

1

Algoritmy a počítačová grafika

2

Poznámky k přednáškám

3

PRACOVNÍ VERZE

4

Pozor, musí být spuštěn na pozadí WEB prohlížeč

5

testováno s MS IE 8

6

DRAFT – 2.16-01

7

Rád bych požádal o zaslání nalezených chyb, nekorektních či nejasných formulací apod.,
které bych rád v další verzi textu opravil.

8

(ideálně skala@kiv.zcu.cz předmět zprávy: APG&ZPG / verze;
v textu zprávy uveďte stránku a řádku na stránce)

9

10

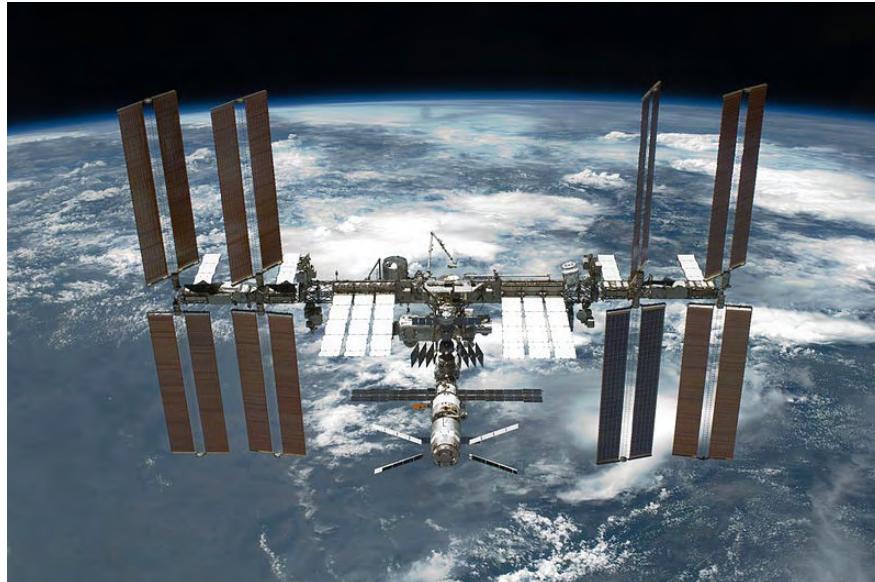
Text je doplněn přednáškami a videozáznamy z let 2007/8 - 2012/13

11

12

13

14



15

16

17

18

Nešířit

19

20

prof.Ing.Václav Skala, CSc.

c/o Katedra informatiky a výpočetní techniky

Fakulta aplikovaných věd

Západočeská univerzita v Plzni

skala@kiv.zcu.cz http://www.VaclavSkala.eu

© Václav Skala 2014

Created: 2010-04-10

21

1	Obsah	
2	Profil přednášejícího.....	9
3	Základy počítačové grafiky (ZPG) - obsah přednášek.....	11
4	Algoritmy a počítačová grafika (APG) - obsah přednášek.....	12
5	1. Úvod	13
6	1.1. Přehled vývoje počítačové grafiky.....	13
7	1.2. Počítačová grafika a ostatní oblasti.....	21
8	1.3. Architektura grafických systémů.....	22
9	1.4. Grafické knihovny	26
10	1.4.1. OpenGL.....	26
11	1.4.2. DirectX	26
12	1.4.3. Příklad s OpenGL.....	27
13	2. Algoritmy a složitost algoritmů	29
14	2.1. Typy algoritmů.....	29
15	2.2. Posuzování algoritmů	30
16	2.2.1. Složitost algoritmů.....	30
17	2.2.2. Robustnost algoritmu	32
18	2.3. Experimentální vyhodnocení výpočetní náročnosti algoritmu	33
19	2.4. Experimentální porovnávání algoritmů.....	34
20	3. Matematický aparát počítačové grafiky	36
21	3.1. Souřadné systémy v počítačové grafice	37
22	3.1.1. Souřadné systémy v E^2	37
23	3.1.2. Souřadné systémy v E^3	38
24	3.1.3. Datové typy	39
25	3.2. Homogenní souřadnice a jejich geometrická interpretace	40
26	3.2.1. Operace v Eukleidovském prostoru.....	40
27	3.3. Dualita a její aplikace.....	42
28	3.3.1. Princip duality	42
29	3.3.2. Vektorový součin a řešení soustav lineárních rovnic	43
30	3.3.3. Aplikace duality v prostoru E^3	46
31	3.3.4. Vzdálenost v projektivním prostoru	47
32	3.4. Kvaterniony	48
33	3.4.1. Komplexní čísla a jejich reprezentace	48
34	3.4.2. Kvaterniony	49
35	3.4.3. Základní operace s kvaterniony	49
36	3.4.4. Kvaterniony a rotace	50
37	3.5. Plückerovy souřadnice.....	52
38	3.6. Numerická reprezentace a stabilita výpočtů.....	54

1	3.6.1.	Vybrané příklady a ukázky numerických problémů	55
2	3.6.2.	Rekurze	58
3	3.6.3.	Další možnosti zvýšení přesnosti	58
4	3.6.4.	Příklad překvapivé nestability výpočtu.....	59
5	3.7.	Příklady katastrof způsobených výpočetními chybami	60
6	4.	Základní geometrické transformace.....	63
7	4.1.	Základní transformace v E^2	63
8	4.1.1.	Posuv (translation)	63
9	4.1.2.	Rotace (rotation)	64
10	4.1.3.	Změna měřítka (scaling)	65
11	4.1.4.	Zrcadlení (mirroring).....	66
12	4.1.5.	Zkosení (shearing)	68
13	4.2.	Řetězení transformací	69
14	4.3.	Window - Viewport transformace.....	72
15	4.4.	Normalizovaný souřadný systém	73
16	4.5.	Základní transformace v E^3	74
17	4.6.	Rotace okolo dané osy v E^3	77
18	4.7.	Transformace rotace v E^3 okolo osy založená na rotaci vektorů.....	78
19	4.8.	Transformace přímek a rovin	79
20	4.9.	Generace rotačního tělesa	80
21	4.10.	Generace povrchu koule	81
22	4.10.1.	Generování pomocí sférických souřadnic	81
23	4.10.2.	Povrch generovaný iso-metricky	81
24	4.11.	Geometrické transformace - vektorově orientované formulace	83
25	4.11.1.	Operace zrcadlení podle roviny v obecné poloze v E^3	84
26	4.11.2.	Rotace okolo obecné osy v E^3	85
27	5.	Projekce	87
28	5.1.	Perspektivní projekce	89
29	5.2.	Matematický aparát rovinné projekce	90
30	5.3.	Paralelní projekce	96
31	5.4.	Perspektiva a pseudovzdálenost	98
32	5.5.	Stereoskopická projekce	102
33	5.6.	Nerovinné projekce	103
34	5.7.	Projekce a faktor dynamiky	106
35	6.	Metody ořezávání v E^2 a E^3 a operace s n-úhelníky.....	107
36	6.1.	Algoritmy ořezávání v E^2	107
37	6.1.1.	Cohen-Sutherland algoritmus	107
38	6.1.2.	Sutherland Hodgman algoritmus	110
39	6.1.3.	Cyrus Beck algoritmus	110
40	6.1.4.	Algoritmus se složitostí $O(\lg n)$	111

1	6.1.5.	Algoritmus Smart-Clip (S-Clip)	112
2	6.2.	Algoritmy ořezávání v E^3	117
3	6.2.1.	Cohen-Sutherland algoritmus	117
4	6.2.2.	Cyrus-Beck algoritmus	117
5	6.2.3.	Sutherland-Hodgman algoritmus	117
6	6.2.4.	Algoritmus se složitostí $O_{\text{expected}}(\text{VN})$	119
7	6.3.	Operace s n-úhelníky v E^2	121
8	6.3.1.	Weiler-Athertonův Algoritmus.....	121
9	6.3.2.	Množinové operace.....	122
10	7.	Datové struktury.....	123
11	7.1.	Základní typy popisu dat	123
12	7.2.	Základní geometrická primitiva	125
13	7.3.	Reprezentace třírozměrných objektů a scén.....	126
14	7.4.	Dělení prostoru a Binární masky	127
15	7.4.1.	Dělení prostoru.....	127
16	7.4.2.	Rezidenční maska	127
17	7.4.3.	Binární masky	128
18	7.4.4.	Quadtree a Octree.....	129
19	7.5.	Hraniční reprezentace – dodělat Half-Edge	130
20	7.6.	Eulerovy operátory a manifoldy	136
21	7.7.	CSG stromy	137
22	7.8.	STL formát	141
23	7.9.	Algoritmy výpočtu průsečíků a ohraničující tělesa.....	142
24	8.	Polygonální síť – dodělat	144
25	8.1.	Progressive mesh refinement - dopsat	144
26	8.2.	Metody redukce trojúhelníkových sítí - dopsat	144
27	8.3.	Diskrétní charakteristiky polygonálních povrchů (křivosti apod.) - dopsat.....	144
28	8.4.	Hashing a eliminace duplicit.....	145
29	8.5.	Rekonstrukce trojúhelníkové sítě z množiny trojúhelníků	147
30	9.	Reprezentace scény.....	148
31	9.1.	Graf scény (Scene Graph)	148
32	9.2.	Reprezentace grafu scény	149
33	10.	Interpolace a approximace uspořádaných a neuspořádaných dat	150
34	10.1.	Lineární interpolace	151
35	10.2.	Sférická interpolace	156
36	10.3.	RBF interpolace	157
37	10.4.	Aproximace	162
38	10.5.	Metoda nejmenších čtverců	163
39	10.6.	Výpočet vzdálenosti bodu od trojúhelníka v E^3	165
40	10.7.	Metody výpočtu průsečíků	166

