeby můžeme rozdělit na biologické, které se dlouhodobě nevyvíjí, a na společenské, které souvisí se vzděláním a potažmo s vývojem technologií, jejichž rozvoj je překotný. Zjednodušeně od pazourku po virtuální realitu. V dnešní době je civilizovaná společnost již naprosto závislá na námi vytvořeném virtuálním světě, na světě počítačů.

Z toho se odvíjí i trend digitálního navrhování.

V podstatě ho můžeme rozdělit na dva proudy, z nichž jeden se ubírá směrem tupého využití 2D a 3D nástrojů, jichž je nepřeberné množství a je zde celkem problematická kompatibilita. Tento směr pouze spočívá ve využití PC jako tužky.

Další postup, kterému s trochou nadsázky můžeme říkat digitální architektura, se baví parametrickým navrhováním, genetickými algoritmy apod. Jsou při tom využity stejné nástroje, přičemž jsou do nich implementovány nadstavby v podobě scriptů a plug-inů. Ambiciózněji projekty vytváří unikátní prostředí s vlastními algoritmy... Můžeme konstatovat, že u většiny těchto projektů jde zejména o hledání nových prostorových forem, přičemž vnitřní členění je znova podřízeno lidským

potřebám, potažmo stavebnímu programu a je závislé na lidském vstupu. Ojediněle najdeme i výjimky, kde je celý prostor včetně interiéru tvořen “parametricky”. Do jisté míry jde ale jen o empirické pokusy. Neustále se ale hledají důvody proč jít touto cestou a potažmo nová východiska.

My jsme hledali východiska jinde než v hledání nových forem za využití stávajících systémů. Inspiroval nás současný stav společnosti a její závislost na technologiích. Důležité je přiznat si tuto závislost i vzhledem k tomu, že jsme její součástí. Dále to byla příroda, kde můžeme uvést jako příklad živý organismus, který obsahuje genetickou informaci – predispozice a základní vzorce pro přežití. V neposlední řadě to je lidské myšlení, které se vzhledem k vzdělání neustále vyvíjí.

Zjednodušeně řečeno chceme vytvořit digitální paralelu k lidskému myšlení. Využít schopnosti asociativní paměti, analýzy a syntézy. Je zde ovšem jeden velmi těžko dosažitelný aspekt a to cítění. Dnes využívá virtuální svět vědomostí zejména jeho tvůrce, člověk. Chceme vytvořit nový systém, kterému nadefinujeme základní predispozice a budeme ho učit. Dále ale dokáže samostatně využívat informace z virtuálního světa a na jejich základě se vyvíjet, učit se bez dalšího lidského vstupu.

Pozn.: Text vznikl v roce 2009 jako součást prezentace konceptu.

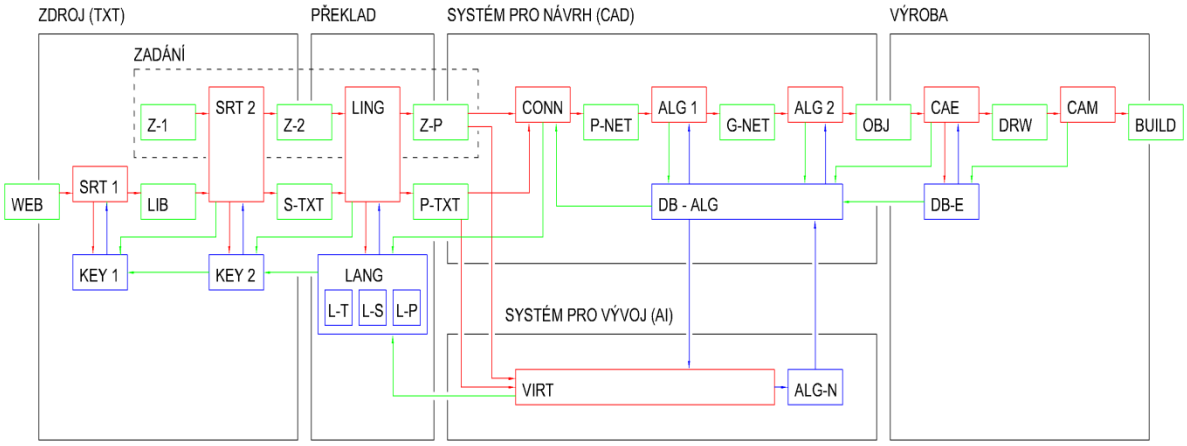
Vzhledem k následnému vývoji se již dnes s některými tvrzeními a definicemi nemohu ztotožnit. Zejména se to týká digitální architektury, ta je v textu definována jako zábava, a je omezena na parametrický design, nějaké algoritmy a scriptování. Dle mého názoru je definice výrazně širší, obecnější. Například nový navrhovaný systém bude také digitální architekturou, nevidím tedy jediný důvod se vůči tomuto termínu vymezovat.

4.2 Schéma systému

Prvotní myšlenkou, která definuje strukturu navrhovaného systému, je tvrzení, že počítač bude vždy dělat jen to, co ho naučíme. Program nikdy nebude chytřejší než jeho tvůrce. Může být ale rychlejší, přesnější a pracovat s větším množstvím dat.

Struktura systému je tedy navržena jako digitální obdoba lidského procesu navrhování.

4.2.1 Celkové schéma



obr. 171 Schéma systému – obdoba procesu architektonického navrhování člověkem

Celý systém je rozdělen do několika propojených podsystémů. Na počátku jsou zdroje, zadání projektu a informace potřebné pro samotný návrh. V další části jsou prvotní informace tříděny a analyzovány. Třetí část se věnuje samotnému navrhování staveb. Poslední částí je výroba – realizace. Navíc je systém pro navrhování podporován samostatným systémem pro vývoj – teoretickou podporou, výzkumem, vývojem.

V dalších kapitolách se budu věnovat jednotlivým podsystémům detailněji.

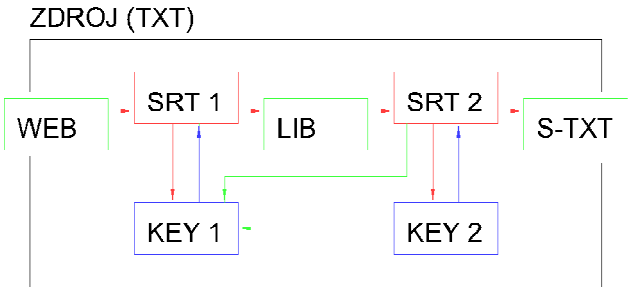
4.2.2 Subsystém 1 - Zdroj

Systém je určen pro získávání informací, které jsou potřebné pro navrhování. Požadavky na typ informací na systém kladou další subsystémy, které je potřebují.

Vzhledem ke snaze maximálně digitalizovat celý proces, je zřejmě nejsnazší vyhledávat již digitalizovaná data. Největší přístupnou databází je dnes internetová síť, proto je systém primárně určen pro získávání dat právě z internetu. Nejsou však vyloučeny ani jiné možnosti.

Zajímavou otázkou je typ dat, která mají být vyhledávána. Nejjednodušší jsou data ve formě textu, jejich vyhledávání je totožné se stávajícími používanými vyhledávacími algoritmy. Pro architekturu jsou ale velmi zajímavá data grafického charakteru. Zatím je možné je vyhledávat pouze na základě textového popisu

a indexace. Vyhledání relevantních grafických dat na základě jejich obsahu je zatím "v plenkách ".



obr. 172 Schéma subsystému – Zdroj

Ve schématu jsou jednotlivé programy, databáze a vazby mezi nimi. V textu jsou popsány jednotlivé toky dat.

WEB => LIB Sběr dat, výběr, třídění a katalogizace

WEB Obecný zdroj dat (internet)

SRT 1 Program pro vyhledávání a třídění dat souvisejících s architekturou Vyhledávání je řízeno databází klíčových slov KEY 1

KEY 1 Databáze klíčových slov, slovních spojení, článků s příslušným ratingem relevance. Rating je modifikován na základě zpětných vazeb (posouzení relevance zpracovaných dat v dalších fázích procesu, požadavky na vyhledání nových dat)

LIB Katalogizovaná knihovna dat (textů) souvisejících s problematikou architektonického navrhování. Každý text je programem SRT 1 ohodnocen, indexován a zařazen do knihovny. Data jsou na základě hodnocení zpětně dohledatelná, porovnatelná. Knihovna slouží jako zdroj dat pro navrhování.

LIB <=> S-TXT Výběr a přizpůsobení dat pro překlad

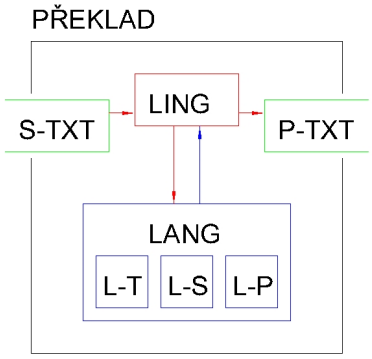
LIB Zdroj dat – knihovna

SRT 2 Program pro selekci, třídění přeložitelných výrazů, definice použité syntaxe, detailní rozbor textu. Proces probíhá na základě parametrů v databázi KEY 2

KEY 2 Databáze parametrů a postupů řídících program SRT 2. Parametry v databázi jsou modifikovány na základě zpětných vazeb z následujících fází procesu návrhu

S-TXT Výrazy, slovní spojení, vazby, texty apod. připravené pro překlad. Každá položka v databázi je indexována připojenými parametry.

4.2.3 Subsystém 2 - Překlad



obr. 173 Schéma subsystému – Překlad

Systém je určen pro modifikaci získaných dat do formy parametrické struktury, se kterou bude umět pracovat systém pro návrh. V opačném směru překládá požadavky systému pro návrh do textové formy, se kterou umí pracovat systém Zdroj a požadované data vyhledat.

S-TXT <=> P-TXT Překlad z textové formy do podoby parametrických objektů

S-TXT Zdroj dat pro překlad (data jsou získána systémem Zdroj)

LING — Program na překlad dat z podoby textu do podoby parametrických objektů. Proces vyhledávání a přiřazování parametrických (geometrických a číselných) objektů odpovídajících překládaným výrazům v textové podobě. Proces je oboustranný. Program je řízen databází (slovníkem) LANG

LANG — Slovník je složen ze tří částí:

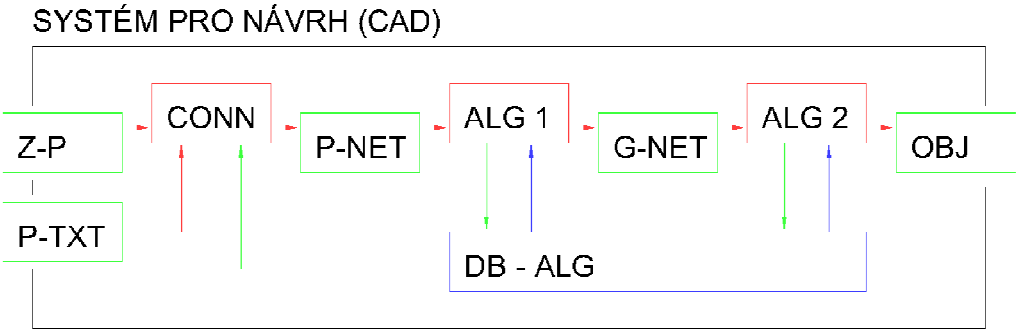
- L-P — Databáze předdefinovaných parametrických objektů (jednoduché grafické objekty a vazby mezi objekty, modifikovatelné parametry vyjadřují vlastnosti a možné chování objektů)
- L-T — Databáze výrazů, slovních spojení ve formě textu
- L-S — Databáze vazeb pro přiřazování parametrických objektů a hodnot k textům. Relevance přiřazování je hodnocena ratingem, rating je průběžně modifikován zpětnou vazbou z dalších fází projektu.

P-TXT — Výstup překladu – parametrické objekty s přiřazenými hodnotami a vazbami

4.2.4 Subsystém 3 – Systém pro návrh

Asi nejsložitější částí je samotné navrhování. Na základě vyhledaných a předpřipravených dat (data obecná a konkrétní zadání) je pomocí algoritmů poskládán a vyladěn požadovaný objekt. Výsledný objekt je ve formě 3D modelu a dalších dat, která jsou srozumitelná pro běžné grafické programy.

Pro každé jednotlivé zadání jsou vyhledány nejvhodnější kreativní algoritmy. Skladba jednotlivých programů v systému je může průběžně měnit dle postupujícího vývoje celého systému. Prozatím je navržena s ohledem na fungování známých existujících softwarů a algoritmů.



obr. 174 Schéma subsystému – Návrh

P-TXT <=> OBJ Proces tvorby finální podoby objektu dle konkrétního zadání.

P-TXT Zdroj dat - obecná data

Z-P Zdroj dat - data konkrétního zadání

CONN Program propojující jednotlivé nadefinované parametrické objekty na základě jejich známých vzájemných vazeb.

P-NET Komplexní síť složená z propojených parametrických objektů.

DB-ALG Databáze algoritmů. Výběr algoritmů pro řešení daného zadání je určován jejich ratingem. Rating je dán úspěšností algoritmu.

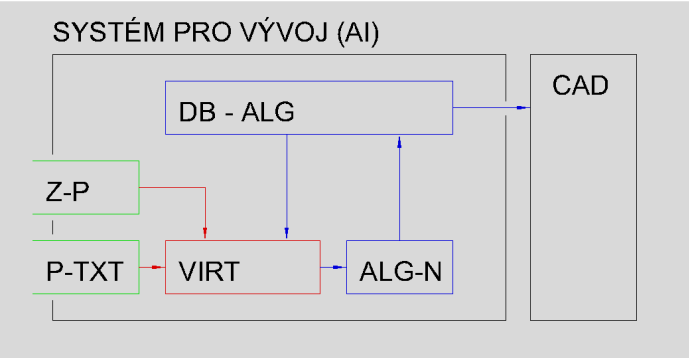
ALG 1 Algoritmus vytváří ze sítě parametrických objektů konkrétní parametrickou architektonickou formu (G-NET).

G-NET Konkrétní komplexní parametrická forma – grafický modifikovatelný objekt.

ALG 2 Algoritmy modifikující G-NET na konkrétní geometrickou formu. Výstup programu do zvoleného existujícího grafického prostředí (3D CAD Software)

OBJ Konkrétní výsledná digitální geometrická forma, např. 3D model.

4.2.5 Subsystém 4 – Vývoj



obr. 175 Schéma subsystému – Vývoj

Tento systém slouží jako teoretické zázemí pro systém pro návrh. Jsou zde na základě náhodně generovaných vazeb, genetických algoritmů a genetického programování vyvíjeny nové algoritmy. Vyvinuté algoritmy – metody jsou posléze aplikovány systémem pro návrh. Systém může být na lidském vstupu téměř nezávislý. Autodidaktické algoritmy a princip náhodných asociací se přibližují principům umělé inteligence.

P-TXT <=> DB ALG — Zapojení vývojové jednotky nezávislé na člověku, umělá inteligence

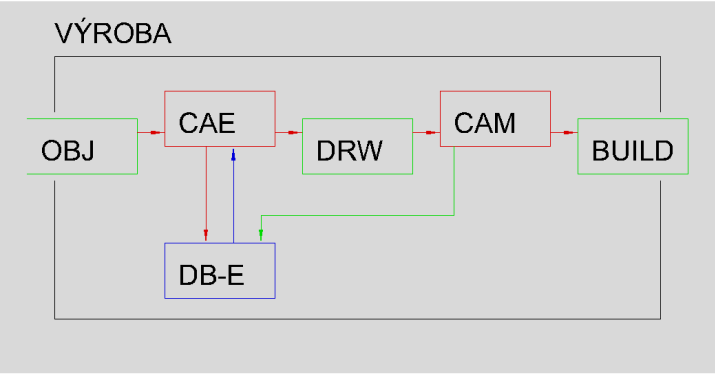
P-TXT, Z-P — Zdroj dat obecných i konkrétního zadání

VIRT — Automatizovaný autodidaktický systém vyvíjející nové postupy (algoritmy) na základě hledání neznámých vazeb.