1	10.7.1.	Výpočet průsečíku přímky a trojúhelníka v E^3	166
2	10.7.2.	Určení přímky jako průsečnice dvou rovin.....	167
3	10.7.3.	Určení nejbližšího bodu na průsečnici dvou rovin.....	168
4	10.7.4.	Výpočet průsečíku přímky a koule v E^3	169
5	10.7.5.	Výpočet obalové koule opsané trojúhelníku v E^3	170
6	10.7.6.	Výpočet průsečíku přímky a kvadratické plochy v E^3	171
7	10.7.7.	Detekce existence průsečíku s kvadratickou plochou.....	171
8	10.7.8.	Detekce existence průsečíku s koulí – zkontrolovat !!!!!!!.....	174
9	10.7.9.	Výpočet průsečíku dvou trojúhelníků v E^3 – čtenář si doplní sám	175
10	10.8.	RBF aproximace	176
11	11.	Parametrické křivky a plochy.....	178
12	11.1.	Parametrické kubické křivky.....	180
13	11.2.	Vykreslování parametrických křivek.....	186
14	11.3.	Vzájemný převod kubických parametrických křivek	187
15	11.4.	Hladké napojování kubických křivek	188
16	11.5.	Parametrické plochy	190
17	11.6.	Bilineární a bikubický Coonsův plát.....	195
18	11.7.	Vzájemný převod bikubických plátů.....	196
19	11.8.	Vykreslování parametrických ploch.....	198
20	11.9.	Hladké napojování.....	200
21	11.10.	Výhody a nevýhody parametrické reprezentace.....	201
22	12.	Algoritmy řešení viditelnosti	202
23	12.1.	Odstranění „zadních“ stěn	204
24	12.2.	Řešení viditelnosti pro plochy $2\frac{1}{2}$ D – $z=f(x,y)$	205
25	12.3.	Hloubkový buffer – z-buffer	207
26	12.4.	Algoritmus Malíře	208
27	12.5.	BSP strom	209
28	13.	Barvy a barevné systémy.....	210
29	13.1.	Achromatické světlo	211
30	13.2.	Půltónování.....	212
31	13.2.1.	Metoda vzorů	212
32	13.2.2.	Metoda rozptylu chyb (Floyd-Steinberg).....	213
33	13.2.3.	Dithering	214
34	13.3.	Chromatické světlo - barvy.....	216
35	13.3.1.	Systém RGB.....	217
36	13.3.2.	Systém CMY a CMYK.....	218
37	13.3.3.	Systém XYZ a CIE-xy	219
38	13.3.4.	Systém YIQ.....	225
39	13.3.5.	Systém Lab – doplnit	225
40	13.3.6.	Systém AC ₁ C ₂ – doplnit	225
41	13.3.7.	Systém Opponent – doplnit	225

1	13.3.8.	Profily ICC – doplnit	225
2	13.3.9.	Systém HSV, HLS a HSI.....	226
3	13.3.10.	Barevné vzorníky	229
4	14.	Modely osvětlení a stínování.....	230
5	14.1.	Modely osvětlení	230
6	14.2.	Metody stínování – lokální metody	234
7	14.2.1.	Konstantní stínování.....	234
8	14.2.2.	Gouraud stínování	235
9	14.2.3.	Phong stínování	237
10	14.2.4.	Torrance-Sparrow stínování a Blinn modifikace	238
11	15.	Globální metody	242
12	15.1.	Metoda sledování paprsku	243
13	15.2.	Základní algoritmus Ray-tracing	247
14	15.3.	Příklad	250
15	15.4.	CSG stromy a sledování paprsku	256
16	16.	Radiační metoda.....	257
17	16.1.	Princip radiační metody.....	257
18	16.2.	Výpočet konfiguračních faktorů	260
19	17.	Mapovací techniky – texturování – vložit text	265
20	17.1.	Texturovací techniky	265
21	17.2.	Techniky zvrásnění	266
22	17.3.	Mapování okolí	266
23	17.4.	Procedurální texturování	266
24	18.	Animace, principy a inverzní kinematika – vložit text	267
25	18.1.	Kinematika, Inverzní kinematika – vložit text	267
26	18.2.	Dynamika – vložit text	267
27	18.3.	Particles – vložit text	267
28	18.4.	Ryv a jeho důsledky pro animaci a taktelní I/O – vložit text	267
29	19.	Rastrová grafika – základní algoritmy	268
30	19.1.	Bresenhamův algoritmus – úsečka.....	268
31	19.2.	Bresenhamův algoritmus a algoritmus plovoucího horizontu	271
32	19.3.	Castle-Pitteway algoritmus generování úsečky.....	273
33	19.4.	Bresenhamův algoritmus pro kružnice.....	274
34	19.5.	Michnerův algoritmus	276
35	19.6.	Řádková konverze.....	277
36	19.7.	Algoritmy plnění a šrafování	278
37	19.7.1.	Semínkové plnění	278
38	19.7.2.	Řádkové semínkové plnění	279
39	20.	Vizualizace informací a dat	281

1	20.1.	Vizualizace informací.....	281
2	20.2.	Vizualizace dat	286
3	20.2.1.	Modelování a simulace reality.....	286
4	20.2.2.	Typy dat	287
5	20.2.3.	Datové struktury.....	289
6	20.2.4.	Objemová (volumetrická) data.....	291
7	20.2.5.	Vizualizace dynamických dat	292
8	21.	Geometrická algebra - základy	294
9		Doporučená literatura	295
10	22.	Příloha A – Vektory a geometrická interpretace	297
11	22.1.	Základní operace s vektory	298
12	22.2.	Vzdálenost bodu od přímky v E^2 a v E^3	301
13	22.3.	Další operátory	302
14	23.	Příloha B – Matice a operace s maticemi	303
15	23.1.	Základní operace s maticemi	303
16	23.2.	Determinanty	304
17	24.	Příloha C – Derivace vektorových a maticových funkcí	305
18	24.1.	Základní pravidla pro derivaci s vektory a maticemi, resp. vektorových a maticových funkcí.	305
20	24.2.	Základní identity	307
21	25.	Příloha D – Diferenciální a integrální operátory	308
22	25.1.	Diferenciální operátory	308
23	25.2.	Integrální operátory	310
24	26.	Příloha E – Řecká abeceda.....	311
25			
26			

1 Profil přednášejícího

2 (<http://www.VaclavSkala.eu>)

3
4 Prof. Ing. Václav Skala, CSc. se zabývá počítačovou grafikou od r. 1975, kdy na
5 Vysoké škole strojní a elektrotechnické v Plzni zavedl a následně vyučoval
6 předmět Počítačová grafika a umělá inteligence. V roce 1981 obhájil
7 disertační práci na téma relačních databází se specializací na reprezentaci
8 relací a optimalizaci dotazů uživatele. V současné době se odborně věnuje
9 především algoritmům počítačové grafiky, vizualizaci dat a algoritmům
10 obecně, včetně matematických aspektů. Je vedoucím Centra počítačové
11 grafiky a vizualizací (<http://Graphics.zcu.cz>) na Katedře informatiky a
12 výpočetní techniky na Fakultě aplikovaných věd Západočeské univerzity v Plzni. V letech 1984-1989
13 působil na Brunel University v Londýně a později přednášel na NATO Advanced Study Institute
14 v zahraničí. V roce 1996 se stal profesorem na Západočeské univerzitě, odborně působil na Bath
15 University v U.K., na Gavle University ve Švédsku a na dalších zahraničních odborných pracovištích.

16 Prof. Skala je řešitelem zahraničních i národních odborných výzkumných projektů, členem
17 několika redakčních rad prestižních impaktovaných zahraničních odborných časopisů, členem
18 programových výborů mezinárodních odborných konferencí. Od r. 1992 je organizátorem
19 mezinárodních odborných konferencí WSCG zaměřených na počítačovou grafiku, vizualizaci dat a
20 počítačové vidění (<http://www.WSCG.eu>), dříve též organizoval konference .NET Technologies
21 (<http://dotnet.zcu.cz>) a GraVisMa – Graphics, Vision and Mathematics (<http://www.GraVisMa.eu>).

22 Prof. Skala je profesorem na Západočeské univerzitě v Plzni. Odborně působil též na
23 VŠB-Technické univerzitě v Ostravě v letech 2010-2013, na Ostravské univerzitě v Ostravě v letech
24 2010-2011.

25 V roce 2010 prof. Skala získal významné ocenění mezinárodní asociace pro počítačovou grafiku
26 **EUROGRAPHICS Association** (<http://www.eg.org>)

27 „**Fellow of the Eurographics Association**“

28 za dlouhodobé odborné výsledky a organizační aktivity v oblasti počítačové grafiky.

30 Poděkování

31 Je milou povinností poděkovat všem, kteří stimulovali moje odborné aktivity a které jsem měl tu čest
32 nejen potkat na Brunel University a na NATO Advanced Study Institutes, ale i s mnohými odborně
33 spolupracovat. Patří k nim zejména:

34 **Prof. Mike L.V. Pitteway**

Brunel University, U.K.



Prof. Jack E. Bresenham

IBM Corp., USA,
Winthrop University, USA



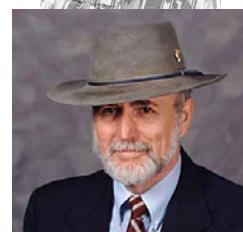
Prof. F.R.A. Hopgood

Rutherford Appleton
Laboratory, U.K.
Oxford Brookes University,
U.K.



Prof. David F. Rogers

U.S. Naval Academy, USA



35 Poděkování patří též studentům na Západočeské univerzitě v Plzni za jejich připomínky, zejména pak
36 kolegům Ing.Zuzaně Majdišové a Ing.Michalovi Šmolíkovi za konstruktivní návrhy a korekce textu.

1 Tento text vznikl na základě přednášek předmětu bakalářského studia

2

3 Základy počítačové grafiky (ZPG)

4

5 a přednášek předmětu navazujícího studia

6

7 Algoritmy a počítačová grafika (APG)

8

9 na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

10

11 Doprovodné materiály byly získány z WEBu; pokud není stanovenno jinak, jsou poskytnuty s licencí
12 Wikipedia Commons nebo jinou obdobnou. Za případné opomenutí se autorům předem omlouvám
13 a rád zahrnu na jejich práci příslušný odkaz.

14 Text neprošel žádnou jazykovou kontrolou a slouží výhradně pro pracovní potřebu.

15

16 Rád bych upozornil na dodatečné zdroje, které jsou k dispozici na DVD, a to:

- 17 • Záznam přednášek předmětu ZPG z roku 2008 ([CLICK off-line](#))
- 18 • Skala,V: Algoritmy počítačové grafiky I, skripta 2011 ([CLICK off-line](#))
- 19 • Skala,V.: Algoritmy počítačové grafiky II, skripta 2011 ([CLICK off-line](#))
- 20 • Skala,V.: Algoritmy počítačové grafiky III, skripta 2011 ([CLICK off-line](#))
- 21 • Skala,V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia, 1993
([CLICK off line](#))
[\(http://herakles.zcu.cz/~skala/ZPG/Svetlo_barvy_barevne_systemy.pdf\)](http://herakles.zcu.cz/~skala/ZPG/Svetlo_barvy_barevne_systemy.pdf)
- 22 • PowerPoint prezentace dřívějších přednášek ZPG s příklady s OpenGL ([CLICK off-line](#))

23 Ostatní relevantní dodatečné zdroje informací jsou pak vždy uvedeny v příslušné kapitole, resp. na
24 konci textu.

25

26

27

28

29

1 Základy počítačové grafiky (ZPG) - obsah přednášek

- 2 1. Úvod, typické aplikace počítačové grafiky a vizualizace dat. Základní architektura grafických
3 systémů a grafická rozhraní OpenGL/DirectX/SVG - principy.
- 4 2. Souřadné systémy v počítačové grafice, homogenní souřadnice a jejich geometrická
5 interpretace. Numerická reprezentace a stabilita výpočtů.
- 6 3. Základní geometrické transformace v E2 a E3, řetězení operací. Geometrické entity, princip
7 duality.
- 8 4. Transformace Window-Viewport. Promítání, rovinné projekce, pozice kamery.
- 9 5. Datové struktury, hierarchické modely a geometrické transformace
- 10 6. Světlo a barevné modely. Modely osvětlení a metody stínování. Textury a bitové mapy.
- 11 7. Základní algoritmy řešení viditelnosti, metoda sledování paprsku a radiační metoda.
- 12 8. Interpolace, křivky a plochy v počítačové grafice.
- 13 9. Metody ořezávání v E2 a E3, množinové operace s n-úhelníky.
- 14 10. Vizualizace dat: datové struktury, geometrie a data, výšková pole a iso-čáry/plochy,
15 zobrazení povrchů a skalárních polí (CT, MRI), zobrazení vektorových polí.
- 16 11. Animace, principy a inverzní kinematika
- 17 12. Rastrová grafika a základní algoritmy pro kreslení úseček a kružnic, algoritmy šrafování a
18 plnění, anti-aliasing.
- 19 13. Zvaná přednáška nebo 3D displeje, haptické systémy a systémy virtuální reality. Architektura
20 grafických systémů (GPU/CUDA/TESLA)
- 21 Relevantní přílohy jsou součástí přednášek, resp. jejich doplněním.

29 Kapitoly, resp. jejich části, pro předmět Základy počítačové grafiky – KIV/ZPG

<ul style="list-style-type: none">• 1.1 - 1.4• 2.2• 3.1 - 3.6• 4.1 - 4.6, 4.9 - 4.10• 5.1 - 5.3, 5.5• 7.1 - 7.3, 7.7 - 7.9• 9.1 - 9.2• 10.1 - 10.3, 10.7• 11.1 - 11.2, 11.5 - 11.8, 11.10	<ul style="list-style-type: none">• 12• 13• 14• 15• 16.1• 17• 19• 20
---	---

30
31 a relevantní přílohy.
32

1 Algoritmy a počítačová grafika (APG) - obsah přednášek

- 2 1. Úvod, organizace předmětu. Metodologie porovnávání algoritmů. Urychlovací techniky,
3 předzpracování, dělení prostoru.
- 4 2. Projektivní reprezentace a princip duality. Plückerovy souřadnice a jejich aplikace.
- 5
- 6 3. Geometrické transformace v E2 a E3 (vektorová formulace). Neplanární projekce a jejich
7 aplikace.
- 8 4. Interpolace uspořádaných dat. Parametrické křivky a plochy. Modelování povrchu tuhých
9 těles.
- 10 5. Geometrické výpočty a algoritmy výpočtů průsečíků geometrických entit.
- 11
- 12 6. Metody a datové struktury pro reprezentaci objektů a volumetrická data.
- 13
- 14 7. Modelování složitých objektů, hierarchické struktury, CSG stromy. Graf scény.
- 15
- 16 8. Generace povrchu z objemových dat a objektů zadaných implicitním popisem.
- 17
- 18 9. Modely osvětlení a metody stínování.
- 19
- 20 10. Algoritmy sledování paprsku, multispektrální rendering.
- 21
- 22 11. Principy radiační metody a path tracing. Image based rendering.
- 23
- 24 12. Animace, kinematika a inverzní kinematika.
- 25
- 26 13. Interpolace neuspořádaných dat. Základy geometrické algebry.
- 27
- 28

1. Úvod

1.1. Přehled vývoje počítačové grafiky

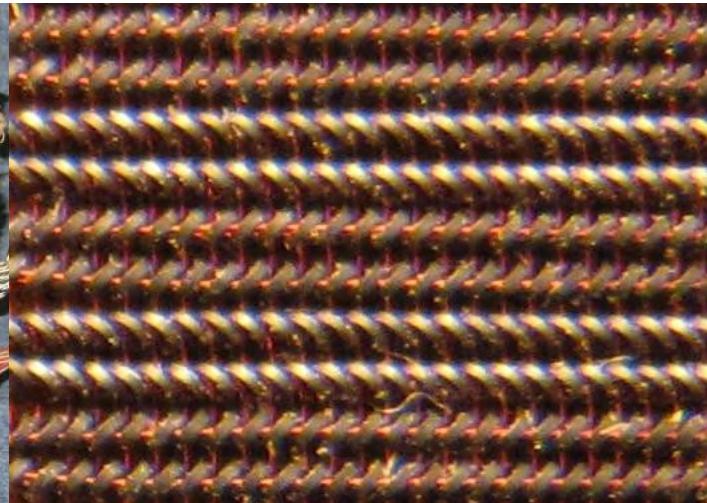
Počítačová grafika je poměrně mladá vědní oblast, která se formovala po II. světové válce, především po roce 1950, v oblastech spojených s vojenským průmyslem a vojenskými aplikacemi a byla především zpočátku chápána jen jako grafický výstup z počítače. Od jednoduchých grafických výstupů na „primitivních“ grafických výstupních zařízeních se rozvinula do nebývalé šíře nejen z teoretického hlediska, ale především v oblasti aplikací od počítačových her a tvorby filmů, technických i netechnických simulací s grafickým výstupem s možností interakce, až po aplikace haptických zařízení (zařízení se silovou zpětnou vazbou) pro tele-operace, vojenský výcvik s použitím 3D zobrazování a prostředků virtuální reality.

První aktivity lze datovat do roku 1954, kdy byl použit „první grafický display“ v rámci projektu SAGE (Semi-Automatic Ground Environment) jako součást obraněného protiraketového systému v době „studené války“. Je nutné zdůraznit, že použitý superpočítač SAGE měl v té době „super kapacitní“ paměť $4 \times 64\text{Kslov}$, 60 000 elektronek, vážil 250 tun, obsluha měla 100 osob a cena je odhadována na 8-12 miliard dolarů v relacích r. 1964!

17



Obr.1.1: Blok operační paměť 451x452 bitů
Velikost 12,7x12,7x18,7 [cm]



Obr.1.2: Detail paměti
Feritová jadérka mají 0,457 [mm]

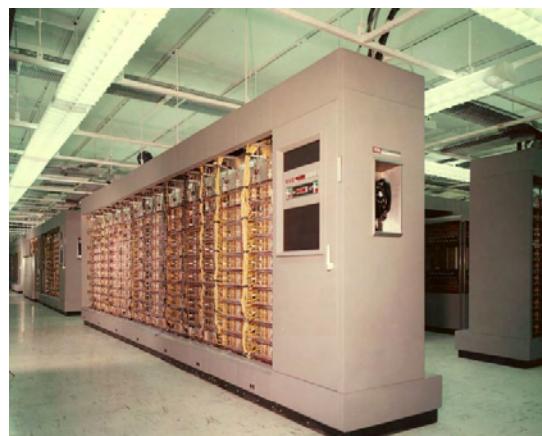
18

19

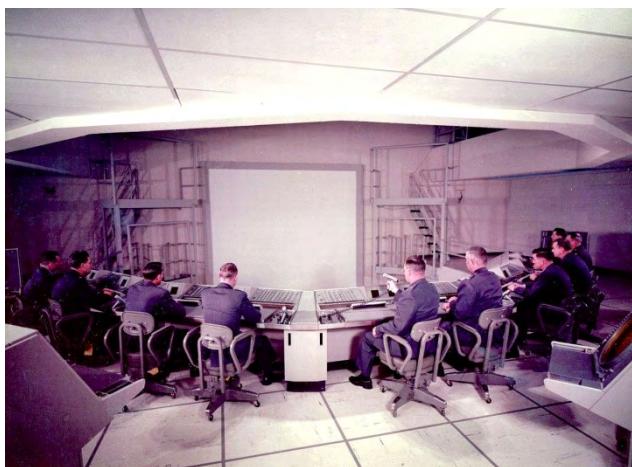
1



Obr.1.3: SAGE monitor



Obr.1.4: Typická konfigurace systému



Obr.1.5: Operační sál

2

3 Za skutečný začátek éry počítačové grafiky lze považovat však až vznik legendárního programu
4 SketchPad, který byl vytvořen Ivanem Sutherlandem v roce 1963 v rámci jeho PhD práce. Systém
5 umožňoval nejen zobrazování „složitých“ objektů, ale především interakci pomocí světelného pera.
6 To byl vlastně začátek oboru, který je dnes označován jako „Human Computer Interaction“ (HCI),
7 tedy interakce člověk-počítač.

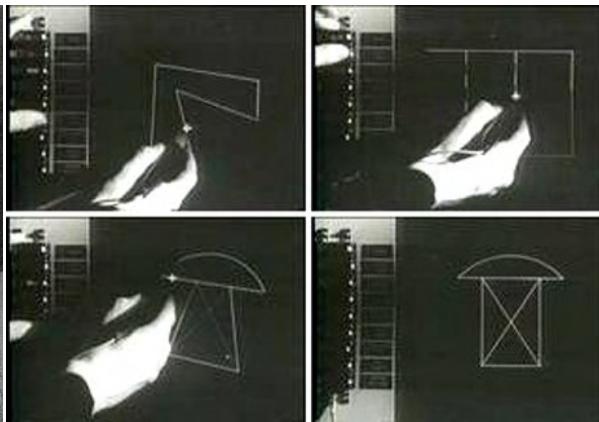
8

9 SketchPad systém byl realizován na MIT (Massachusetts Institute of Technology) na počítači Lincoln
10 TX-2, který měl „bájných“ 64Kslov o 36 bitech v době, kdy se používalo téměř výlučně dávkové
11 zpracování a děrné štítky.

12



Obr.1.6: SkatchPad systém



Obr.1.7: Typický výstup systému

[Click for video - Sutherland ScatchPad-YouTube](#)

1
2 Práce Ivana Sutherlanda a další výzkum a vývoj pak otevřely cestu k mnoha následným navazujícím
3 oblastem. Mezi nejdůležitější patří:

- 4 • počítačová grafika (Computer Graphics) – zabývající se především metodami generování
5 vizuálních objektů na základě jejich popisu geometrického apod. Hlavní aplikace byly
6 zpočátku v oblasti vojenské, později pak i v herním a zábavném průmyslu
- 7 • CAD/CAM systémy – které umožňují nejen vlastní zobrazování geometrických objektů, ale i
8 jejich strukturovaný návrh. Ve spojení se simulačními prostředky navíc umožňují ověření
9 jejich fyzikálních vlastností. Rozvoj CAD/CAM systémů byl dán především požadavky
10 leteckého, lodního a raketového průmyslu

11
12 K zásadnímu rozvoji počítačové grafiky však především přispěl rozvoj personálních počítačů
13 s legendárními počítači Sinclair ZX Spektrum se 48 KB paměti a programovacím jazykem Pascal 4T,
14 Apple II nebo Amiga (technické popisy viz Wikipedia.org).