ALG-N — Nově vzniklý algoritmus. Tento algoritmus bude následně zařazen do DB-ALG a testován v systému pro návrh (CAD) na konkrétních úlohách.

DB-ALG — Databáze algoritmů. Úspěšnost nového algoritmu se odráží ve výši jeho ratingu. Rating je zvyšován po úspěšném splnění úkolu algoritmu. Následné nasazování algoritmu do procesu je určováno výškou ratingu.

4.2.6 Subsystém 4 – Výroba



obr. 176 Schéma subsystému – Výroba

Poslední částí systému je převedení projektu do formy, která je čitelná pro obráběcí stroje. Tuto fázi není jednoduché automatizovat. Plně závisí na aktuálních možnostech a softwarovém prostředí zvolených výrobních prostředků.

OBJ <=> BUILD Systém pro vytváření výrobní dokumentace a podporu realizace

OBJ Zdroj dat – projekt v digitální podobě

DB-E Databáze specifikací požadované dokumentace pro výrobu, konkrétních fyzických materiálů, jejich obrábění, technických a technologických postupů.

CAE Proces vytvářející realizační dokumentaci.

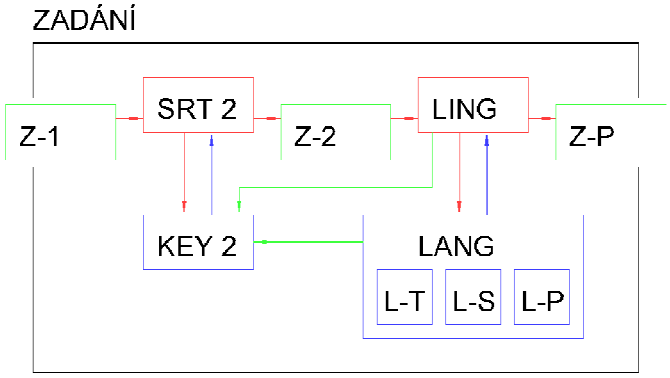
DRW Dílenská dokumentace.

CAM Výroba.

BUILD Kompletace finálního produktu.

4.2.7 Zadání

Ve schématu zbývá nepopsána velmi důležitá část – zadání. Nejde o nové funkce, programy a algoritmy, ale o vytvoření přijatelného uživatelského rozhraní pro vstup člověka do celého, jinak automatizovaného, procesu.



obr. 177 Zadání

Z-1 <=> Z-P — Modul pro zadávání konkrétní úlohy do systému pro návrh a vývoj

Z-1 — Konkrétní zadání ve formě textu

SRT 2 — Text zadání je uzpůsoben pro překlad (rozbor textu, syntaxe)

Z-2 — Text zadání přizpůsobený pro překlad

LING — Překlad zadání do formy parametrických objektů

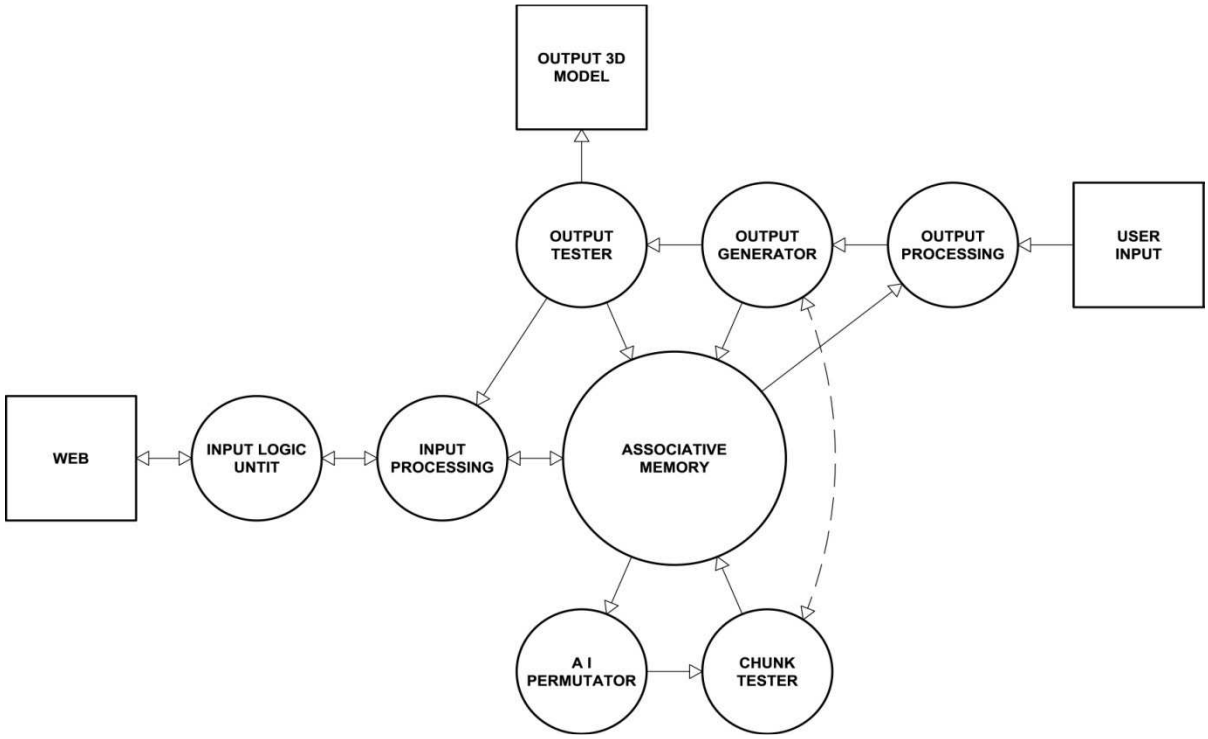
Z-P — Konkrétní parametrické objekty vyjadřující zadání

Toto schéma je připraveno na chvíli, kdy by celý systém fungoval bezchybně a na základě zadání vytvořil celý návrh automaticky. Je ale nutné předpokládat možnost lidského vstupu ve všech fázích návrhu, jednak z důvodu kontroly a vývoje, ale také pro umožnění spolupráce člověka a počítače na navrhovaném objektu.

4.3 Implementace

Na základě představy o schopnostech systému, prezentovaných v předchozí kapitole, byl ve spolupráci s Petrem Sovišem zpracován koncept systému z pohledu programátora.

4.3.1 Koncept systému



obr. 178 Implemetace - koncept

4.3.2 Web

Internet downloader - stahuje balíčky dat z internetu a snaží se, aby data v jednom balíčku byla relevantní k jednomu tématu – například textový popis bytu s jeho fotkami spolu se schématem rozloženi pokojů.

4.3.3 Input Logic Unit

Jednotka, která je sama o sobě inteligentní a převádí složitý text na jednoduché fráze a soustavy klíčových slov, kterým rozumí INPUT PROCESSING,

vyhledává relevantní obrázky a schémata ze stažených balíčků. Tyto informace jsou dále předány jednotce INPUT PROCESSING. Od ní zpětně získává data, z nichž generuje fráze a klíčová slova, která předává WEB jednotce. Tato jednotka by měla obsahovat modul, který bude umět převádět fotografie do 3D prostoru a do 2D schémat.

4.3.4 Associative memory

Jednoduchá asociativní paměť, která obsahuje data jako objekty (tzv. chunk) a jejich jednoduchou vazbu mezi dvěma chunky hodnotou od 0 do 1.
Chunky jsou například typu:
Room Chunk – parametricky popis rozložení pokojů, jejich sousednosti, ...
Building Chunk – parametricky popis objektu pomoci primitiv, rozměrů, materiálu
Terrain Chunk – parametricky popis krajiny
apod.

4.3.5 Input Processing

Modul, který v asociativní paměti klíčových vět, 2D, 3D objektu vytváří chunky a nové vazby.

4.3.6 Output Processing

Jednotka, která na základě modulu USER INPUT vyhledá ve vazbách asociativní paměti data a snaží se z nich definovat chunky, které spolu dobře “fungují”.

4.3.7 Output Generator, Output Tester

Vytvoří 3D model s parametry, který je možný posoudit na realističnost a použitelnost. To provede OUTPUT TESTER. Oba dva zpětně upravují vazby v asociativní paměti podle toho, jak se jim povedlo vygenerovat výstup. Vazby mezi chunky se v případě úspěchu posilují a naopak v případě neúspěchu zeslabují nebo ruší. OUTPUT TESTER také nabízí INPUT PROCESSINGU nové vhodné fráze,

nové podněty, nová klíčová slova a na základě toho, co si přál uživatel vygenerovat a na základě úspěšnosti výsledku.

4.3.8 Output 3D Model

Výsledek celého procesu ve formě trojrozměrného objektu, specifikacemi atd.

4.3.9 AI Permutator

Jednotka, která se snaží vytvářet nové kreace objektů, kombinace stávajících chunků – ať už stejného typu (genetické algoritmy) anebo rozdílných typů – nové kompozice, tyto kreace se ohodnotí CHUNK TESTEREM a v případě úspěchu se vloží/posílí vazby v asociativní paměti. Může také vytvářet úplně nové chunky a to právě mutací. CHUNK TESTER samozřejmě využívá schopnosti OUTPUT GENERATORu a OUTPUT TESTERu.

4.4 Vyhodnocení stavu, další vývoj

Některé subsystémy zmíněné v kapitole 4.2 je možné vyvíjet víceméně odděleně, každý se týká jiného oboru. Je nutné zhodnotit, v jakém pořadí na jednotlivých částech pracovat.

U prvních fází – vyhledávání dat a jejich překlad považuji za rozumnější prozatím vyčkat. Internetové vyhledavače již existují a na jejich vývoji se pracuje na celém světě. Se softwary pro rozpoznávání a pochopení jazyka je to podobné.

Na poslední fázi nemá smysl pracovat odděleně, dokud není co vyrábět.

Zbývá tak fáze návrhová a fáze vývojová. Vývojová fáze je přitom na návrhové zcela závislá.

Energii je tedy dle mého názoru třeba věnovat systému pro návrh, vývoji jednotného prostředí, jednotlivých algoritmů, geometrie.

5 Návrh CAD systému

Po sérii pokusů s parametrickým, genetickým a matematicky definovaným designem jsem se rozhodl vytvořit objekt, se kterým by šlo všemi těmito metodami pracovat. Objekt natolik obecný, aby jím bylo možné vyjádřit architektonické dílo.

5.1 Požadavky

- Široký okruh vstupní geometrie
- Tvarová přizpůsobivost objektu
- Parametrická přizpůsobivost
- Analýza vlastností
- Interaktivní reakce

5.2 Koncepce

1. Objekt bude primárně definován jako plocha v trojrozměrném prostoru.

Rozhodoval jsem se mezi třemi variantami.

Definovat objekt jednorozměrně (čárově) – z hlediska programování je tato varianta úsporná, čárově jdou poměrně dobře a komplexně vyjádřit pouze objekty definované ve dvourozměrném prostoru - např. dispoziční řešení, půdorys patra… U trojrozměrných objektů, a architektura je definována ve třech rozměrech, začíná být definice nedostatečná.

Další možností bylo vytvořit objekt plošný. Z hlediska programování geometrie a matematiky je tento přístup řádově složitější. Umožňuje vyjádřit prakticky kompletní škálu prostorových objektů, objekt lze snáze analyzovat, trojrozměrné objekty je možno kvalitně zobrazit. Plošná definice neumožňuje přímou práci s volumetrickými objekty.

Třetí variantou byla práce s volumetrickým objektem. Z hlediska množství užitečných informací, které by byly součástí objektu je tento přístup ideální, časová náročnost tvorby programu značná. Po několika pokusech s volumetrickými objekty jsem raději tuto variantu opustil. Nesplňovaly jeden ze základních požadavků –

nebyly dostatečně interaktivní. Lidský vstup do geometrie byl jen velmi omezený, komplikovaný a neumožňoval zpětnou vazbu (viz kapitola 2.6 - Datové struktury a vazby).

2. Plocha bude definována jako trojúhelníková síť.

Zase bylo na výběr více variant. Pro definici zakřivených ploch byla vyvinuta geometrie NURBS. Tyto plochy jsou bohužel zásadně definovány na základě řídící sítě ve čtvercové soustavě. Požadavky na objekt jsou výrazně obecnější oproti možnostem, které NURBS poskytují.

Rozhodl jsem se proto pro interpretaci plochy trojúhelníkovou sítí. Vnitřní struktura je na rozdíl od NURBS výrazně matematicky jednodušší a mohu ji od počátku definovat sám. Struktura je natolik jednoduchá, že prakticky nemá tvarová omezení.

3. Prostředí, programovací jazyk.

Výchozím zobrazovacím softwarem je Rhinoceros. Používám jej zvláště pro jeho otevřenou a zveřejněnou vnitřní strukturu, strukturu objektů, s nimiž pracuje.

Grasshopper – plug-in pro Rhinoceros, ideální prostředí pro práci s parametrickými objekty. Program se velmi rychle vyvíjí, programátoři jsou schopni a dokonce ochotni reagovat na nové požadavky uživatelů.

VBScript – jednoduchý a přehledný programovací jazyk. Je natolik jednoduchý, že téměř není třeba se jej učit, stačí psát. Grasshopper umožňuje ovládat objekty i pomocí VBScriptu.

5.3 Řešení, implemetnace

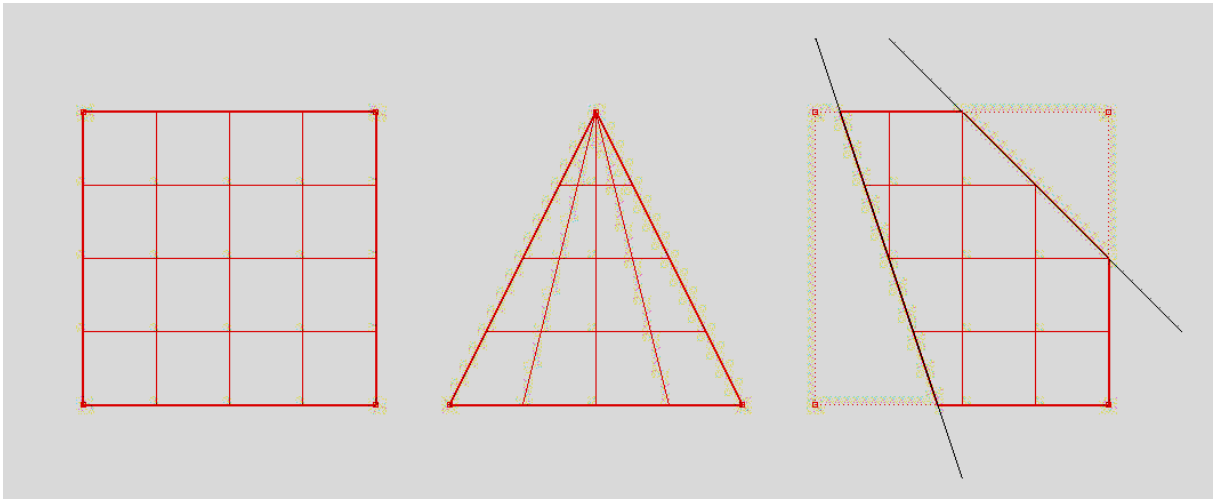
V této kapitole se pokusím zachytit vývoj objektu/plochy/softwaru tak jak vznikal na základě jednotlivých požadavků.

Jednotlivé kroky jsou zdokumentovány slovně, částmi kódu a obrazově na příkladech.

5.3.1 Vstupní geometrie

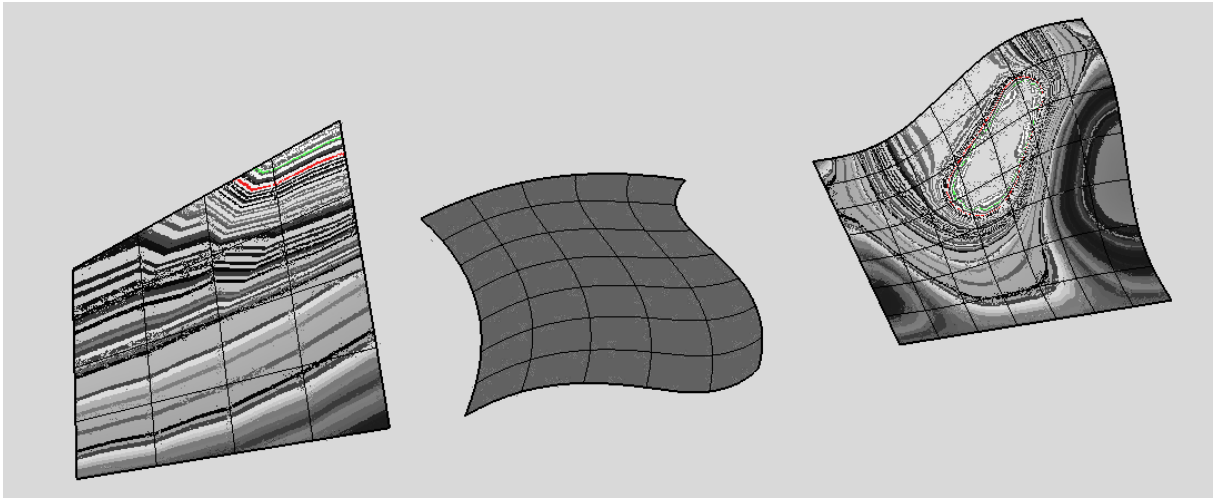
Na základě požadavku, aby byl okruh vstupní geometrie pokud možno co nejširší, obecný, jsem analyzoval možné geometrické vstupy.

Vzhledem k prostředí Rhinoceros jsou ideální a dostatečně obecné plochy NURBS. Ty potom dělím na plochy se základní definicí (plochy se čtyřmi hranami, odpovídajícími původním definičním hranám), plochy degradované (plochy se třemi hranami, ale definicí čtyř hran) a plochy modifikované (vznikají ořezáváním původních ploch, mají libovolný počet hran – definice zůstává čtyřhranná, a je mimo výslednou plochu). Viz obr. 179.



obr. 179 NURBS – 4E, 3E, Trim

Vstupní objekty jsou definovány jako NURBS plochy, to umožňuje práci s širokou škálou různě křivých ploch.



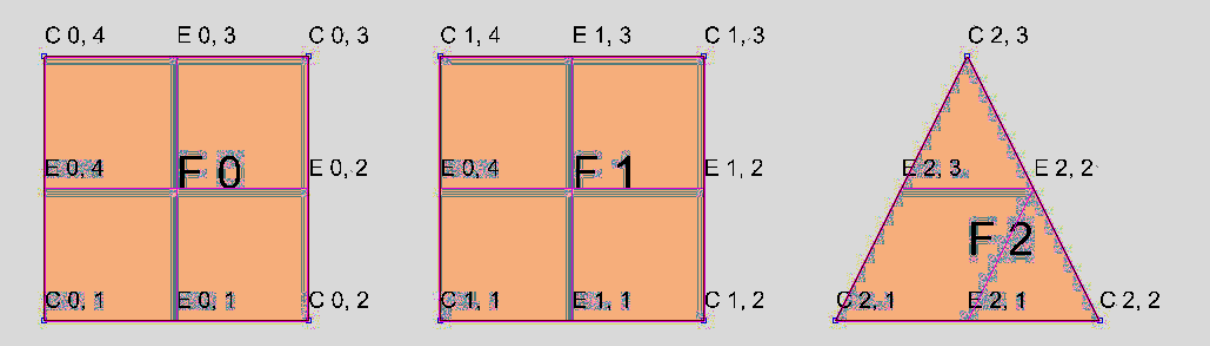
obr. 180 NURBS – křivé

5.3.2 Modifikace vstupů

Pro dostatečnou šíři definice vstupních objektů by pochopitelně nestačila jediná plocha, proto vzniká první algoritmus, který pracuje s více různými plochami a zapojuje je do jedné kompaktní struktury.

Jednotlivé plochy lze do struktury kdykoli přidávat a odebírat. Co je však mnohem důležitější, při modifikaci či vymazání vstupní plochy algoritmus automaticky reaguje na provedené změny.

Zde se začíná tvořit základní struktura budoucího objektu. Základní struktura je složena z jednotlivých vstupních ploch (Face / F) – primárního členění struktury. Každá plocha má čtyři hrany (Edge /E) a čtyři vrcholy (Corner – C). V případě trojúhelníku nejsou čtvrtý vrchol ani čtvrtá hrana používány ani zobrazeny.



obr. 181 Face, Edge, Corner, (FEC)

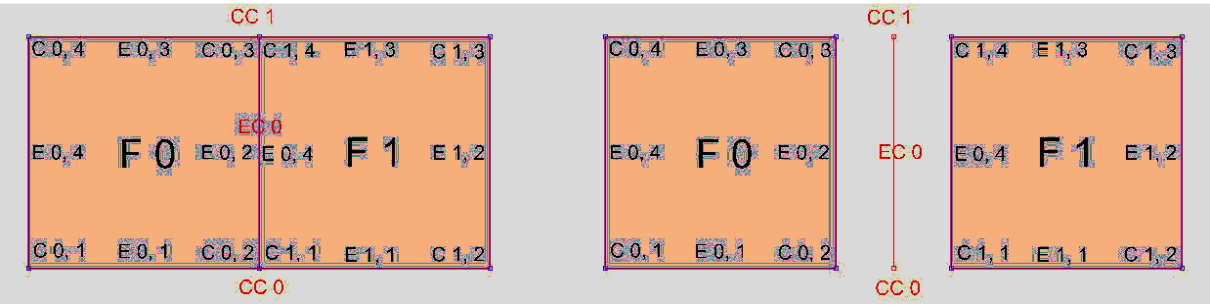
Faces, Edges a Corners jsou uloženy v programu Rhinoceros, Struktura je tvořena seznamy odkazů na tyto objekty a seznamu dalších informací (např. počet hran). Seznamy jsou ukládány externě ve formátu *.txt. Při přidání nebo odebrání plochy jsou seznamy aktualizovány.

Máme tedy definovány a naplněny mj. následující seznamy:

F_PID_List	(odkazy na původní objekty ploch v Rhinu)
E1_PID_List	(odkazy na vytvořené hrany 1)
E2_PID_List	(odkazy na vytvořené hrany 2)
E3_PID_List	(odkazy na vytvořené hrany 3)
E4_PID_List	(odkazy na vytvořené hrany 3)
C1_PID_List	(odkazy na vytvořené vrcholy 1)
C2_PID_List	(odkazy na vytvořené vrcholy 2)
C3_PID_List	(odkazy na vytvořené vrcholy 3)
C4_PID_List	(odkazy na vytvořené vrcholy 4)

5.3.3 Sousednost, konektivita

V případě, že se plochy dotýkají hranami, jsou automaticky propojeny ve struktuře novými vztahy. Jsou vytvořeny hrany a vrcholy, kde se plochy stýkají. (Edge Connection / EC a Corner Connection / CC). Na obr. 182 je znázorněn styk dvou ploch přesný a nepřesný. Je umožněno propojit i plochy, které na sebe nesedí zcela přesně, přesnost lze stanovit (vzájemná vzdálenost hran)



obr. 182 EC, CC

Struktura doplněna o další seznamy – EC a CC

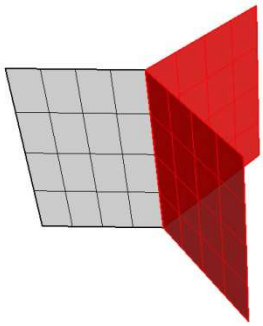
EC_Geo_List	(geometrie hrany)
CC_Geo_List	(geometrii - pozice vrcholu)

a seznamy vazeb mezi EC a E, CC a C a naopak

EC_E_Nr_List	(pozice v seznamu hran)
EC_E_Dir_List	(pořadí hrany v ploše (1-4) a směr hrany (+/-) – spojené hrany mohou mít opačný směr)
E1_EC_Nr_List	(odkaz na spojení hran)
E2_EC_Nr_List	… E3, E4
C1_CC_Nr_ List	(odkaz vrcholu plochy na společný vrchol)
C2_CC_Nr_ List	… C3, C4
CC_C_Nr_Set_List	(pro každý společný vrchol existuje seznam rohů ploch, které se v něm stýkají)
CC_C_Dir_Set_List	(pořadí vrcholů v ploše (1-4))

Při propojování ploch je nutné vyřešit několik zásadních požadavků.

- Nemohou se stýkat více než dvě plochy v jedné hraně (nutno vyloučit ze spojování)



obr. 183 EC – 3E

Implementace spojování hran EC:

```
print("   7.1 - Start create new connections EC (add/rh.E  x  All.E.Border)")

Dim NEC_Nr_List As New list (Of Int32)' list for mesh connecting

Dim NCC_Geo_List As New list (Of Point3d)
Dim NCC_CA_Nr_List As New list (Of Int32)
Dim NCC_CA_Dir_List As New List (Of Int16)
Dim NCC_CB_Nr_List As New list (Of Int32)
Dim NCC_CB_Dir_List As New List (Of Int16)

For i = 0 To ChangeFaceList.count - 1

  Dim ChangeFaceA As String = ChangeFaceList(i)

  If ChangeFaceA.Contains("CHANGE_") And ((ChangeFaceA.Contains("_ADD_STATUS") Or (Not ChangeFaceA.Contains("_STATUS"))))Then

    print("      Start Face A: " & i)

    Dim Curve_A_List As New List (Of Curve)
    Dim Curve_A As CURVE

    Curve_A = E1_Geo_List(FacePos_to_EdgePos_List(i))
    Curve_A_List.Add(Curve_A)
    Curve_A = E2_Geo_List(FacePos_to_EdgePos_List(i))
    Curve_A_List.Add(Curve_A)
    Curve_A = E3_Geo_List(FacePos_to_EdgePos_List(i))
    Curve_A_List.Add(Curve_A)
    If F_E_Count_List(FacePos_to_EdgePos_List(i)) = 4 Then
      Curve_A = E4_Geo_List(FacePos_to_EdgePos_List(i))
      Curve_A_List.Add(Curve_A)
    End If

    Dim ii As Int16

    print("        -Number of edges in face A: " & Curve_A_List.Count)

    For ii = 0 To Curve_A_List.Count - 1

      print("         Start Edge A: " & ii + 1)

      Curve_A = Curve_A_List(ii)

      Dim E_A_SP As point3d ' start point of first edge (i)
      Dim E_A_CP1 As point3d ' point of first edge in one third (i)
      Dim E_A_CP2 As point3d ' point of first edge in two thirds (i)
      Dim E_A_EP As point3d ' end point of first edge (i)

      E_A_SP = Curve_A.PointAtStart
      E_A_CP1 = Curve_A.PointAtNormalizedLength(0.3456789)
      E_A_CP2 = Curve_A.PointAtNormalizedLength(0.6543211)
      E_A_EP = Curve_A.PointAtEnd


      Dim STOP_J As Int16 = 0
      j = 0 ' !!!!!!!
      Do 'all edges 1234 AND border

        Dim ChangeFaceB As String = ChangeFaceList(j)

        If Not (ChangeFaceB.Contains("REM") Or ChangeFaceB.Contains("DEL") Or (i = j) Or F_Pid_List(j).Contains("_STATUS"))  Then '
        if seccond edge is not (del or rem) then try connection

          print("           Start Face B: " & j)

          Dim jj As Int16

          Dim Curve_B_List As New List (Of Curve)
          Dim Curve_B_E_Nr_List As New List (Of Int16)

          Dim Curve_B As CURVE
          Dim Curve_B_E_Nr As Int16

          If (E1_Con_EdgeNr_List(FacePos_to_EdgePos_List(j))) = -1  Then ' if seccond edge is border then try connection
            Curve_B = E1_Geo_List(FacePos_to_EdgePos_List(j))
            Curve_B_List.add(Curve_B)
            Curve_B_E_Nr = 1
            Curve_B_E_Nr_List.Add(Curve_B_E_Nr)
          End If
          If (E2_Con_EdgeNr_List(FacePos_to_EdgePos_List(j))) = -1  Then ' if seccond edge is border then try connection
            Curve_B = E1_Geo_List(FacePos_to_EdgePos_List(j))
            Curve_B_List.add(Curve_B)
            Curve_B_E_Nr = 2
            Curve_B_E_Nr_List.Add(Curve_B_E_Nr)
          End If
          If (E3_Con_EdgeNr_List(FacePos_to_EdgePos_List(j))) = -1  Then ' if seccond edge is border then try connection
            Curve_B = E1_Geo_List(FacePos_to_EdgePos_List(j))
            Curve_B_List.add(Curve_B)
            Curve_B_E_Nr = 3
            Curve_B_E_Nr_List.Add(Curve_B_E_Nr)
          End If
          If  F_E_Count_List(FacePos_to_EdgePos_List(j)) = 4 Then
            If (E4_Con_EdgeNr_List(FacePos_to_EdgePos_List(j))) = -1  Then ' if seccond edge is border then try connection
              Curve_B = E1_Geo_List(FacePos_to_EdgePos_List(j))
              Curve_B_List.add(Curve_B)
              Curve_B_E_Nr = 4
              Curve_B_E_Nr_List.Add(Curve_B_E_Nr)
```

```
  End If
End If

print("              -Number of edges in face B: " & Curve_B_List.Count)

For jj = 0 To Curve_B_List.Count - 1

  If Not STOP_J = 1 Then

    If Curve_B_E_Nr_List(jj) = 1 Then
      curve_B = E1_Geo_List(FacePos_to_EdgePos_List(j))
    ElseIf Curve_B_E_Nr_List(jj) = 2 Then
      curve_B = E2_Geo_List(FacePos_to_EdgePos_List(j))
    ElseIf Curve_B_E_Nr_List(jj) = 3 Then
      curve_B = E3_Geo_List(FacePos_to_EdgePos_List(j))
    ElseIf Curve_B_E_Nr_List(jj) = 4 Then
      curve_B = E4_Geo_List(FacePos_to_EdgePos_List(j))
    Else
      print("!!! Curve_B_E_Nr_List(jj) is wrong")
    End If

    Dim E_B_SP As point3d ' start point of seccond edge (j)
    Dim E_B_CP1 As point3d ' point of seccond edge in one third (j)
    Dim E_B_CP2 As point3d ' point of seccond edge in two thirds (j)
    Dim E_B_EP As point3d ' end point of seccond edge (j)

    E_B_SP = Curve_B.PointAtStart
    E_B_CP1 = Curve_B.PointAtNormalizedLength(0.3456789)
    E_B_CP2 = Curve_B.PointAtNormalizedLength(0.6543211)
    E_B_EP = Curve_B.PointAtEnd

    Dim EC_DIR As Int16 = 0

    If(  (  E_A_SP.DistanceTo(E_B_SP)  <  DistTolerance)  And  (  E_A_CP1.DistanceTo(E_B_CP1)  <  DistTolerance)  And  (
E_A_CP2.DistanceTo(E_B_CP2) < DistTolerance) And (E_A_EP.DistanceTo(E_B_EP) < DistTolerance)) Then ' is connected in dir
      EC_DIR = 1
    ElseIf(  (  E_A_SP.DistanceTo(E_B_EP)  <  DistTolerance)  And  (  E_A_CP1.DistanceTo(E_B_CP2)  <  DistTolerance)  And  (
E_A_CP2.DistanceTo(E_B_CP1) < DistTolerance) And (E_A_EP.DistanceTo(E_B_SP) < DistTolerance)) Then ' is connected in opp
      EC_DIR = -1
    Else
      EC_DIR = 0
    End If

    If EC_DIR = 1 Or EC_DIR = -1 Then ' make connection in dir or in opposite

      STOP_J = 1 ' if once connected never try again

      'create EC curve

      Dim ECNr As Int32 = EC_Pid_List.count
      Dim EC_Geo As Curve = Curve_A
      Dim EC_Pid As String
      Dim EC_Id As guid
      Dim att As Rhino.DocObjects.ObjectAttributes = doc.CreateDefaultAttributes()
      att.ColorSource = Rhino.DocObjects.ObjectColorSource.ColorFromObject
      att.ObjectColor = ConnectColor
      att.Name = ("CONNECTEDGE_" & ECNr & "_F" & i & "E" & ii + 1 & " x F" & j & "E" & Curve_B_E_Nr_List(jj) )
      EC_Id = doc.Objects.AddCurve(EC_Geo, att)
      EC_Pid = EC_Id.ToString

      Dim EC_EA_Nr As Int32 = FacePos_to_EdgePos_List(i)
      Dim EC_EB_Nr As Int32 = FacePos_to_EdgePos_List(j)
      Dim EC_EA_Dir As Int16 = (ii + 1)
      Dim EC_EB_Dir As Int16 = Curve_B_E_Nr_List(jj) * EC_DIR ' +- direction

      'add EC to list

      EC_Geo_List.add(EC_Geo)
      EC_Pid_List.Add(EC_Pid)
      EC_EA_Nr_List.Add(EC_EA_Nr)
      EC_EB_Nr_List.Add(EC_EB_Nr)
      EC_EA_Dir_List.Add(EC_EA_Dir)
      EC_EB_Dir_List.Add(EC_EB_Dir)

      Dim NEC_Nr As int32
      NEC_Nr = EC_Pid_List.Count - 1
      NEC_Nr_List.add(NEC_Nr)

      'change data in edge connection lists

      Dim E_EC_Nr As Int32
      E_EC_Nr = EC_Pid_List.Count - 1' new EC is at the end of list

      If ii = 0 Then ' EA = 1
        'EA-EC
        E1_EC_Nr_List(FacePos_to_EdgePos_List(i)) = E_EC_Nr
        E1_EC_Dir_List(FacePos_to_EdgePos_List(i)) = 1  ' + first edge is always in direction
        'EA-EB
        E1_Con_EdgeNr_List(FacePos_to_EdgePos_List(i)) = (FacePos_to_EdgePos_List(j))
        E1_Con_EdgeDir_List(FacePos_to_EdgePos_List(i)) = Curve_B_E_Nr_List(jj) * EC_DIR' + as in direction
      ElseIf ii = 1 Then ' EA = 2
        'EA-EC
        E2_EC_Nr_List(FacePos_to_EdgePos_List(i)) = E_EC_Nr
        E2_EC_Dir_List(FacePos_to_EdgePos_List(i)) = 1  ' + first edge is always in direction
        'EA-EB
        E2_Con_EdgeNr_List(FacePos_to_EdgePos_List(i)) = (FacePos_to_EdgePos_List(j))
        E2_Con_EdgeDir_List(FacePos_to_EdgePos_List(i)) = Curve_B_E_Nr_List(jj) * EC_DIR' + as in direction
      ElseIf ii = 2 Then' EA = 3
        'EA-EC
```