ZX Spectrum – 48KB, 1982



Amiga, 1985



Apple II, 1997

Obr.1.8: První personální počítače

15 V té době také vznikaly první počítačové hry včele s legendárním Maniac Miner. Rozvoj her
16 inicializoval vývoj levných HW prostředků pro počítačovou grafiku s jejich masovou výrobou.



Obr.1.9: Maniac Miner

<http://www.youtube.com/watch?v=F7khL9Ms4ow> ([CLICK off-line](#) – MP4)

- 1
2 V současné době jsou principy vyvinuté v oblasti počítačové grafiky zastoupeny ve většině
3 výpočetních systémů, např. ve formě knihoven GUI (Graphical User Interface), nebo moderních
4 mobilních systémů.



Obr.1.10: Quake



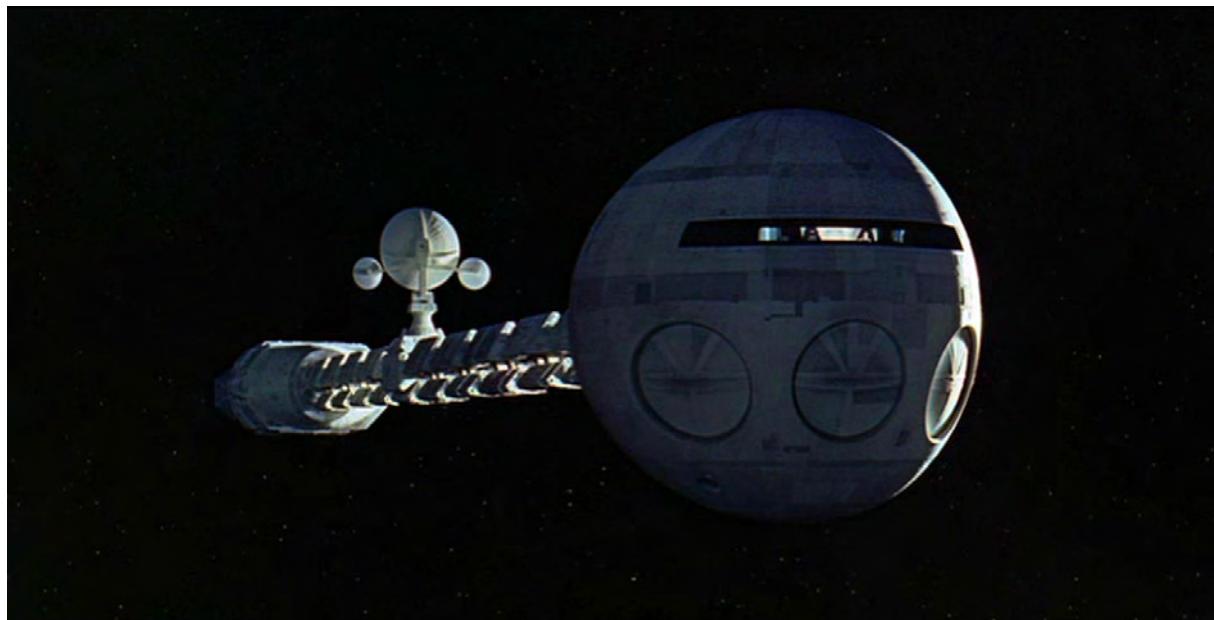
Obr.1.11: Toy Story

- 5 Za prudkým rozvojem technických prostředků počítačové grafiky je nutné vidět především herní a
6 filmový průmysl. Zajisté hlavními milníky bylo uvedení filmu Odysea roku 2001 Arthura C. Clarka
7 z roku 1968, uvedení 3D filmu Toy Story v roce 1995 a počítačové hry Quake v roce 1996.
8 Zajímavý přehled historie lze nalézt na http://en.wikipedia.org/wiki/Computer_graphics.
9

- 10 Počátky počítačové grafiky na Západočeské univerzitě, vlastně na jejím předchůdci Vysoké škole
11 strojní a elektrotechnické v Plzni, jsou datovány do roku 1975, kdy vzniká nový předmět Počítačová
12 grafika a umělá inteligence, který je následně vyučován. Z počátku byla počítačová grafika chápána
13 spíše jako matematická disciplína vzhledem k matematickým základům než jako disciplína patřící do
14 Computer Science, což bylo dánno dostupností programového vybavení a technických prostředků.

15 Uvedeme několik zajímavých oblastí aplikací počítačové grafiky:

- 16 • simulace v oblasti vojenství, ať už při nácviku vojenských operací, nebo např. při řešení
17 post-traumatických onemocnění, viz např. Virtuální krajina ([CLICK off-line](#)) z r.2002
- 18 • Simulace záplav v Plzni r. 2002:
 - 19 ○ Údolí Mže ([CLICK off-line](#)), Roudná ([CLICK off-line](#)),
20 vliv průtoku na zaplavení ([CLICK off-line](#))



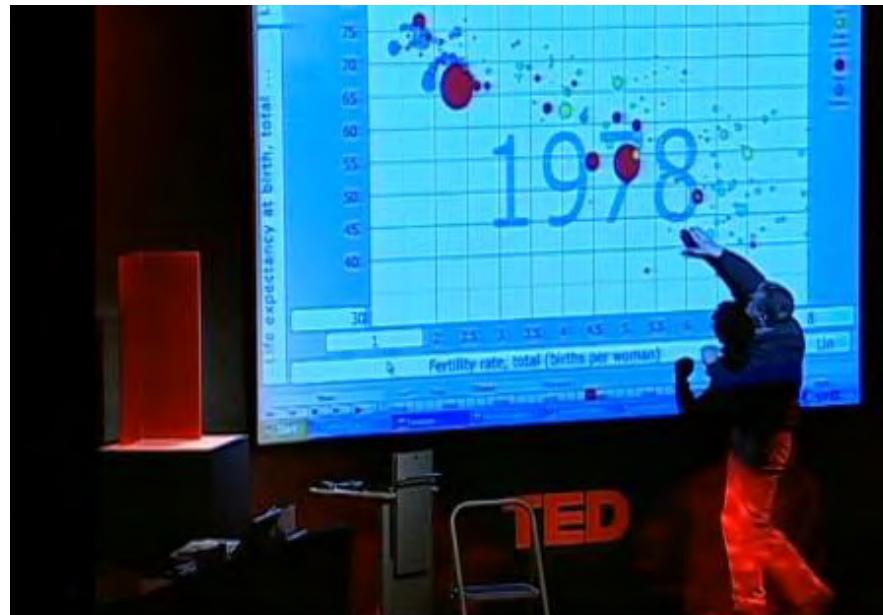
Obr.1.12: Vesmírná Odyssea 2001 ([CLICK off-line](#))
<http://www.youtube.com/watch?v=q3oHmVhviO8>

- 1 Vymezení oblasti počítačové grafiky není zcela jednoduché, neboť dnes již „prorostla“ do většiny
- 2 aplikací výpočetních a mobilních systémů. V současné době je chápána jako interdisciplinární obor
- 3 stojící na základech matematiky a computer science, nicméně prakticky zasahující dnes již do všech
- 4 vědních oblastí.



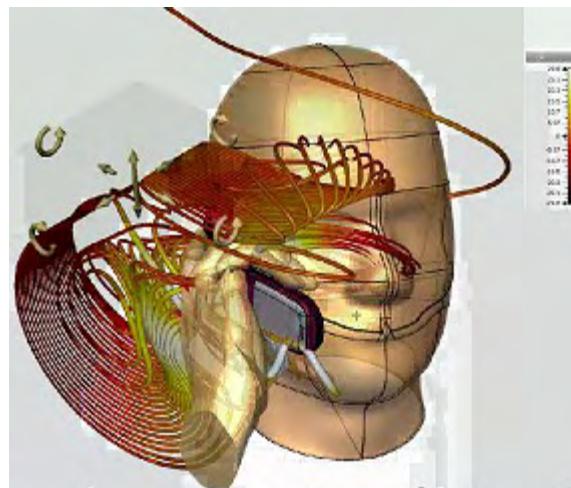
- 5 Obr.1.13: Dr. Albert Rizzo: Post Traumatic Stress Disorder, WSCG2008 ([CLICK off-line wmv](#))
- 6 Courtesy of NVIDIA
- 7

- 1 • vyhodnocení rozsáhlých dynamických dat, např. v oblasti ekonomie.
2 Hans Rosling v roce 2006 přednesl přednášku: „Stats that reshape your worldview“
3 (Statistika, která změní váš pohled na svět). Tato přednáška je pěknou ukázkou, jak důležitou
4 rolí hraje čas, resp. dynamika zobrazovaného fenoménu.



Obr.1.14: Rosling,H.: Statistika, která změní váš pohled na svět
([CLICK off-line](#))

- 5
6 • Simulace a analýza elektromagnetického pole mobilního telefonu



Obr.1.15: Simulace elektromagnetického pole mobilního telefonu
([CLICK off-line](#))

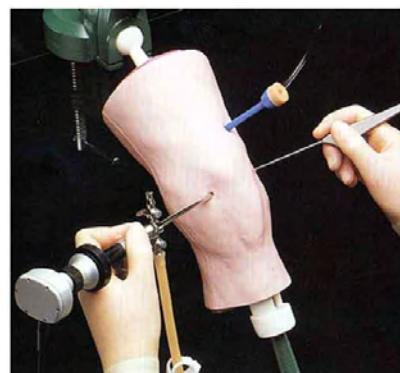
- 1 Zpracování grafické informace se zejména vyznačuje:
- 2 • velkým objemem zpracovávaných dat
- 3 • vysokými nároky prakticky na všechny parametry výpočetního systému, zejména pak na paměť
- 4 • numerickou náročností jednotlivých geometrických a numerických výpočtů

5 Pro návrh a realizaci metod počítačové grafiky je nutné pochopit základní principy algoritmů a metod, které však nelze jen naprogramovat, ale je nutné je optimalizovat vzhledem k velkému počtu prováděných operací. Mnohé také závisí na použitých technických prostředcích, architektuře výpočetního systému, na použitém CPU, vlastnostech sběrnice pro přenos dat a použitých specializovaných procesorů, např. GPU.