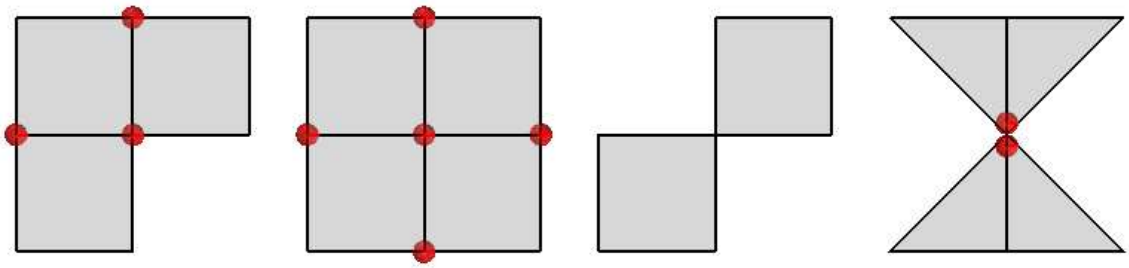
```
  E3_EC_Nr_List(FacePos_to_EdgePos_List(i)) = E_EC_Nr
  E3_EC_Dir_List(FacePos_to_EdgePos_List(i)) = 1  ' + first edge is always in direction
  'EA-EB
  E3_Con_EdgeNr_List(FacePos_to_EdgePos_List(i)) = (FacePos_to_EdgePos_List(j))
  E3_Con_EdgeDir_List(FacePos_to_EdgePos_List(i)) = Curve_B_E_Nr_List(jj) * EC_DIR' + as in direction
ElseIf ii = 3 Then' EA = 4
  'EA-EC
  E4_EC_Nr_List(FacePos_to_EdgePos_List(i)) = E_EC_Nr
  E4_EC_Dir_List(FacePos_to_EdgePos_List(i)) = 1  ' + first edge is always in direction
  'EA-EB
  E4_Con_EdgeNr_List(FacePos_to_EdgePos_List(i)) = (FacePos_to_EdgePos_List(j))
  E4_Con_EdgeDir_List(FacePos_to_EdgePos_List(i)) = Curve_B_E_Nr_List(jj) * EC_DIR' + as in direction
End If

If Curve_B_E_Nr_List(jj) = 1 Then ' EB = 1
  'EB-EC
  E1_EC_Nr_List(FacePos_to_EdgePos_List(j)) = E_EC_Nr
  E1_EC_Dir_List(FacePos_to_EdgePos_List(j)) = EC_DIR
  'EB-EA
  E1_Con_EdgeNr_List(FacePos_to_EdgePos_List(j)) = (FacePos_to_EdgePos_List(i))
  E1_Con_EdgeDir_List(FacePos_to_EdgePos_List(j)) = (ii + 1) * EC_DIR' + as in direction
Else If Curve_B_E_Nr_List(jj) = 2 Then' EB = 2
  'EB-EC
  E2_EC_Nr_List(FacePos_to_EdgePos_List(j)) = E_EC_Nr
  E2_EC_Dir_List(FacePos_to_EdgePos_List(j)) = EC_DIR
  'EB-EA
  E2_Con_EdgeNr_List(FacePos_to_EdgePos_List(j)) = (FacePos_to_EdgePos_List(i))
  E2_Con_EdgeDir_List(FacePos_to_EdgePos_List(j)) = (ii + 1) * EC_DIR' + as in direction
Else If Curve_B_E_Nr_List(jj) = 3 Then' EB = 3
  'EB-EC
  E3_EC_Nr_List(FacePos_to_EdgePos_List(j)) = E_EC_Nr
  E1_EC_Dir_List(FacePos_to_EdgePos_List(j)) = EC_DIR
  'EB-EA
  E3_Con_EdgeNr_List(FacePos_to_EdgePos_List(j)) = (FacePos_to_EdgePos_List(i))
  E3_Con_EdgeDir_List(FacePos_to_EdgePos_List(j)) = (ii + 1) * EC_DIR' + as in direction
Else If Curve_B_E_Nr_List(jj) = 4 Then' EB = 4
  'EB-EC
  E4_EC_Nr_List(FacePos_to_EdgePos_List(j)) = E_EC_Nr
  E4_EC_Dir_List(FacePos_to_EdgePos_List(j)) = EC_DIR
  'EB-EA
  E4_Con_EdgeNr_List(FacePos_to_EdgePos_List(j)) = (FacePos_to_EdgePos_List(i))
  E4_Con_EdgeDir_List(FacePos_to_EdgePos_List(j)) = (ii + 1) * EC_DIR' + as in direction
End If

'create NCC points on start and end of EC

Dim NCCS_Geo As point3d = Curve_A.PointAtStart
Dim NCCS_CA_Nr As int32 = FacePos_to_EdgePos_List(i)
Dim NCCS_CA_dir As int16 = ii + 1

Dim NCCS_CB_Nr As int32 = FacePos_to_EdgePos_List(j)
Dim NCCS_CB_dir As int16
If EC_DIR = 1 Then
  NCCS_CB_dir = Curve_B_E_Nr_List(jj)
ElseIf EC_DIR = -1 Then
  If Curve_B_E_Nr_List(jj) = 3 And F_E_Count_List(FacePos_to_EdgePos_List(j)) = 3 Then
    NCCS_CB_dir = 1
  Else If Curve_B_E_Nr_List(jj) = 4 Then
    NCCS_CB_dir = 1
  Else
    NCCS_CB_dir = Curve_B_E_Nr_List(jj) + 1
  End If
End If

Dim NCCE_Geo As point3d = Curve_A.PointAtEnd
Dim NCCE_CA_Nr As int32 = FacePos_to_EdgePos_List(i)
Dim NCCE_CA_dir As int16
If ii + 1 = 3 And F_E_Count_List(FacePos_to_EdgePos_List(i)) = 3 Then
  NCCE_CA_dir = 1
Else If ii + 1 = 4 And F_E_Count_List(FacePos_to_EdgePos_List(i)) = 4 Then
  NCCE_CA_dir = 1
Else
  NCCE_CA_dir = ii + 2
End If

Dim NCCE_CB_Nr As int32 = FacePos_to_EdgePos_List(j)
Dim NCCE_CB_dir As int16
If EC_DIR = 1 Then
  If Curve_B_E_Nr_List(jj) = 3 And F_E_Count_List(FacePos_to_EdgePos_List(j)) = 3 Then
    NCCE_CB_dir = 1
  Else If Curve_B_E_Nr_List(jj) = 4 And F_E_Count_List(FacePos_to_EdgePos_List(j)) = 4 Then
    NCCE_CB_dir = 1
  Else
    NCCE_CB_dir = Curve_B_E_Nr_List(jj) + 1
  End If
ElseIf EC_DIR = -1 Then
  NCCE_CB_dir = Curve_B_E_Nr_List(jj)
End If

' add NCC to lists

NCC_Geo_List.add(NCCS_Geo)
NCC_CA_Nr_LIst.add(NCCS_CA_Nr)
NCC_CA_dir_list.add(NCCS_CA_dir)
NCC_CB_Nr_list.add(NCCS_CB_Nr)
NCC_CB_dir_list.add(NCCS_CB_dir)

NCC_Geo_List.add(NCCE_Geo)
NCC_CA_Nr_LIst.add(NCCE_CA_Nr)
```

```
                NCC_CA_dir_list.add(NCCE_CA_dir)
                NCC_CB_Nr_list.add(NCCE_CB_Nr)
                NCC_CB_dir_list.add(NCCE_CB_dir)

              End If ' make connection in dir or in opposite
              print("               End Edge B: " & Curve_B_E_Nr_List(jj))
            End If'  If Not Stop_J = 1 Then
          Next '(jj) next BE
        End If '  if seccond edge is not (del or rem) then try connection
        j = j + 1 '!!!!!!!!!!!!!!!!!!!
      Loop Until (j = ChangeFaceList.count Or STOP_J = 1)    '  next B if A is not connected
      print("         End Edge A: " & ii + 1)
    Next' ii (next A - 1 or 2 or 3 or 4)
    print("      End Face A: " & i)
  End If 'A = add or rh
Next ' next A
print("   7.1 End create new connections EC (add/rh.E  x  All.E.Border)")
```

Příklady implementace dále budu stať zatěžovat jen výjimečně. Jednak pro jejich obtížnou srozumitelnost pro kohokoli vyjma autora, potom také, protože celý kód má aktuálně cca 15000 řádků.

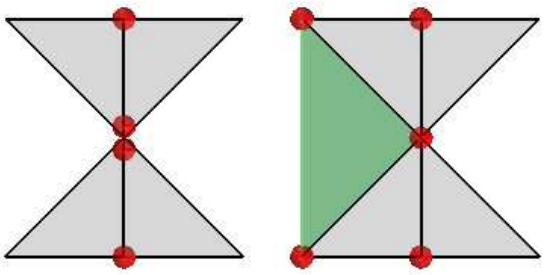
Další z podmínek:

- Je třeba ošetřit množství vrcholů (CC) v jednom místě. Nastává množství různých variant, pro každou je třeba nalézt řešení.



obr. 184 CC – varianty

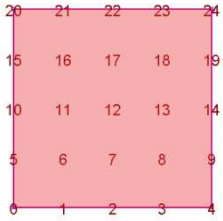
Při přidávání a ubírání ploch se počty vrcholů různě mění. Problematické je zvláště ošetření podmínky, která zakazuje spojení ploch jediným bodem. Díky tomu mohou být v jednom bodě definovány dva různé vrcholy. Pakliže je plocha následně propojena hranami, vrcholy se sloučí v jeden. Opačně při ubírání ploch. Viz obr. 185.

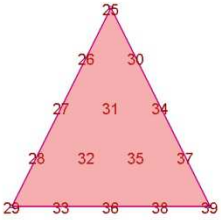


obr. 185 CC – var. 12

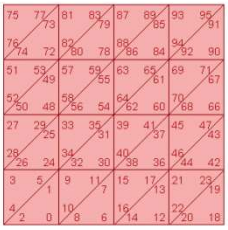
5.3.4 Polygonální síť

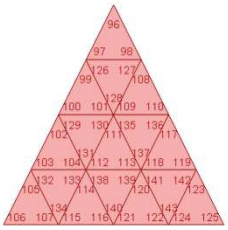
Na základě vstupní geometrie (NURBS Plochy) je vygenerována na každé ploše polygonální – trojúhelníková síť (Mesh). Struktura je tvořena vrcholy (MV) ve kterých se potkávají rohy jednotlivých trojúhelníků (MC).





obr. 186 MV





obr. 187 MC

Body jsou vygenerovány na základě okrajových podmínek vstupních NURBS ploch (Face). Byl zvolen jiný algoritmus pro polygony o čtyřech hranách a jiný pro trojúhelníkové. U trojúhelníků tím byl odstraněn problém se čtvrtou, potlačenou hranou v definici NURBS. U čtverců bylo možné použít v algoritmu přímo definici NURBS plochy, u pseudotrojúhelníkových ploch jsem byl nucen vyvinout nové diferenciální rovnice plochy s vícesměrným (3) pohybem v ploše.

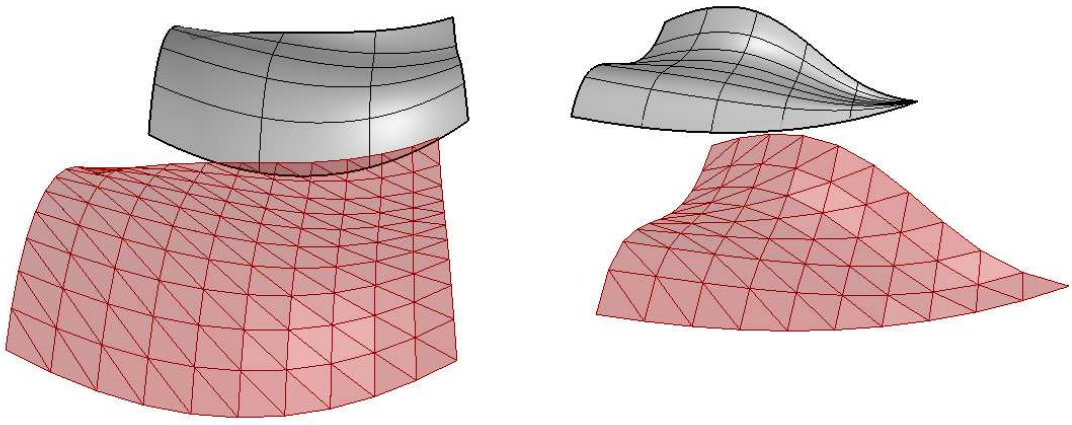
Geneze bodů ve čtvercové ploše:

```
For ii = 0 To M_G
   For jj = 0 To M_G

      Dim MV_Geo As point3d

      Dim Vec As Vector3d

      Dim VecU As Vector3d
      Dim VecV As Vector3d

      Dim VecU_E1 As Vector3d
      Dim VecU_E3 As Vector3d
      Dim VecV_E4 As Vector3d
      Dim VecV_E2 As Vector3d

      VecU_E1 = Crv1.PointAtNormalizedLength(jj / M_G) - Crv1.PointAtStart
      VecU_E3 = Crv3.PointAtNormalizedLength((M_G - jj) / M_G) - Crv3.PointAtEnd
      VecU = (VecU_E1 * (M_G - ii) + VecU_E3 * ii) / M_G

      VecV_E4 = Crv4.PointAtNormalizedLength((M_G - ii) / M_G) - Crv4.PointAtEnd
      VecV_E2 = Crv2.PointAtNormalizedLength(ii / M_G) - Crv2.PointAtStart
      VecV = (VecV_E4 * (M_G - jj) + VecV_E2 * jj) / M_G

      Vec = VecU + VecV

      MV_Geo = New point3d(Vec + Crv1.PointAtStart)

   Next
Next
```

Geneze bodů ve trojúhelníkové ploše:

```
For ii = 0 To M_G
   For jj = 0 To M_G – ii

      Dim MV_Geo As point3d

      Dim VecAB As Vector3d
      Dim VecAC As Vector3d
      Dim VecBA As Vector3d
      Dim VecBC As Vector3d
      Dim VecCA As Vector3d
      Dim VecCB As Vector3d

      Dim VecA As Vector3d
      Dim VecB As Vector3d
      Dim VecC As Vector3d

      Dim MagA As Double
      Dim MagB As Double
      Dim MagC As Double

      Dim VecFin As Vector3d

      VecA = Crv1.PointAtStart
      VecB = Crv2.PointAtStart
      VecC = Crv3.PointAtStart

      VecAB = Crv1.PointAtNormalizedLength(jj / M_G) - VecA
      VecAC = Crv3.PointAtNormalizedLength((M_G - ii) / M_G) - VecA
      VecBA = Crv1.PointAtNormalizedLength((ii + jj) / M_G) - VecB
      VecBC = Crv2.PointAtNormalizedLength(ii / M_G) - VecB
      VecCA = Crv3.PointAtNormalizedLength((M_G - ii - jj) / M_G) - VecC
      VecCB = Crv2.PointAtNormalizedLength((M_G - jj) / M_G) - VecC

      MagA = (M_G - ii - jj) / M_G
      MagB = jj / M_G
      MagC = ii / M_G

      VecFin = MagA * (VecAB + VecAC + VecA) + MagB * (VecBA + VecBC + VecB) + MagC * (VecCA + VecCB + VecC)

      MV_Geo = New Point3d(VecFin)

   Next
Next
```

Nyní je převedena většina NURBS ploch do podoby parametricky rovnoměrných trojúhelníkových sítí.

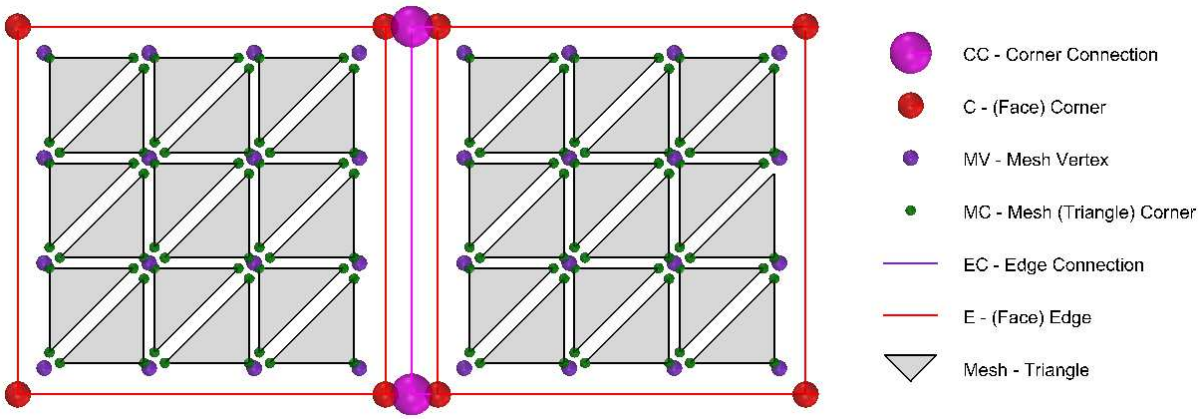


obr. 188 Mesh Generate

Vzniklé body jsou zapojeny do struktury a jsou doplněny oboustranné vazby mezi sítí a vstupními objekty.

Doplněné seznamy:

MV_Geo_List	poloha vrcholů
MV_FEC_Type_List	informace, na jakém typu objektu (F,E,C,EC,CC) vrchol leží
MV_FEC_Nr_List	číslo objektu
MV_FEC_Dir_List	pořadí (pro hrany a vrcholy)
MV_Rest_Set_List	seznam důležitých informací o chování bodu – řídí budoucí pohyb bodu v prostoru
F_MV_Nr_Set_List	seznam vrcholů ležících uvnitř plochy
E1_MV_Nr_Set_List	seznam vrcholů ležících na hraně , E2…, E3, E4
C1_MV_Nr_ List	vrcholy ležící v rohu, C2…, C3, C4
EC_MV_Nr_Set_List	seznam vrcholů na spojení ploch
CC_MV_Nr_List	vrcholy ležící na spojujících rozích
MC_Dir_List	pořadí vrcholů v trojúhelnících
MC_OrFEC_Type_List	původní typ objektu na němž roh trojúhelníku leží (před spojením – důležité při odebírání ploch)
MC_OrFEC_Nr_List	číslo objektu
MC_OrFEC_Dir_List	pořadí (pro hrany a rohy)
MC_MV_Nr_List	Poloha – odkaz na MV – vrchol, na kterém leží roh trojúhelníka
MF_Rest_Set_LIst	Seznam o důležitých informacích o chování trojúhelníků – sdružuje analytické informace o ploše
F_MC_Nr_Set_List	seznam trojúhelníků v ploše
MV_MC_Nr_Set_List	seznam rohů trojúhelníků ležících na vrcholu MV

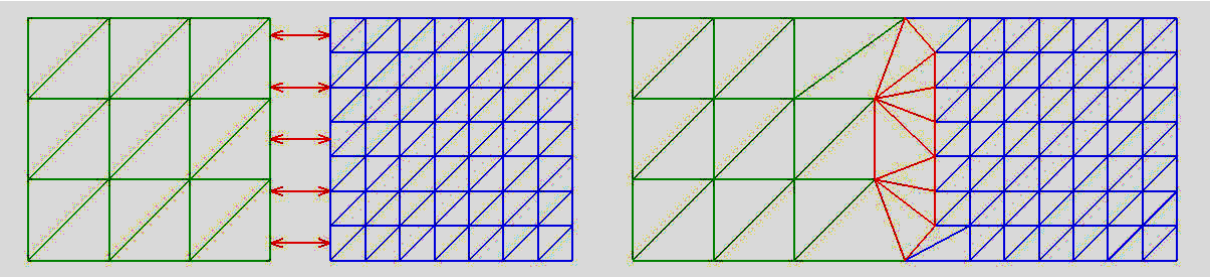


obr. 189 Struktura - schéma

5.3.5 Spojení dvou sítí

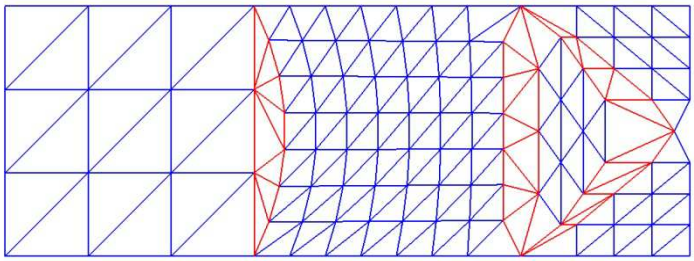
Tak jako byly propojeny vstupní plochy v datové struktuře, nyní je zapotřebí propojit i vytvořené sítě. Jednotlivé plochy mohou mít síť různě deformovanou, mezi hranicemi ploch jsou tolerovány menší mezery – nepřesnosti, případně může být i síť v průběhu práce odsunuta mimo původní hranice plochy. Z toho důvodu jsem napsal trochu složitější algoritmus, který mezi dvě plochy (obecně starou a novou) vytvoří

novou síť, kterou se obě plochy propojí. Nová síť se poté stává součástí jedné z ploch.



obr. 190 Mesh - Connect

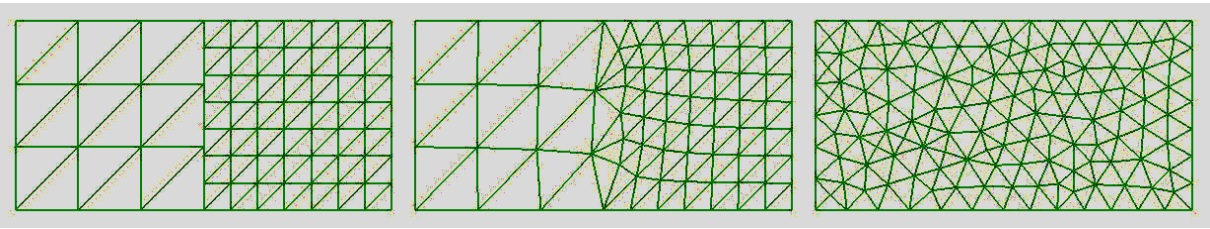
Algoritmus je napsán tak, aby vyhledával nejmenší možné trojúhelníky mezi dvěma hranami. Nejmenší nejsou reálné délky hran ale parametrické vzdálenosti. Algoritmus je tak schopen vyhledat i prostorově komplikovaná řešení a nevadí mu ani vzdálenosti nulové.



obr. 191 Mesh – Connect 2

5.3.6 Regenerace polygonální sítě (ME_Short, ME_Long)

V mnoha případech nastáva situace, kdy je trojúhelníková síť velmi nerovnoměrná. Některé algoritmy vyžadují alespoň přibližnou rovnoměrnost celé sítě, proto jsem vyvinul soustavu několika algoritmů, které v libovolný moment celou síť optimalizují.

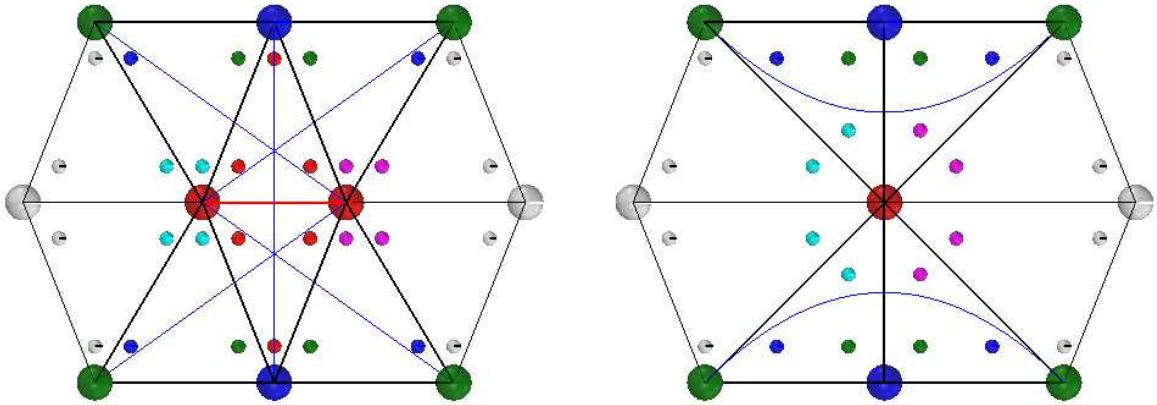


obr. 192 Mesh Regen

Na obr. 192 je zachycena původní síť, jeden z mezikroků a síť optimalizovaná.

Základem optimalizace jsou dva algoritmy. První vyhledává trojúhelníky s příliš dlouhými hranami, které rozdělí. Druhý naopak vyhledá trojúhelníky s příliš krátkými hranami, které ze sítě eliminuje. Geometricky lze tento problém řešit poměrně snadno, problém spočívá v komplexitě struktury. Struktura obsahuje velké množství pevných vazeb, které je nutné při změnách nahradit.

5.3.6.1 ME_Short – Eliminace trojúhelníků s krátkými hranami



obr. 193 Short Mesh Edge - diagram

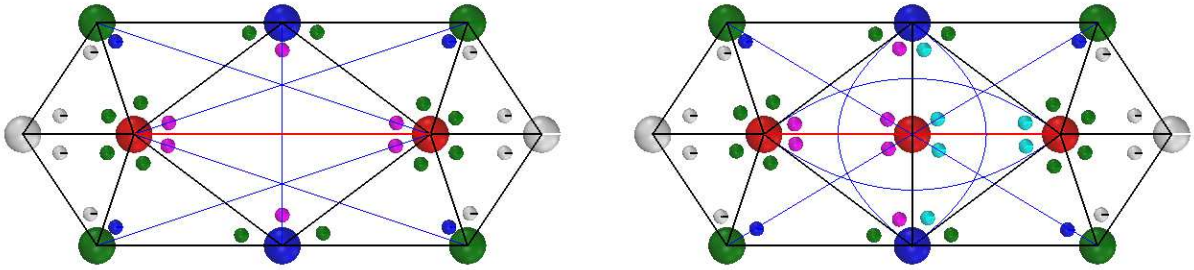
Je vyhledána krátká hrana trojúhelníka, jsou vyhledány všechny okolní vrcholy, všechny vazby. Trojúhelníky s vyhledanou krátkou hranou jsou odstraněny, vrcholy při krátké hraně sloučeny, všechny původní vazby jsou nahrazeny novými.

Na diagramu 192. Je naznačena geometrie, které se změna jedné hrany dotýká. Jsou naznačeny objekty na úrovni Mesh, u nichž je nutné měnit vazby. Je pochopitelně nutné, aby se tyto změny projevily i ve vazbách na původní geometrii, což vytváří mnoho různých variant, které je nutno ošetřit.

Na obr. 192 jsou modře vyznačeny důležité horizontální vztahy sousednosti mezi trojúhelníky. Tyto vazby jsou později různými algoritmy využívány pro analýzu křivosti ploch nebo šíření vlastností do okolí.

5.3.6.2 ME_Long – Eliminace trojúhelníků s dlouhými hranami

Obdobný algoritmus řeší opačný problém. Vyhledává dlouhé hrany a dva sousední trojúhelníky nahrazuje čtyřmi s kratšími hranami. Znovu jsou nahrazeny všechny dotčené vazby.



obr. 194 Long Mesh Edge - diagram

5.3.6.3 Limitní délky hran

Oba předchozí algoritmy jsou řízeny nadřazeným systémem, který umožňuje definovat minimální a maximální délku hrany v síti. Na základě těchto hranic střídavě spouští algoritmus ME_Long a ME_Short, tak aby v co nejkratší době vznikla optimalizovaná síť. Algoritmus je založen na iteraci a postupném přibližování hranic délek hran trojúhelníku stanovenému ideálu.

5.3.7 Omezení, vlastnosti, pravidla

Pro každý vrchol (MV) v síti a pro každý trojúhelník je definován seznam pravidel, podle nichž se poté řídí jednotlivé algoritmy, které s předdefinovanou sítí pracují.

Prvním pravidlem – vlastností, které bylo definováno, je informace zdali se vrchol nachází na okraji plochy. Tato informace nám například umožňuje modifikovat celou síť, ale zachovat původní hranice objektu.

Další pravidla pro vrcholy nebo trojúhelníky mohou být doplňována v průběhu práce podle potřeb jednotlivých algoritmů. Pravidla se mohou definovat buď ručně – zadáním pro určité skupiny bodů, mohou být generována automaticky (např. informace o hranici), nebo jsou definována a upravována pomocí algoritmů, které si síť přizpůsobují pro vlastní potřebu.