11 Počítačová grafika je chápána obvykle v úzkém slova smyslu, tj. tzv. „generativní“ grafika, kdy se ze zadaných dat, např. geometrického modelu, generuje vizuální výstup. V širším slova smyslu pak počítačová grafika dnes zahrnuje oblasti vizualizace dat a informací (Data Visualization & Information Visualization), zpracování obrazu (Image Processing), rozpoznávání obrazu, resp. vzorů (Pattern Recognition), počítačového vidění (Computer Vision), virtuální reality (Virtual Reality) a haptických systémů (Haptic Systems), tj. systémů se silovou zpětnou vazbou. Počítačová grafika tvoří podstatnou část nejen CAD/CAM systémů, tj. např. systémů pro návrh mechanických součástí, letadel, lodí, raket a kosmických systémů, systémů GIS (Geographical Information Systems) atd., ale i systémů výpočetní fyziky, chemie nebo biologie, kde je rozhraní pro uživatele pro zobrazení výsledků z výpočetního systému a interakci s počítačovým modelem.



Obr.1.16: Haptické pero



Obr.1.17: Operace kolena – trénink s haptickým zařízením

22

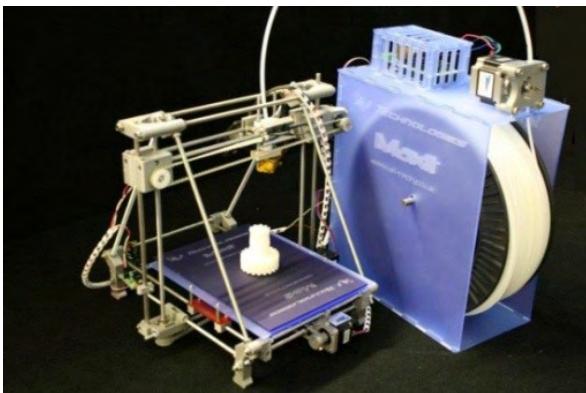
Obr.1.18: Vizualizace mlhoviny ([CLICK](#) off-line)Obr.1.19: Vizualizace informací ([CLICK](#) off-line)

23

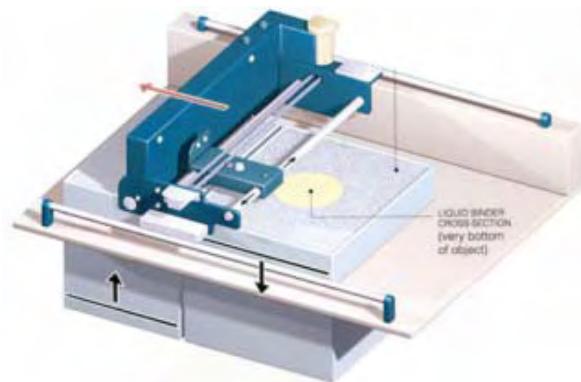
Obr.1.20: Virtuální realita ([CLICK](#) off-line)Obr.1.21: Simulace a vizualizace DNA chromosomů ([CLICK](#) off-line)

1

- 2 Je možné též „vytisknout“ generovaný objekt na 3D tiskárně (tzv. rapid prototyping), tj. vytvořit 3D fyzický model, a to i velkých rozměrů. V současné době se testuje možnost 3D tisku i pro tak náročné použití jako jsou trysky raketových motorů.



Obr.1.22: 3D tiskárna založená na tavícím se plastu



Obr.1.23: 3D tiskárna – lepený prášek

5



Obr.1.24: Lebka vytiskněná na 3D tiskárně

Obr.1.25: 3D tisk velkých objektů

6

- 7 Poměrně často se používá stereoskopie pro generaci 3D obrazů ve spojení se systémy virtuální reality. V současné době je možné nejen objekty zobrazovat na 2D průmětně, např. na stínítku obrazovky, ale též zobrazovat pomocí 3D zobrazení s použitím různých fyzikálních principů.

1 1.2. Počítačová grafika a ostatní oblasti

2 Je zřejmé, že počítačová grafika souvisí s mnoha vědními oblastmi, zejména pak s oblastí percepce
 3 člověka, tj. jak člověk vnímá vnější svět a naopak, jak jej ovlivňuje. Je proto otázkou, v jakém vztahu
 4 jsou vjemové „kanály“ člověka a která oblast se věnuje zpracování těchto signálů.

		výstup				
		popis	vizuální	zvukový	taktilelní	orientace v prostoru
vstup	popis	symbolická manipulace	počítačová grafika	hlasový výstup	haptický systém	navigace?
	vizuální	rozpoznávání obrazu	zpracování obrazu		hmatový grafický display?	
	zvukový	rozpoznávání zvuku	vizualizace zvuku?	zpracování zvuku		
	taktilelní	rozpoznávání hmatové informace	dotyková obrazovka?			
	*orientace v prostoru	záznam tras GPX ?	mapa?	hlasová navigace?		

5 * Orientace v prostoru – vnímání orientace polohy v gravitačním poli, zrychlení apod. je dalším
 6 faktorem, který nebývá zmíněn v souvislosti s lidskými senzory, nicméně je zde podmíněnost
 7 existencí gravitačního pole, principy zachování energie apod. Je vhodné upozornit na fakt, že člověk
 8 vnímá i změnu zrychlení, tzv. ryz, což je důležitý faktor např. pro pocit komfortu ve vlaku musí být
 9 minimalizován, naopak v případě zábavního průmyslu se maximalizuje.

10 Tab.1.1

11 Tab.xx

12

13

1.3. Architektura grafických systémů

2 Architektura grafických systémů se v zásadě ustálila na dvou hlavních architekturách, a to:

- 3 architektura pro aplikace v oblasti generovaných obrazů, tj. v oblasti počítačové grafiky
4 založené na GPU procesorech, jejichž hlavními reprezentanty jsou NVIDIA a ATI
- 5 • architektura pro zpracování obrazu, videa a aplikace v oblasti počítačového vidění, kde
6 hlavním reprezentantem je MATROX. Systémy jsou založeny převážně na architektuře FPGA.

7

8 V současné době je zřejmě nejvýkonnější systém NVIDIA Kepler 110, který má 7,1 miliardy tranzistorů
9 a poskytuje výpočetní výkon přes 1 Terra FLOP v dvojnásobné přesnosti.



Obr.1.26: Architektura NVIDIA Kepler 110

10

11 Podrobnější informace lze nalézt na:

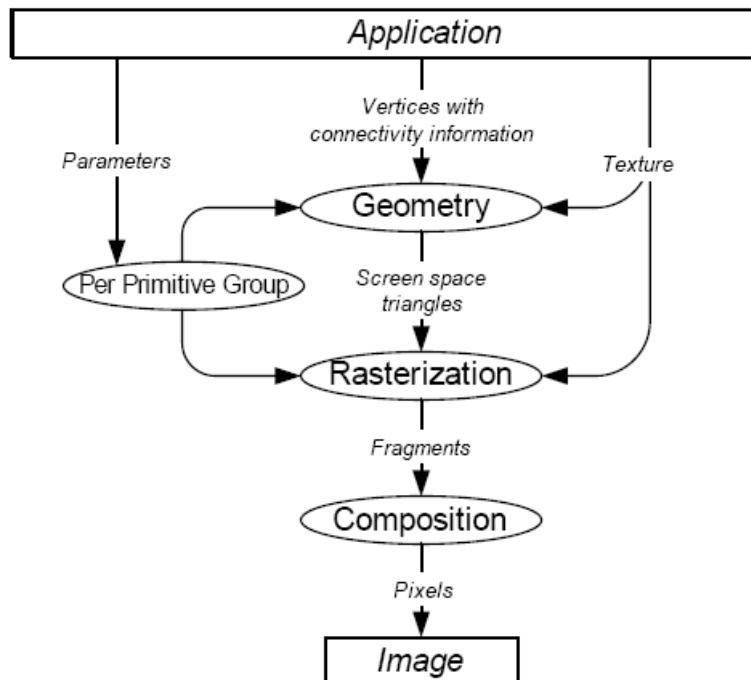
- 12 • <http://www.nvidia.com/object/nvidia-kepler.html>
- 13 • <http://www.youtube.com/course?list=EC4A8BA1C3B38CFCA0> (10 hod. přednášek)

14

15 V současné době jsou systémy Kepler určeny především pro výpočty na GPU, tzv. GPGPU Computing
16 (General Purpose GPU).

17

1 V následujícím se budeme především zabývat základními vlastnostmi současných běžně dostupných
2 grafických systémů založených na GPU architektuře, jejichž základní vlastnosti jsou znázorněny na
3 Obr.1.27.
4