5.3.8 Iterace, aproximace (run)

Vlastní algoritmy pro práci s předefinovaným objektem jsou založeny na posouvání jednotlivých bodů do polohy, která vyhovuje zadání.

Proces je založen na iteraci. V každém kroku je vyhodnocen směr pohybu každého bodu k ideálnímu stavu a body jsou posunuty o malý kousek. Obdobně se šíří i vlastnosti po ploše. V každém kroku se informace pokusí rozšířit na okolní trojúhelníky a je vyhodnoceno, je li to v souladu se záměrem.

Po dostatečném počtu kroků je tak aproximován ideální výsledek. Celý proces je otevřený pro libovolné úpravy, na vygenerované řešení je možné aplikovat další, jiný algoritmus a objekt tak tvořit i postupně.

5.4 Funkce, algoritmy

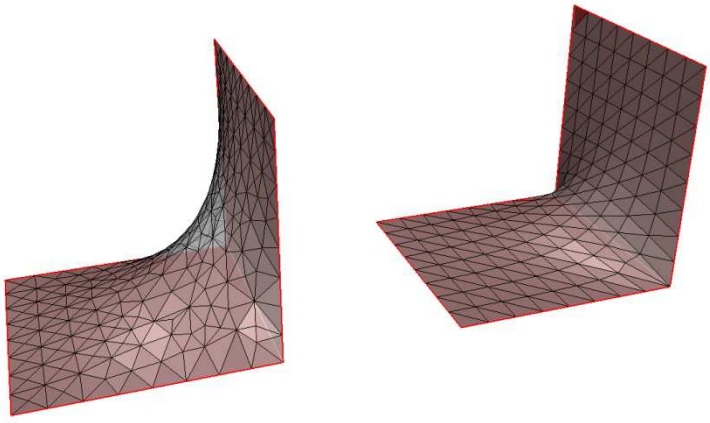
V minulé kapitole byly naznačeny principy fungování vytvořeného objektu obecně. V této kapitole se dostáváme k reálným příkladům. Představím zde několik testovacích algoritmů a jejich aplikaci na objekt.

Do této chvíle naprogramovanou strukturu obecného objektu je třeba chápat jako jádro programu. Nyní jsou pouze přidávany jednotlivé funkce, aniž by bylo nutné na dosavadní práci cokoli měnit.

5.4.1 Minimální plocha

Pro ukázku fungování tvorby objektu, je nejjednodušším zadáním požadovat plochu minimální. V rámci okrajových neměnných podmínek (hranice plochy) je vygenerována plocha s minimálním obsahem.

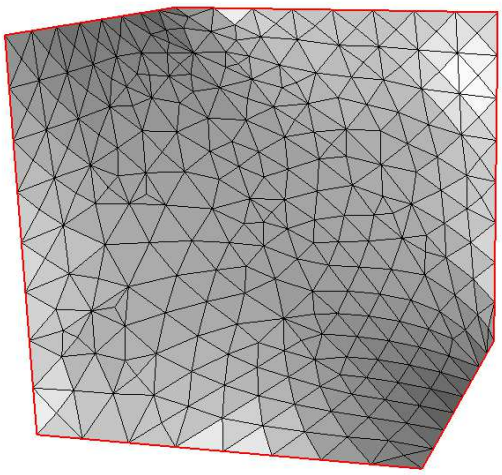
Existují dvě metody – plocha je definována jako drátová síť a minimalizují se délky drátů. Druhá metoda pracuje s plochami a minimalizuje reálné plochy trojúhelníků. Výsledky jsou lehce odlišné.



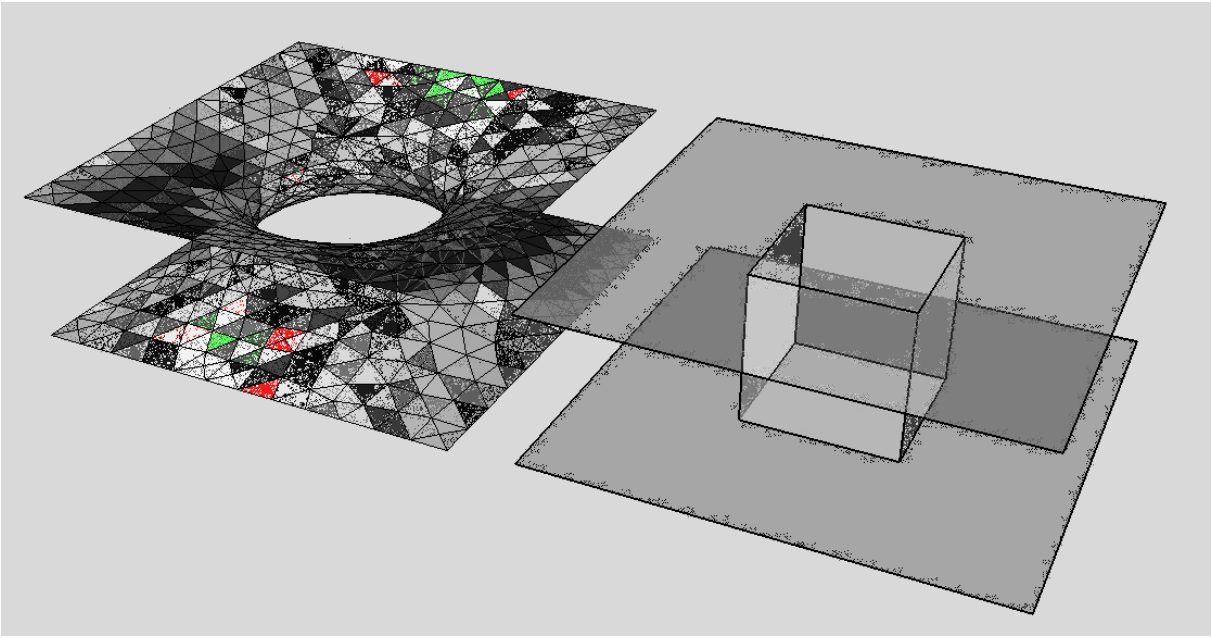
obr. 195 Minimální plocha – dvě varianty

Na obr. 195 jsou znázorněny na nejjednodušších případech dvě řešení minimálních ploch. Na pravém výstupu je výpočet založen na porovnávání reálných délek hran, pracuje s drátovým modelem. Na levém obrázku je algoritmus založen na porovnávání obsahu ploch v blízkosti bodu, pracuje tak s plnohodnotnou plošnou informací.

Rozdíl mezi oběma výstupy spočívá v závislosti na vnitřní struktuře plochy. V pravém případě jsou síly v ploše vedeny pouze po drátech. Struktura drátů je tedy určující. V levém případě síly probíhají přímo plochami, vnitřní členění tedy pro výsledek není určující. Plocha odpovídá principu membrány – mýdlové bublině.

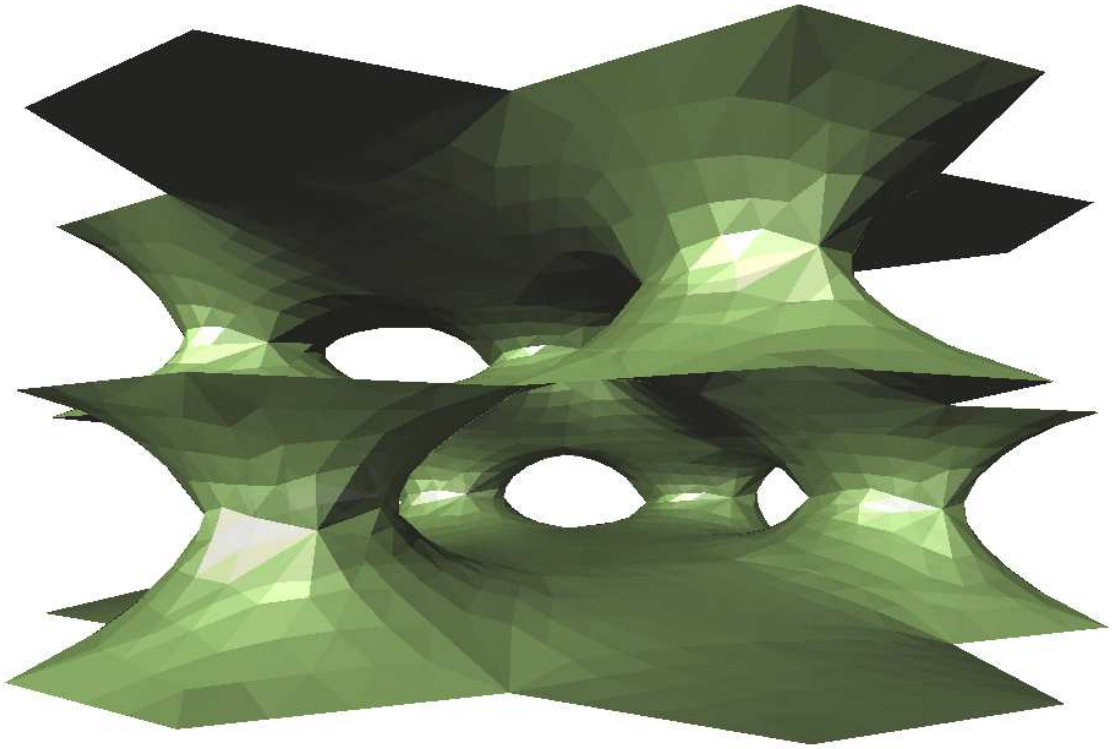


obr. 196 minimální plocha 2

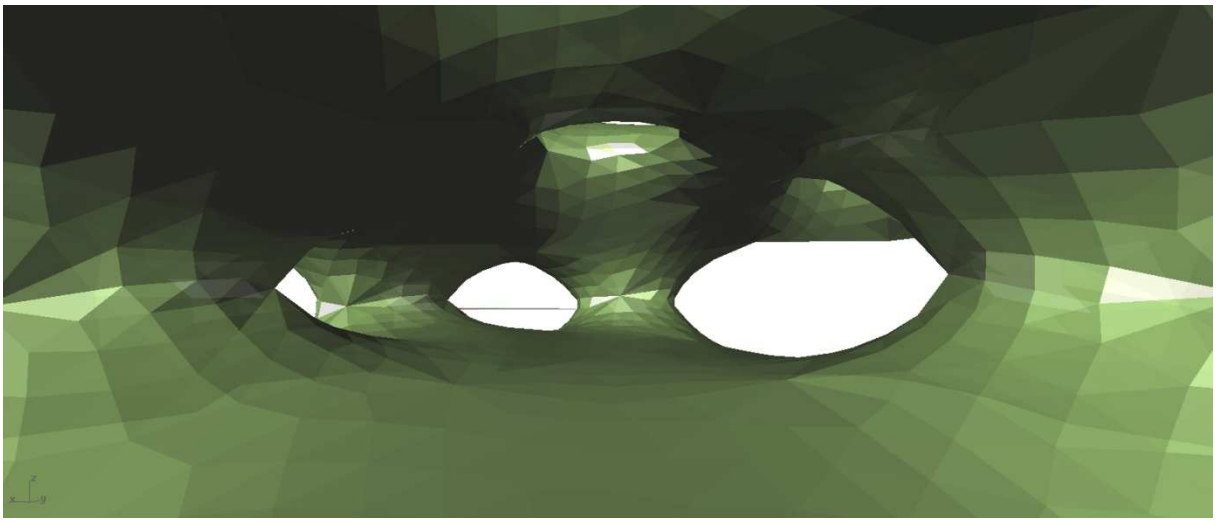


obr. 197 minimální plocha 3, výsledný objekt a vstupní geometrie

Obdobným způsobem lze vytvořit libovolně složitý objekt. Tento objekt (obr. 198, 199) není definován pomocí TPMS rovnic (srovnej projekt v kapitole 3.38), objekt je vytvořen na základě jednoduché vstupní geometrie – přibližný tvar - a pravidla pro chování plochy. V tomto případě je plocha definována rovností tahových sil v každém bodě. Jde tedy o membránu nataženou mezi pevně stanovené hranice objektu.

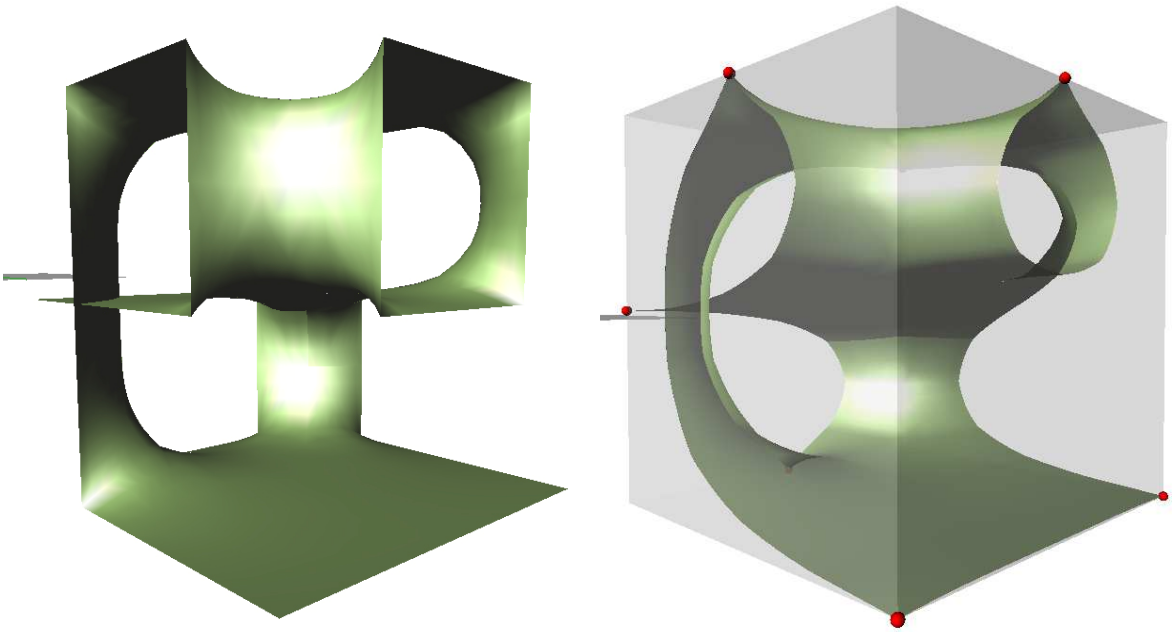


obr. 198 kontinuální plocha



obr. 199 kontinuální plocha – vnitřní prostor

V dalším případě je představena možnost definovat pro jednotlivé části objektu různá pravidla. Na obr. 200 je vygenerovaná mambrána s pevně danými hranicemi. Na obr. 201 jsou hranice uvolněny, jsou pro ně stanovena jiná pravidla (hraniční body musí zůstat v rovinách šedého kubusu). Některé jednotlivé body byly fixovány zcela (červeně). Informace o těchto vlastnostech (restrikcích) jsou uchovávány v seznamech u každého vrcholu (MV_Rest_Set_List).



obr. 200, 201 – restrikce pro skupiny vrcholů

Na tomto objektu lze předvést další vlastnost programu. Je to jeho nezávislost na normálovém směru ploch. Grafické programy, závislé na normálách ploch, nejsou schopny vytvořit kontinuální Möbiovu plochu, která má nejednoznačnou definici normálového směru. Prezentovaný objekt je případem Möbiovy plochy, plocha nemá dvě strany.

5.4.2 Řetězovka

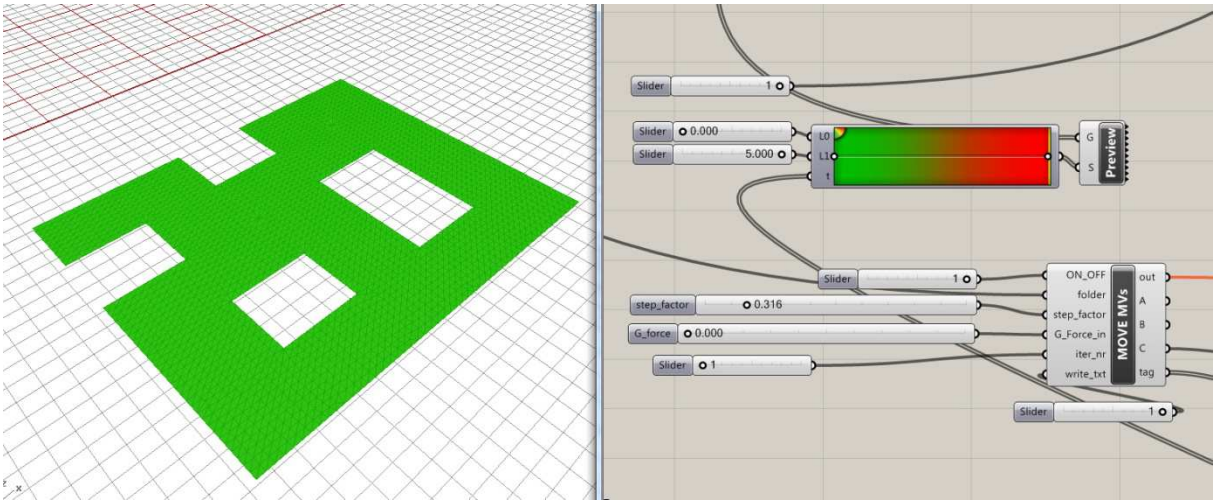
Další z již napsaných algoritmů pracuje na podobném systému, jak definoval svá díla Gaudí. Algoritmus je ale napsán trochu složitěji. Na rozdíl od Gaudího, který vyvěsil drátový model a sledoval průběhy tahových sil po jednotlivých prutech, já se pokusil definovat geometrii na základě průběhu sil v ploše.

Přístup á la Gaudí byl prezentován v projektu Voussoir Cloud od kanceláře IwamotoScott Architecture (viz kapitola 3.33). V projektu je viditelný princip statické posloupnosti základ – žebro – pruty – klenba.

obr. 146 IwamotoScott Architecture a Buro Happold, Voussoir Cloud, LA

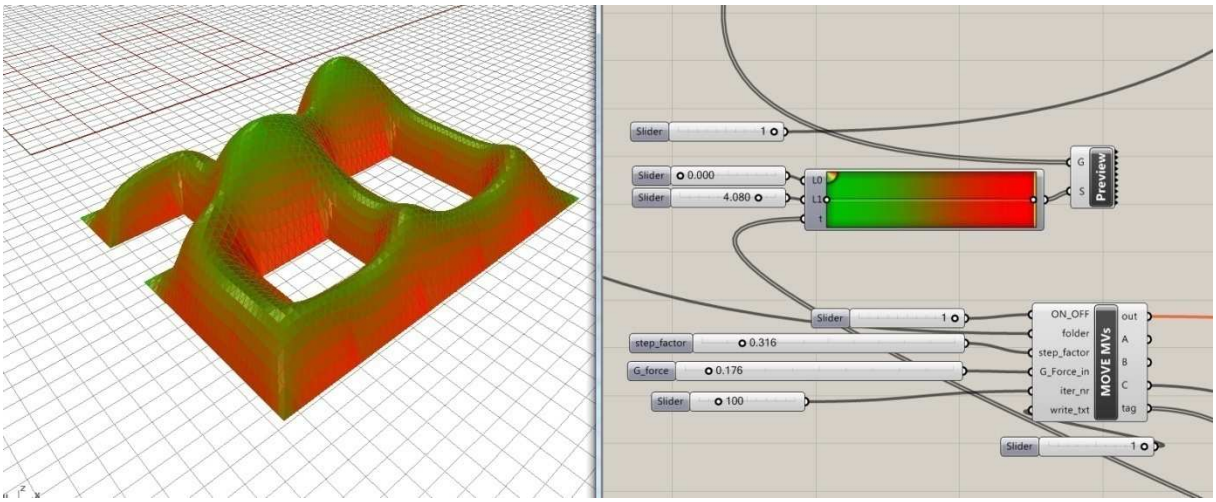
Algoritmus, který navrhuji, je zcela nezávislý na vnitřní struktuře kleneb. Geometrie vychází z principů minimálních tažených ploch a fyzikálního modelování dle chování pružných řetězovek.

Libovolná plocha v počátečním stavu je definována jako bez napětí. Postupná deformace plochy vlastní vahou má za následek jednak vytvarování do tvaru řetězovky obecné, poté se začne plocha napínat a deformovat do tvaru řetězovky pružné, při nadimenzování segmentů plochy dle vyvolaného napětí se plocha deformuje do řetězovky pružné, tížné. Více o řetězovkách se dozvíme v: */Přehled užité matematiky, Karel Rektorys a spol., Česká matice technická, STNL, Praha 1963, str 157-161/.*



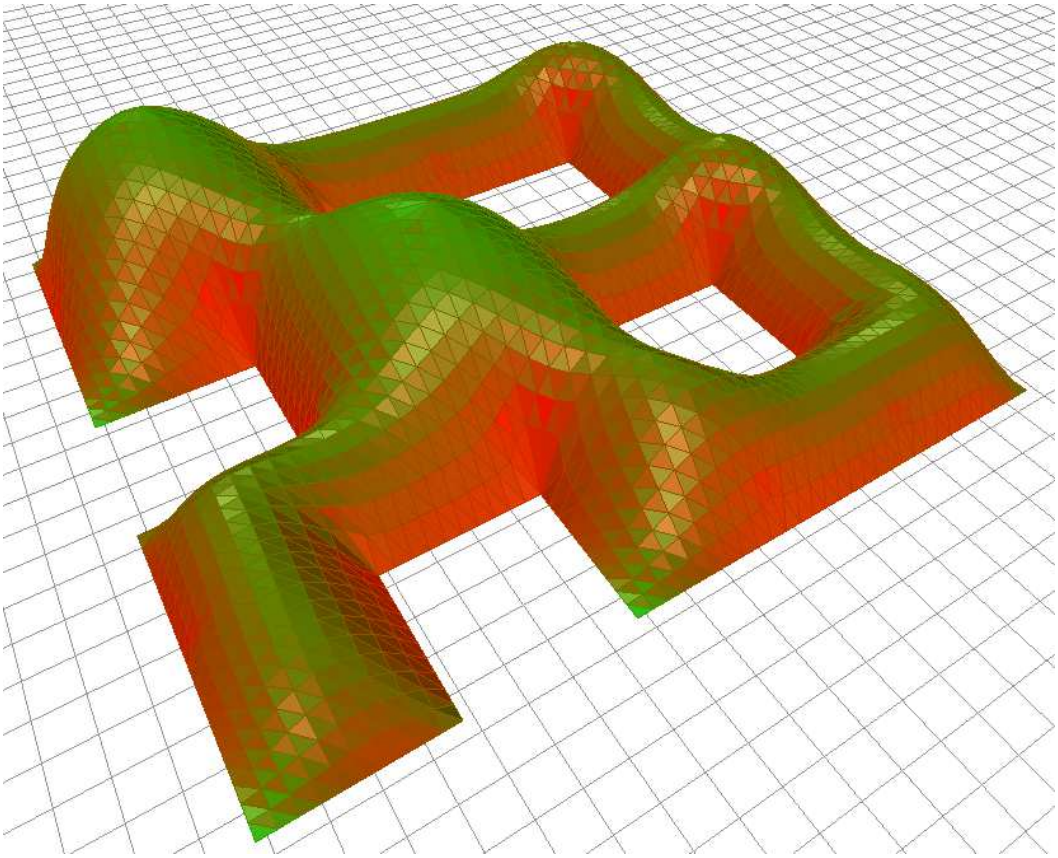
obr. 202 Řetězovka – stav 0

Na obr. 202 vidíme libovolnou zvolenou plochu v počátečním stavu, plocha je rovinná, k deformaci bez protažení nedochází.



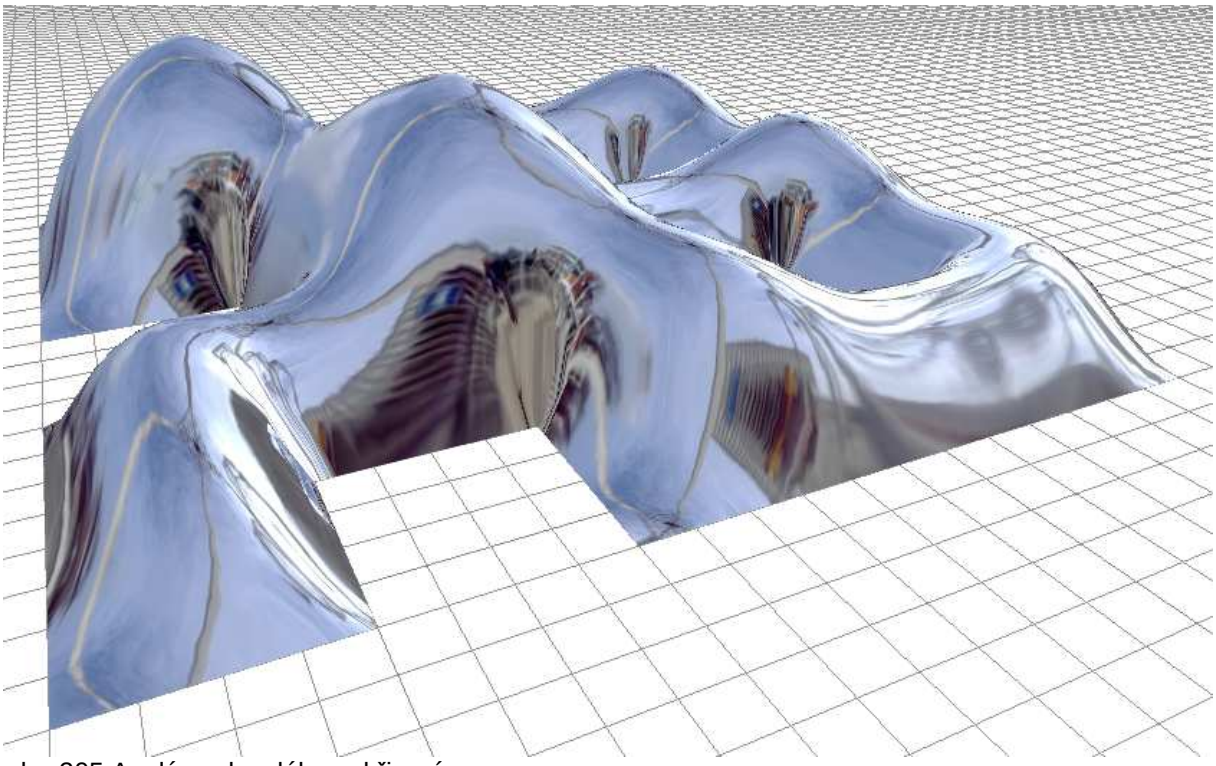
obr. 203 Řetězovka – stav 100

Na obr. 203 je stav po 100 krocích iterace. Plocha se vytvarovala dle principů pružné řetězovky. Barevně je znázorněna analýza deformace – protažení a zatížení plochy. Modul pružnosti je modifikovatelný, v tomto případě je definován ε= ΔS, tedy lineárně s nárůstem plochy.



obr. 204 Řetězovka

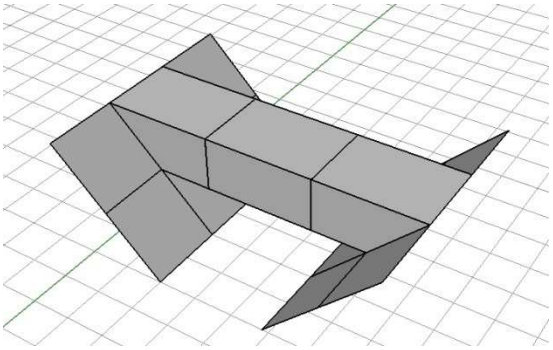
Na vzniklé ploše není znát vliv vnitřní struktury, plocha není definována pomocí žeber. Kdybychom hledali podobný případ v přírodě, mohli bychom si představit mýdlovou bublinu – membránu – napjatou do rámu a poté prohnutou vlastní vahou nebo vyboulenou větrem



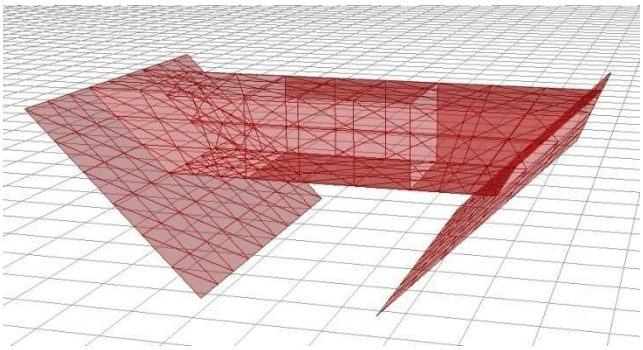
obr. 205 Analýza plynulého zakřivení

Po přisouzení reflexivního materiálu můžeme sledovat plynulost výsledné geometrie bez vnitřních hran.

Tak jako celý program ani tento algoritmus není omezen pouze na zadání plošných objektů. Jako příklad předvedu jeden prostorový oblouk.

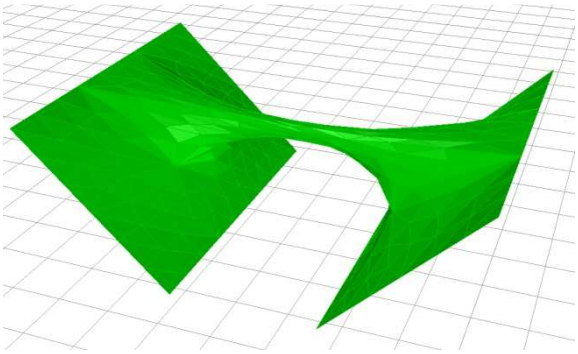


obr. 206 prostorové zadání

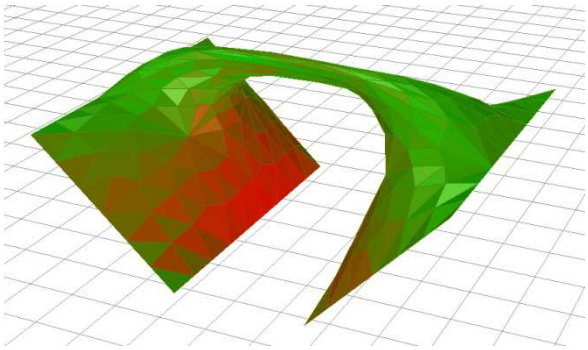


obr. 207 vygenerevaná síť

Na obr 206 vidíme zadáni pomocí 22 rovinných ploch, na obr. 207 je vygenerováná síť se všemi vnitřními vazbami.