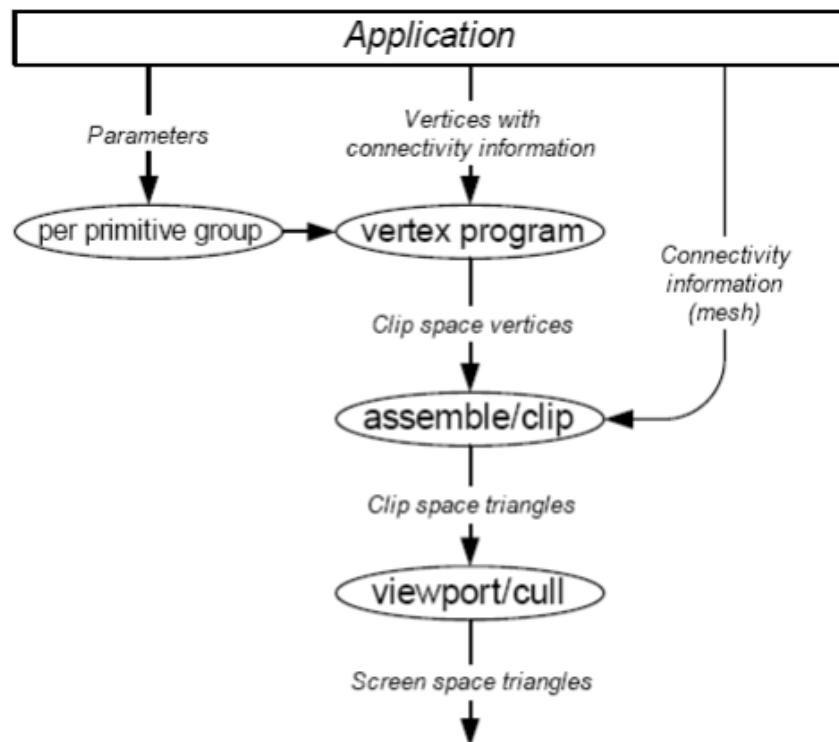


Obr.1.27: Schéma zpracování geometrických primitiv

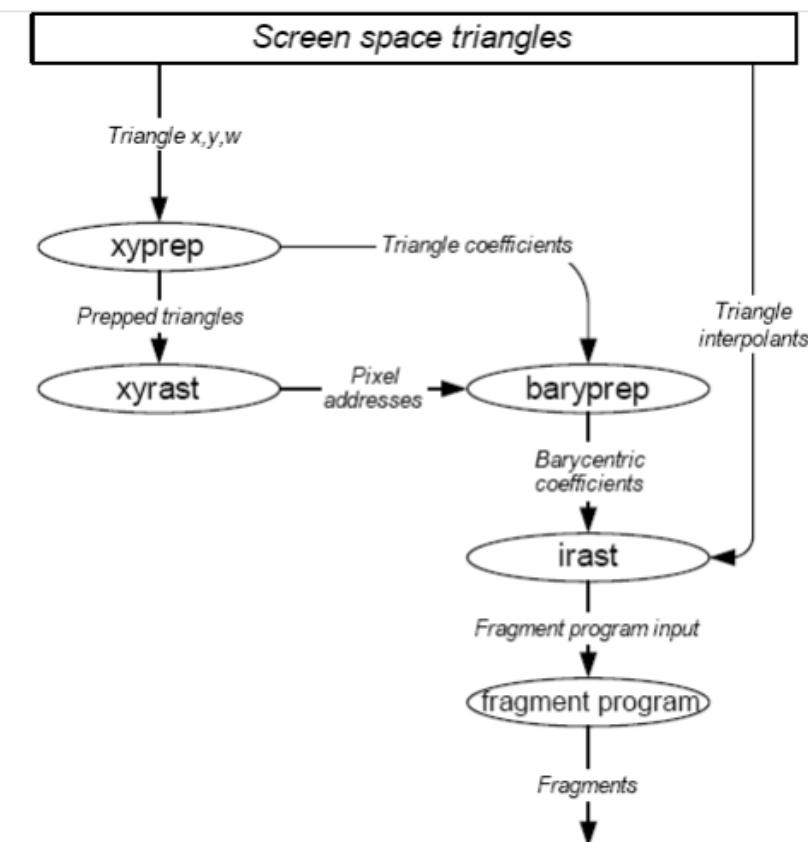
Podrobnější informace lze nalézt např. v:

http://graphics.stanford.edu/papers/jowens_thesis/jowens_thesis.pdf

Poznámka – pod pojmem vertex je vhodné si představit některý bod základního primitiva, tj. např. koncové body úsečky nebo vrcholy trojúhelníka apod. Rasterizace je proces, kdy se trojúhelník daný svými vrcholy převede do rastrové podoby, tj. do pixelů odpovídající barvy, a výsledek se nazývá fragment. Následně se pak jednotlivé fragmenty, tj. diskrétní reprezentace jednotlivých grafických primitiv, např. trojúhelníků, sestaví do finálního obrazu.

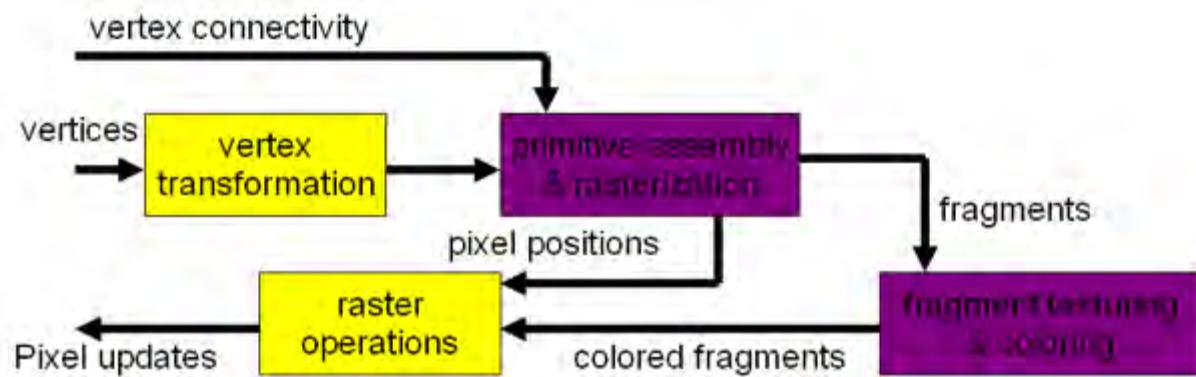
1
2
3

Obr.1.28: Zpracování geometrie

4
5

Obr.1.29: Rasterizace geometrických objektů

1



2

3

4 Obr.1.30: Základní schéma zpracování geometrických primitiv na GPU architekturách

5

1.4. Grafické knihovny

2 Pro aplikace počítačové grafiky se používají programovací nástroje, které:

- 3 • produkují přímo spustitelný nativní kód, např. C, C++, Pascal/Delphi apod.
- 4 • produkují interpretovanou meziformu (např. IL - Intermediate Language), a to např. C# nebo
- 5 Java
- 6 • jsou přirozeně interpretační, jako je např. Python

7 Jednou z nejrozšířenějších grafických knihoven je OpenGL.

10 1.4.1. OpenGL

11 Knihovna OpenGL je dostupná jak na platformě MS Windows, tak i na UNIX.

12 Velmi dobrý tutoriál a příklady jsou k dispozici na:

- 13 • <http://www.opengl.org>, resp.
http://www.opengl.org/archives/resources/code/samples/glut_examples/examples/examples.html
- 14 • NEHE - http://nehe.ceske-hry.cz/tut_obsah.php
- 15 • Tutorial prezentovaný na konferenci SIGGRAPG 2013
<https://www.khronos.org/developers/library/2013-siggraph-opengl-bof>

19 Knihovna OpenGL je použitelná jak pro C, C++, C#, Java, tak i pro další.

21 Pro realizaci aplikací v prostředí Python, pak lze doporučit použití PyOpenGL,
22 viz <http://pyopengl.sourceforge.net/>

23 Další zdroje:

- 24 • Tichava,J.: OpenGL v Javě, Bc. kvalifikační práce, ZČU, Plzeň 2007
<http://jogl.tichava.cz/files/bp.pdf>

27 1.4.2. DirectX

28 Knihovna DirectX je knihovna primárně zaměřená na platformu MS Windows.

29 Tutoriály jsou k dispozici na:

- 30 • <http://www.directxtutorial.com>
- 31 • [http://msdn.microsoft.com/en-us/library/windows/desktop/bb153264\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/bb153264(v=vs.85).aspx)

32 V češtině jsou dobrým zdrojem:

- 33 • <http://directx.kvalitne.cz/index.php?id=6>
- 34 • <http://www.monade.cz/item.php?item=11>

35 Porovnání (poněkud staršího data) obou knihoven je k dispozici na adrese:

- 36 • http://woq.nipax.cz/cl_gidx.php

38 DirectX pipeline

39 Podrobné informace viz

40 [http://msdn.microsoft.com/en-us/library/windows/hardware/ff569022\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff569022(v=vs.85).aspx)

41

42

1 **1.4.3. Příklad s OpenGL**

2 Pro názornost uvedeme jednoduchý příklad nakreslení
 3 krychle s použitím knihovny OpenGL.

```

 4
 5 /* Copyright (c) Mark J. Kilgard, 1997. */
 6 /* This program is freely distributable without licensing fees
 7 and is provided without guarantee or warrantee expressed
 8 or implied. This program is -not- in the public domain. */
 9 /* This program was requested by Patrick Earl; hopefully
10 someone else will write the equivalent Direct3D immediate
11 mode program. */
12
13 #include <GL/glut.h>
14
15 GLfloat light_diffuse[] = {1.0, 0.0, 0.0, 1.0}; /* Red diffuse
16 light. */
17 GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0}; /* Infinite light location. */
18 GLfloat n[6][3] = { /* Normals for the 6 faces of a cube. */
19   {-1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {1.0, 0.0, 0.0},
20   {0.0, -1.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, -1.0} };
21 GLint faces[6][4] = { /* Vertex indices for the 6 faces of a cube. */
22   {0, 1, 2, 3}, {3, 2, 6, 7}, {7, 6, 5, 4},
23   {4, 5, 1, 0}, {5, 6, 2, 1}, {7, 4, 0, 3} };
24 GLfloat v[8][3]; /* Will be filled in with X,Y,Z vertexes. */
25
26 void drawBox(void)
27 {
28   int i;
29   for (i = 0; i < 6; i++) {
30     glBegin(GL_QUADS);
31     glNormal3fv(&n[i][0]);
32     glVertex3fv(&v[faces[i][0]][0]);
33     glVertex3fv(&v[faces[i][1]][0]);
34     glVertex3fv(&v[faces[i][2]][0]);
35     glVertex3fv(&v[faces[i][3]][0]);
36   }
37 }
38
39 void display(void)
40 {
41   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
42   drawBox();
43   glutSwapBuffers();
44 }
```



Obr.1.31: Příklad výstupu OpenGL

```
1 void init (void)
2 {
3     /* Setup cube vertex data. */
4     v[0][0] = v[1][0] = v[2][0] = v[3][0] = -1;
5     v[4][0] = v[5][0] = v[6][0] = v[7][0] = 1;
6     v[0][1] = v[1][1] = v[4][1] = v[5][1] = -1;
7     v[2][1] = v[3][1] = v[6][1] = v[7][1] = 1;
8     v[0][2] = v[3][2] = v[4][2] = v[7][2] = 1;
9     v[1][2] = v[2][2] = v[5][2] = v[6][2] = -1;
10
11    /* Enable a single OpenGL light. */
12    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
13    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
14    glEnable(GL_LIGHT0);
15    glEnable(GL_LIGHTING);
16
17    /* Use depth buffering for hidden surface elimination. */
18    glEnable(GL_DEPTH_TEST);
19
20    /* Setup the view of the cube. */
21    glMatrixMode(GL_PROJECTION);
22    gluPerspective( /* field of view in degree */ 40.0,
23                   /* aspect ratio */ 1.0,
24                   /* Z near */ 1.0, /* Z far */ 10.0);
25    glMatrixMode(GL_MODELVIEW);
26    gluLookAt(0.0, 0.0, 5.0, /* eye is at (0,0,5) */
27              0.0, 0.0, 0.0, /* center is at (0,0,0) */
28              0.0, 1.0, 0.); /* up is in positive Y direction */
29
30    /* Adjust cube position to be asthetic angle. */
31    glTranslatef(0.0, 0.0, -1.0);
32    glRotatef(60, 1.0, 0.0, 0.0);
33    glRotatef(-20, 0.0, 0.0, 1.0);
34 }
35
36 int main(int argc, char **argv)
37 {
38     glutInit(&argc, argv);
39     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
40     glutCreateWindow("red 3D lighted cube");
41     glutDisplayFunc(display);
42     init();
43     glutMainLoop();
44     return 0;      /* ANSI C requires main to return int. */
45 }
```

2. Algoritmy a složitost algoritmů

Algoritmus neformálně řečeno je postup, který řeší daný problém pro množinu vstupních dat s předpokládanými vlastnostmi a vede ke správnému výsledku v akceptovatelném čase. Je celá řada algoritmů, která řeší danou úlohu, viz např. algoritmy řazení. Nicméně každý algoritmus má jinou časovou a paměťovou náročnost, tzv. složitost. Proto je nutné algoritmy posuzovat zejména z hlediska časové a paměťové složitosti.