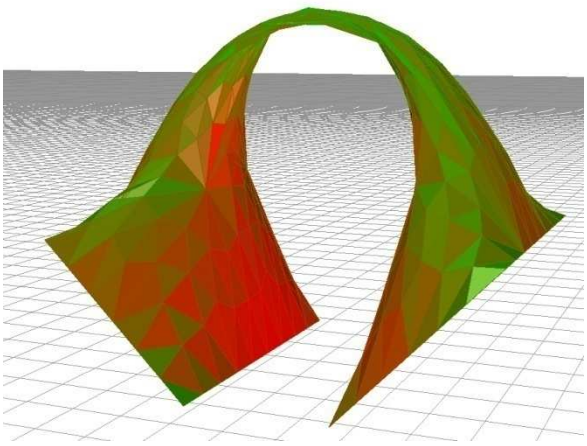


obr. 208 minimální plocha

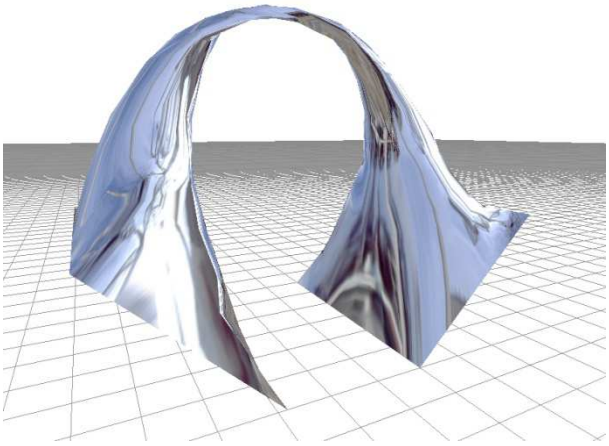


obr. 209 tížná řetězovka

Na obr. 208 je primární deformace pomocí algoritmu pro tvorbu minimálních ploch. Používám jej k definování prvotního kroku pro další algoritmy. Tahové síly v objektu jsou v tuto chvíli vyrovnané. Na obr. 209 je objekt deformován proti zatížení od vlastní váhy. Barevně vyznačeno namáhání v tlaku, vzhledem k definici řetězovek není v ploše namáhání ohybem – momenty. Tvar odpovídá obrácené tížné řetězovce aplikované do plochy.



obr. 210 tížná, pružná řetězovka



obr. 211 výstup

Na obr. 210 byla umožněna další deformace – protažení plochy, tvar objektu nyní odpovídá pružné, tížné řetězovce. Na obr. 211 vyhlazená výstupní geometrie.

5.4.3 Další Algoritmy

Nyní jsou ve vývoji algoritmy, které modifikují geometrii plochy na základě informací o křivosti. Analýza křivosti v ploše je definována pomocí vzájemných úhlů sousedících segmetů – trojúhelníků. Následné modifikace – posuny jednotlivých vrcholů jsou definovány například tak, aby přílišné zakřivení eliminovaly. Tímto způsobem lze stanovit minimální poloměry v ploše.

Dalšími algoritmy v pořadí pro zpracování jsou: šíření vlastností po ploše (pro práci s dispozicemi), metaball a prolínání plochy (řešení průsečíků, křížení plochy, interakce mezi vzdálenými body v ploše), proměnná tloušťka desky (přechod do trojrozměrné struktury) atd.

6 Závěr

Shrnutí získaných poznatků:

V dizertačí práci jsem popsal důležité vědomosti a zdroje, které považuji za potřebné pro vývoj a užívání digitálních metod v procesu architektonického navrhování.

Návrh aplikace poznatků na vzdělávání budoucích architektů:

Poznatky bych doporučoval zavést do osnov teoretické výuky na FaVUT, zejména geometrické principy a matematická vyjádření křivek vyššího řádu, a hlavní principy programování.

Podobně považuji za vhodné při výuce alespoň zmínit, lépe i vysvětlit podstatu, mnoha současných projektů vytvořených digitálními metodami.

Tyto změny navrhuji pro umožnění studentům pochopit a poté i napodobit jeden z výrazných současných směrů v architektuře, který se v českém prostředí bohužel téměř nevyskytuje.

Praktický výstup:

Výstup praktické části – navržený software – je snahou o přiblížení se úrovni digitální architektury dosažené v zahraničí. Cílem je jeho aplikace v praxi, realizace projektu nově navrženými metodami.

Výhled:

Digitální metody v architektonickém navrhování se v posledních deseti letech prudce rozvíjejí. Jejich aplikace zatím můžeme sledovat jen u několika významných architektonických kanceláří. V dalších letech očekávám masové rozšíření digitálních metod a jejich přirozené zapojení do procesu architektonického navrhování. Očekávám a doufám i v nějaké realizace v českém provedení.

Publikace

Sirotek, A.; Müller, R. *Paralela.* In: Sborník XII. vědecké konference doktorandů 2008. Brno: Fakulta architektury VUT, 2008. ISBN 978-80-214-3656-5

Sirotek, A. *Komplexní architektura, komplexní geometrie*. 2009. /www.rhino3d.cz/

Prezentace

Šimonová, B.; Müller, R.; Sirotek, A. *Současná architektura v ČR*, Valencia, 2006, (přednáška na Universitat politechnica de Valencia)

Sirotek, A.; Müller, R.; Soviš, P. *Paralela*. Konference Paracloud, Tel Aviv, 2007 (prezentace výzkumu na Shenkar College of Engeneering and Design)

Seznam vyobrazení

U převzatých ilustrací je uveden zdroj, autorské ilustrace jsou ponechány bez komentáře.

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str.14-16, autor neuveden/

obr. 90 Greg Lynn, Embryological House

/zdroj: časopis AD 76-2, 03-04/2006, strana 92/

obr. 91 Greg Lynn FORM, DADAGROWS, Florencie

/zdroj: časopis AD 77-1, 01-02/2007, strana 38/

obr. 92,93 Biennale Venezia 2004

/foto: Adam Sirotek/

obr. 94 Lars Spuybroek - NOX, oblique WTC, New York

/zdroj: NOX Machining Architecture, Lars Spuybroek, Thames & Hudson, strana 264/

obr. 95 oblique WTC - vývoj

/zdroj: NOX Machining Architecture, Lars Spuybroek, Thames & Hudson, strana 261/

obr. 96 Amazing Whale Jaw, Maurice Nio, Hoofsddorp

/zdroj: Architekt 2007 – 07, str. 41, autor fotografií neuveden/

obr. 97 GregLynn, Alessi Tea & Coffee Towers

/zdroj: časopis AD 76-4, 07-08/2006, strana 88/

obr. 98 Ron Arad Associates, Holon Design Museum, Israel

/zdroj: časopis AD 76-6, 11-12/2006, strana 50/

obr. 99 DR_D_Lab (Art Academy Stuttgart) 2002-2003, DR_D Studio (Berlin)

/zdroj: časopis AD 76-6, 11-12/2006, strana 66/

obr. 100,101 Bienale Venezia 2004

/foto: Adam Sirotek/

obr. 102 iterace zaoblení

obr. 103-106 Lars Spuybroek – NOX a Q. S. Serafijn, D-Tower, Doetinchem

/foto: Adam Sirotek/

obr. 107-108 Lars Spuybroek – NOX a Edwin van der Heide, Son-O-House, Son en Breugel

/zdroj: NOX Machining Architecture, Lars Spuybroek, Thames & Hudson, strana 193/

obr. 109-110 Son-O-House – modely

/Bienale Venezia 2004, foto: Adam Sirotek/

obr. 111 Maison Folie – model

/Bienale Venezia 2004, foto: Adam Sirotek/

obr. 112 Lars Spuybroek - NOX, Maison Folie, Lille

/zdroj: NOX Machining Architecture, Lars Spuybroek, Thames & Hudson, strana 243/

obr. 113 Lars Spuybroek - NOX, Maison Folie, Lille

/zdroj: časopis AD 76-6, 11-12/2006, strana 26/

obr. 114-115 Andrew Kudless, Manifold Honeycomb Morfologies, London

/zdroj: časopis AD 76-2, 03-04/2006, strana 84/

obr. 116 Brandon Williams / Studio Rocker, 3D Game of Life

/ zdroj: časopis AD 76-4, 07-08/2006, strana 24/

obr. 117 Stanley Perelman, Project X, Mannhatan

/zdroj: časopis AD 76-2, 03-04/2006, strana 56/

obr. 118 Jean Nouvel, Torre Agbar, Barcelona

/foto: Adam Sirotek/

obr. 119 Torre Agbar – schéma

/zdroj: časopis Architekt 2006 – 06, str. 14/

obr. 120,121 Satniago Calatrava, Turning Torso, Malmö

/zdroj: časopis Architekt 2006 – 08, str. 6/

obr. 122,123 Massimiliano Fuksas, New Milan Fair Trade, Milano

/zdroj: časopis Architekt 2008 – 03, str. 69/

obr. 124-126 Massimiliano Fuksas, New Milan Fair Trade, Milano

/zdroj: www.fuksas.it/

obr. 127 Shigeru Ban, Arup AGU, Pompidou centre, Metz

/zdroj: časopis AD 76-6, 11-12/2006, strana 87

obr. 128 Pompidou centre - model

/Bienale Venezia 2004, foto: Adam Sirotek/

obr. 129 Zaha Hadit Architects, Phaeno Science Centre, Wolfsburg

/zdroj: časopis AD 77-1, 01-02/2007, strana 32-33

obr. 130-132 Contemporary Architecture Practise – Ali Rahim, Hina Jamelle, Dubai

/zdroj: časopis AD 77-1, 01-02/2007, strana 67-75

obr. 133-134 Alsop Architects , Toronto

/zdroj: časopis AD 76-6, 11-12/2006, strana 38/

obr. 135-136 OCEAN a Scheffer + Partner, Praha

/zdroj: časopis AD 78-2, 03-04/2008, strana 105,107

obr. 137 PTW Architects, CSCEC, CCDI, Arup, Peking

/zdroj: http://www.theage.com.au/photogallery/2008/03/11/1205125883538.html/

obr. 138-140 Weaire–Phelanova struktura

/zdroj: http://en.wikipedia.org/wiki/Weaire%E2%80%93Phelan_structure/

obr. 141 Herzog & de Meuron, ArupSport, Bird Nest, Peking

/zdroj:http://innovativebuildings.net/2010/06/18/beijing-national-stadium-birds-nest-china/ /

obr. 142 Arup AGU, 2005

/zdroj: časopis AD 76-6, 11-12/2006, strana 84

obr. 143 Coop Himmelb(l)au, BMW Welt, Mnichov

/zdroj: časopis AD 78-2, 03-04/2008, strana 20,22

obr. 152 Supermanoeuvre, Trabeculae - půdorys

/ zdroj: http://supermanoeuvre.com/trabeculaetower/ /

obr. 153 Proto-Synthesis & Trabeculae – vývojový diagram

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str. 221 /

obr. 154 Trabeculae – model

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str. 219 /

obr. 155 Trabeculae - metaball

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str. 218 /

obr. 156 Tomáš Legner, Velké pražské Benátky, Praha

/zdroj: časopis Architekt 2008 – 10, str. 64/

obr. 157 Velké pražské Benátky - schéma

/zdroj: časopis Architekt 2008 – 10, str. 64/

obr. 158-160 Tang & Yang Architects, Geno-Matrix

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str. 194-199 /

obr. 158-160 MESNE, Strange Atractor, South Eastern Victoria

/zdroj: Contemporary Digital Architecture: Design and Techniques, Jacobo Krauel, Links, str. 200-203 /

obr. 161-163 Michal Bednář / FLO(W), Liqud Blocks

/zdroj: časopis Architekt 2010 – 01, str. 72-73/

obr. 164 Toyo Ito, Andrea Branzi, Forum for Music, Dance and Visual Culture

/zdroj: časopis AD 76-6, 11-12/2006, strana 17/

obr. 165 SPARCH a Alsop Design, Gallery Building, Shanghai

/zdroj: časopis Architekt 2012 – 01, str. 0/

obr. 166-167 Luka křížek, IO Studio, Vodafone, Praha

/zdroj: časopis Architekt 2012 – 01, str. 68-73/

obr. 168-170 Magma Architecture, Olympic Shooting Venue, Londýn

/zdroj: www.dezeen.com/2012/06/12/olympic-shooting-venue-by-magma-architecture/, foto: J.L. Diehl /

obr. 171 Schéma systému – obdoba procesu architektonického navrhování člověkem

obr. 172 Schéma subsystému – Zdroj

obr. 173 Schéma subsystému – Překlad

Použité zdroje

01 REKTORYS, Karel a spol. Přehled užité matematiky. Praha: Státní nakladatelství technické literatury, 1963

02 GÖSSEL, Peter a LEUTHÄUSER, Gabriele. *Architecture in the Twentieth Century*. Benedikt Taschen. ISBN 3-8228-0550-5

03 ZERBST, Rainer. *Antoni Gaudí i Cornet – život v architektuře*. Benedikt Taschen a Vydavatelstvo Slovart, ISBN 3-8228-9699-3

04 JODIDIO, Philip. *Santiago Caltrava*. Tachen, ISBN 3-8228-5785-8

05 KRAUEL, Jacobo. *Contemporary Digital Architecture: Design and Techniques*. Barcelona: Links. ISBN 978-84-92796-59-5

06 RASHID, Hani a COUTURE, Lise Anne. *Asymptote Flux*. Phaidon Press Limited, 2002, ISBN 0-7148-4172-2

07 FOGDEN, A a HYDE, S.T., *Continuous transformations of cubic minimal surfaces*, The European Physical Journal B 7/1999, str 91-104.

08 SCHRÖDER, G.E., RAMSDEN, S.J., CHRISTY, A.G., HYDE, S.T., *Medial surfaces of hyperbolic structures*. The European Physical Journal B 35/2003, str 551-564.

09 HIGHT, CH., PERRY, CH., *Collective Inteligence in Design*, AD Architectural Design, September/2006, ISBN 978-0-470-02652-6

10 ENDIE-BROWN, P., ANDRASEK, A., *Continuum: A Self-Engeneering Creature-Culture*, AD Architectural Design, September/2006, ISBN 978-0-470-02652-6

11 McCULLOUGH, M., *20 years of scripted space*, AD Architectural Design, July/2006, ISBN 978-0-470-02585-7

12 CHU, K., *Metaphysics of genetic architecture and computation*, AD Architectural Design, July/2006, ISBN 978-0-470-02585-7

13 SILVER, M., *Building without drawings: Automason Ver 1.0*, AD Architectural Design, July/2006, ISBN 978-0-470-02585-7

14 HENSEL, M., MENGES, A., WEINSTOCK, M., *Techniques and Technologies in Morphogenetic Design*, AD Architectural Design, March/2006, ISBN 978-0-470-01529-2

15 RAHIM, A., JAMMELLE, H., *Elegance in the Age of Digital Technique*, AD Architectural Design, January/2007, ISBN 978-0-470-02968-8

16 SCHUMACHER, P., *Arguing for Elegance*, AD Architectural Design, January/2007, ISBN 978-0-470-02968-8

17 RAHIM, A., JAMMELLE, H., *Surface Continuity: An Elegant Intefration*, AD Architectural Design, January/2007, ISBN 978-0-470-02968-8

18 PIKE, S., *Manipulation and Control of Micro-Organic-Matter in Architecture*, AD Architectural Design, November/2007, ISBN 978-0-470-51958-5

19 HENSEL, M., MENGES, M., PEDRESCHI, R., *Versatility and Vicissitude: Performance in Morpho-Ecological Design*, AD Architectural Design, March/2008, ISBN 978-0-470-51687-4

20 SCHUMACHER, P., *Parametricism: A New Global Style for Architecture and Urban Design*, AD Architectural Design, July/2009, ISBN 978-0-470-77300-0

21 FRANKEN, B., *Výstavní pavilon "Bublina" ve Frankfurtu nad Mohanem,* ERA21, 4/05, ISBN 9-771801-089006-09

22 OOSTERHUIS, K., *Protihluková bariéra & kokpit*, ERA21, 4/05, ISBN 9-771801-089006-09

23 NIO, M., *Zastávka ve tvaru velrybí čelisti v Hoofddoprpu*, ERA21, 4/05, ISBN 9-771801-089006-09

24 LYNN, G., *Calculus-Based Form*, přednáška
URL: <http://www.ted.com/talks/greg_lynn_on_organic_design.html>

25 NIO Architekten, *autorská zpráva k projektu*, URL: < http://www.nio.nl>

26 SHNEIDER, P., *B-Splines*,
URL: <https://www.cs.drexel.edu/~david/Classes/CS430/Lectures/L-09_BSplines_NURBS.pdf>

27 *Rhinoceros 5.0 Beta*, (počítačový program), Robert McNeel

28 *Grasshopper*, (počítačový program), Robert McNeel