2.1. Typy algoritmů

Většina běžných algoritmů je tzv. deterministického typu. Existují však i jiné typy algoritmů, které umožňují řešení, např. přibližné, problémů, jejichž řešení není algoritmicky známo, resp. je příliš složité.

Algoritmy se dají v zásadě rozdělit takto:

$$\text{algoritmus} \left\{ \begin{array}{l} \text{deterministický} \left\{ \begin{array}{l} \text{standardní} \\ \text{s heuristikou} \end{array} \right. \\ \text{stochastický} \left\{ \begin{array}{l} \text{standardní} \\ \text{s dodatečnou heuristikou} \end{array} \right. \end{array} \right.$$

Deterministický algoritmus je algoritmus, který pro daná vstupní data vždy vede ke správnému výsledku, např. výpočet maximální vzdálenosti d dvou bodů v rovině. Standardní algoritmus je pak přímou realizací formulace problému:

$$d = \max\{\|x_i - x_j\|\} \quad i, j = 1, \dots, N$$

kde: N je počet daných bodů. Složitost tohoto algoritmu je pak $O(N^2)$. Nicméně lze použít heuristiku, která umožňuje snížení výpočetní náročnosti na očekávatelnou složitost $O_{expected}(N)$, přičemž výsledek je vždy správný, viz

- Skala,V.: Fast Oexpected(N) Algorithm for Finding Exact Maximum Distance in E2 Instead of $O(N^2)$ or $O(N \lg N)$, ICNAAM 2013, Rhodos, Greece, AIP Conf.Proceedings No.1558, pp.2496-2499, AIP Publishing, 2013

Je zřejmé, že pro větší počet bodů, např. pro $N \in \langle 10^5, 10^{10} \rangle$, hraje složitost algoritmu významnou úlohu z hlediska časové výpočetní náročnosti.

Jde tedy o algoritmus deterministický s heuristikou. Dalším příkladem může být algoritmus řešení šachových úloh, tj. úlohy 13. tahem mat s použitím *and/or* grafu.

Na druhé straně jsou používané i algoritmy stochastické, např. založené na metodě Monte Carlo apod. Standardní algoritmy používající metodu Monte Carlo se používají nejen v numerické matematice, ale i např. pro výpočet světelných poměrů v radiační metodě, viz kap. 16 (Radiační metoda). Mezi stochastické algoritmy s dodatečnou heuristikou lze zařadit např. šachové algoritmy. Typickou ukázkou algoritmů stochastických jsou pak genetické algoritmy, které hledají např. řešení daného problému s hodnocením „kvality“ dosažení cíle, nebo šachové hry.

34

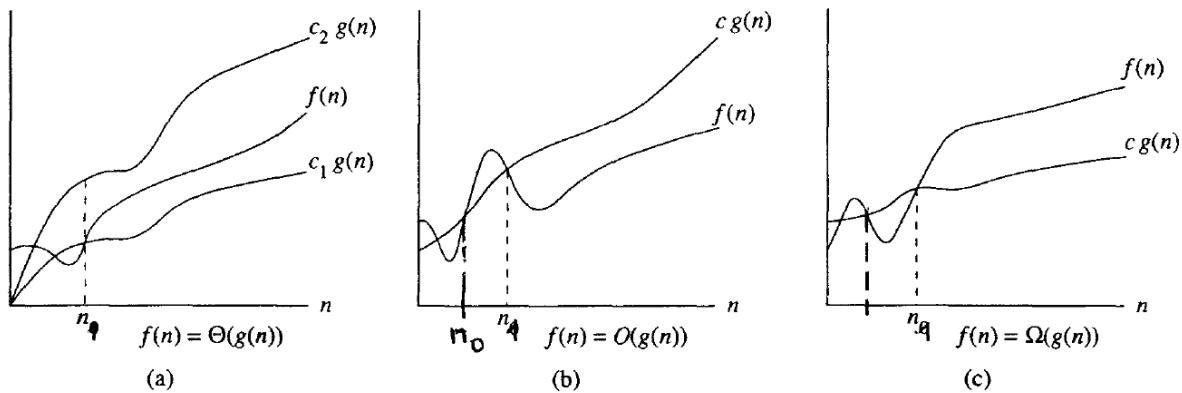
1 2.2. Posuzování algoritmů

2 Posuzování algoritmů je ne zcela triviální záležitostí, neboť v zásadě jde o více-kriteriální optimalizační
 3 problém. U mnoha algoritmů je sice důležitá rychlosť a paměťová náročnost, nicméně v mnoha
 4 případech je důležitým faktorem robustnost algoritmu vůči specifickým situacím. U geometrických
 5 úloh jde o robustnost řešení situací blízkých singulárním situacím, např. průsečík přímek, které jsou
 6 téměř rovnoběžné. Je zřejmé, že rychlosť algoritmu je také určena vlastní implementací a
 7 hardwarem, na kterém je vlastní výpočet realizován.

8 2.2.1. Složitost algoritmů

9 Při posuzování algoritmů je nutné zejména posuzovat:

- 10 • složitost předzpracování dat, pokud algoritmus používá předzpracování (pre-processing)
- 11 • složitost vlastního zpracování (run-time)
- 12 • složitost paměťovou



13 Obr.2.1: Klasifikace složitostí

14 Pro posouzení asymptotické složitosti se obvykle používají notace Θ , O , Ω (někdy se používá ω) a
 15 jejich význam je znázorněn na Obr.2.1, přičemž $c, c_1, c_2 > 0$ jsou konstanty.

16 O algoritmu $f(n)$ řekneme, že je asymptotické složitosti $O(g(n))$, jestliže existuje konstanta c a
 17 hodnota n_1 taková, že pro $\forall n \geq n_1$ platí, že $f(n) < c g(n)$.

18 V praxi je však nutné rozlišovat:

- 19 • asymptotickou složitost, která je složitostí algoritmu pro $n \rightarrow \infty$.
- 20 • složitost algoritmu (časovou, paměťovou) pro interval počtu primitiv zpracovávaných dat,
 21 obvykle $n \in \langle n_0, n_1 \rangle$

22 Je nutné nahlédnout, že v dnešních aplikacích je kritickým faktorem více rychlosť přenosu dat
 23 z paměti do procesoru a využití paměťové koherence, než vlastní rychlosť procesoru. Také se pro
 24 velké hodnoty n začne projevovat faktor stránkování paměti, resp. vliv virtuální paměti apod.

25 Na Obr.2.1.b je situace, kdy algoritmus se složitostí $O(f(n))$ se asymptoticky chová lépe. Pro reálnou
 26 aplikaci, kdy počet zpracovávaných primitiv $n \in \langle n_0, n_1 \rangle$, je však vhodné využít algoritmu se složitostí
 27 $O(g(n))$.

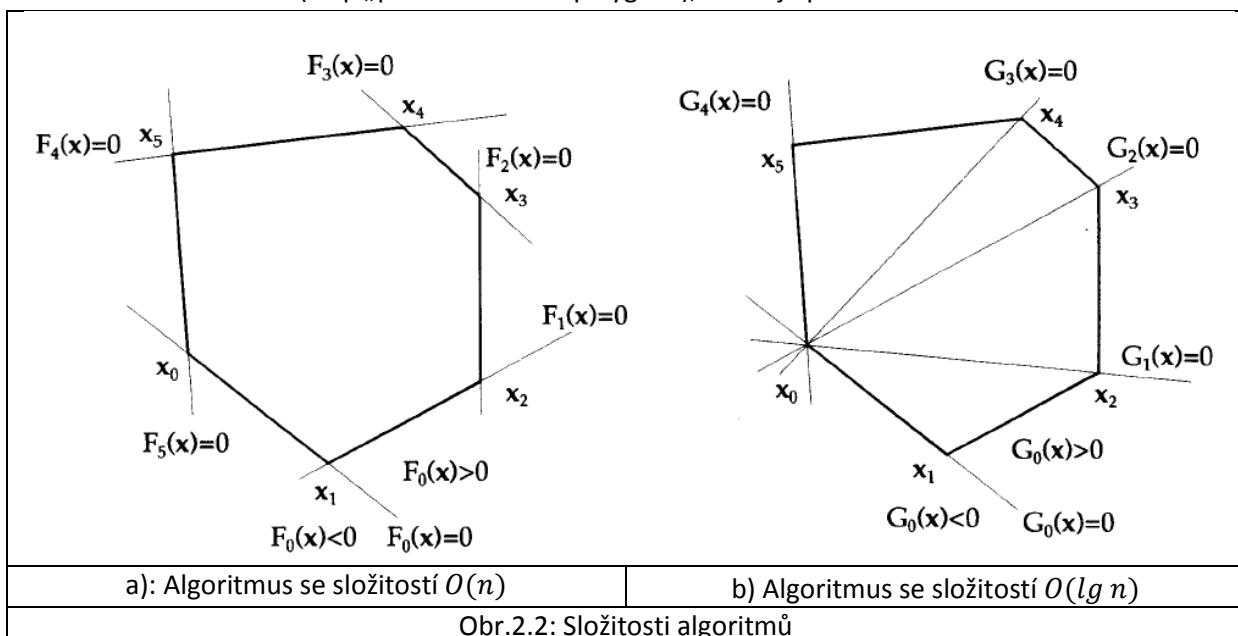
28 K experimentálnímu ověření složitosti algoritmu se většinou používají regresní techniky, resp.
 29 metoda nejmenších čtverců, viz kap.10.5 (Metoda nejmenších čtverců).

1 Při zpracování většího objemu dat je nutné se zamyslet nad tím, jak daný algoritmus zrychlit v daném
 2 kontextu, tj. s ohledem na předchozí a následný způsob zpracování a již použité datové struktury, aby
 3 nedocházelo ke zbytečnému přepisování, resp. kopírování dat apod.

4 K urychlení algoritmu jsou k dispozici v zásadě následující možnosti:

- 5 • *nalezení jiného algoritmu*, který má výhodnější časovou nebo paměťovou složitost, např.
 6 s využitím specifických vlastností předpokládaných datových množin
- 7 • *předzpracování* (pre-processing), který se vyplácí, pokud jde o velké množství
 8 zpracovávaných dat, např. test „point-in-convex polygon“ pro velké množství bodů a
 9 konstantní konvexní n-úhelník
- 10 • *paralelizace algoritmu* – zde je nutno zdůraznit, že např. pro případ, kdy je možno zcela
 11 paralelizovat 97% kódu, získáme při nekonečném počtu procesorů urychlení cca 35 krát
 12 (viz Amdalův zákon http://en.wikipedia.org/wiki/Amdahl's_law).

13
 14 Ukažme si výše uvedené přístupy na velmi jednoduchém případě, a to na testu, zda bod je uvnitř
 15 konvexního n-úhelníka (tesp „point-in-convex-polygon“), kde n je počet vrcholů n-úhelníka.



16
 17 Běžně známé algoritmy pro test „point-in-convex polygon“ jsou:
 18 • *algoritmus se složitostí $O(n)$* - konvexní n-úhelník je dán jako průnik polorovin a testuje se
 19 poloha bodu vůči každé polorovině. Tento algoritmus je evidentně složitosti $O(n)$. Sice
 20 využívá vlastnost konvexity n-úhelníka, avšak nevyužívá zásadní vlastnosti uspořádání indexů
 21 vrcholů. Úloha je vlastně ekvivalentní úloze, zda bod leží uvnitř průniku polorovin, které tvoří
 22 daný konvexní n-úhelník.

- 23 • *algoritmus se složitostí $O(\lg n)$* - algoritmus využívá vlastnosti uspořádanosti vrcholů ve
 24 směru nebo proti směru hodinových ručiček. Algoritmus je založen na půlení intervalu
 25 indexů, podobně jako metoda půlení intervalu pro řešení nelineární rovnice. Tento
 26 algoritmus se považuje za optimální.

27 Algoritmus nejdříve zjistí, zda daný bod x se nachází ve výseči $x_5x_0x_1$, viz Obr.2.2.b. Poté se
 28 určí „poloviční“ index a otestuje se, zda daný pod x je uvnitř výseče $x_5x_0x_3$ nebo $x_3x_0x_1$.
 29 Poté se opět interval indexů rozpůlí atd. Na konci je pak výseč s hranou x_kx_{k+1} , vůči které je

- 1 nutné udělat finální test, zda bod x je uvnitř či vně. Je zřejmé, že jde o algoritmus půlení
 2 intervalu a tedy složitost algoritmu je $O(\lg n)$.
- 3 • *algoritmus s předpracováním s run-time složitostí $O(1)$* - algoritmus je založen na
 4 předpracování, jehož složitost závisí na geometrických vlastnostech n-úhelníka, tj. na pozici
 5 vrcholů konvexního n-úhelníka. Předpracování je složitosti $O(m n \lg n)$, kde m je faktor
 6 daný geometrickým rozložením vrcholů.
 - 7 • *paralelní algoritmus* – brutální algoritmus může být založen na přímočaré paralelizaci
 8 algoritmu se složitostí $O(n)$ a zdánlivě tak dostaneme algoritmus se složitostí $O(1)$, což však
 9 není pravda, neboť výsledky získané z jednotlivých procesů musíme vyhodnotit a zřejmě tak
 10 dostáváme složitost $O(\lg n)$ nebo $O(n)$.

11 Výše uvedené algoritmy, kromě paralelního řešení, a jejich porovnání viz:

- 12 • Skala,V.: Trading Time for Space: an $O(1)$ Average time Algorithm for Point-in-Polygon
 13 Location Problem. Theoretical Fiction or Practical Usage? Machine Graphics and Vision,
 14 Vol.5., No.3., pp. 483-494, , ISSN 1230-0535, 1996. ([CLICK](#) off-line)

17 2.2.2. Robustnost algoritmu

18 Robustnost algoritmů je jedním z velmi důležitých aspektů pro hodnocení algoritmů. V oblasti
 19 geometrických výpočtů je nutné respektovat principiální problém, který je dán nepřesností diskrétní
 20 reprezentace spojitého geometrického aparátu, kdy z čistě matematického hlediska je výpočet
 21 vzorce, rovnice apod. dán naprostě přesně, avšak v „počítačové“ reprezentaci, je výsledek určen
 22 s nějakou chybou ε , jejíž velikost není ani konstantní ani známá.

23 Numerická přesnost je určena primárně reprezentací hodnot na daném výpočetním prostředku
 24 a standardizovanou normou IEEE 578-2008, podrobněji viz kap.3.6 (Numerická reprezentace a
 25 stabilita výpočtů). Příklady numerických nestabilit a jejich faktické důsledky jsou uvedeny v kap.3.7
 26 (Příklady katastrof způsobených výpočetními chybami).

27 Součástí posuzování robustnosti algoritmů je i způsob vlastního kódování, kdy např. výpočet
 28 směrnice k přímky je dán vztahem:

$$k = \frac{y_i - y_j}{x_i - x_j} \quad (2.1)$$

30 přičemž není zaručeno, že $x_i \neq x_j \forall i, j$ apod.

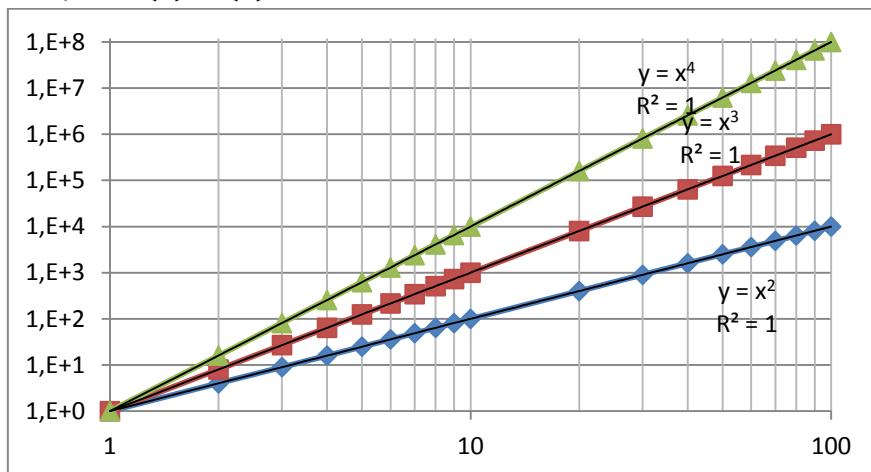
32 Bohužel takové „implicitní“ předpoklady, které nejsou obecně zaručitelné, se běžně při kódování
 33 používají. Jak asi bude výše uvedený výpočet stabilní, pokud:

$$|x_i - x_j| < \varepsilon \quad (2.2)$$

35 a jak velké, resp. malé má ε být?

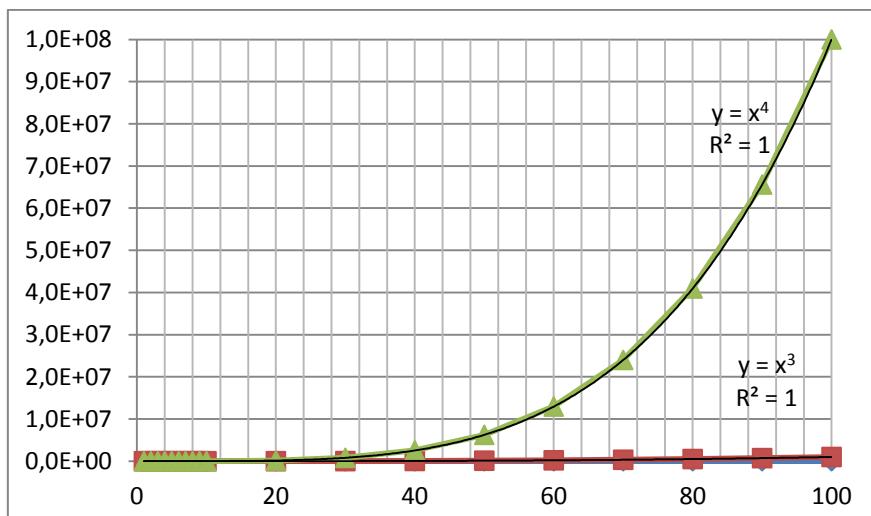
2.3.Experimentální vyhodnocení výpočetní náročnosti algoritmu

Experimentální porovnání algoritmů je nezbytnou nutností pro posouzení vlastností algoritmů. Vzhledem k tomu, že časová a paměťová náročnost se porovnává pro reprezentativní data, je nutné experimenty provést pro různý počet zpracovávaných dat. Označíme-li n počet zpracovávaných dat, pak budeme posuzovat paměťovou $mem(n)$ [Byte] a časovou $t(n)$ [s] náročnost pro velký rozsah dat, např. $n \in (10^6, 10^{12})$. To znamená, že nelze použít lineární měřítko na ose pro hodnoty n , ale měřítko logaritmické. Pro posouzení složitosti je pak vhodné použít logaritmické měřítko i pro svislou osu, tj. pro hodnoty $mem(n)$ a $t(n)$.



Obr.2.3: Graf závislosti výpočetní složitosti s logaritmickým měřítkem

Rozhodně není vhodné používat lineární měřítko na osách, neboť pro velký rozsah dat toto vede ke grafům, které nelze v zásadě spolehlivě interpretovat, viz Obr.2.4.



Obr.2.4: Graf závislosti výpočetní složitosti s lineárním měřítkem

Připomeňme, že v případě logaritmického měřítka na obou osách, se větší mocnina ve složitosti projeví větším sklonem regresní křivky grafu, viz Obr.2.3 a Obr.2.4.

Takže nyní umíme vyhodnotit chování algoritmu jako takového, a to jak z hlediska paměťové náročnosti, tak i náročnosti časové. Otázkou je, jak porovnávat algoritmy vzájemně.

2.4.Experimentální porovnávání algoritmů

Uvažme dva algoritmy, které chceme porovnat. Nový algoritmus má časovou náročnost $t_{new}(n)$ a referenční algoritmus má časovou náročnost $t_{ref}(n)$. Pro posouzení algoritmů budeme používat poměr:

$$\nu(n) = \frac{t_{ref}(n)}{t_{new}(n)} \quad (2.3)$$

To znamená, že z grafu funkce $\nu(n)$ jsme schopni posoudit chování nového algoritmu vůči referenčnímu algoritmu. V mnoha případech však nemáme k dispozici implementaci referenčního algoritmu, např. pokud chování referenčního algoritmu je pouze známo z odborné publikace. Otázkou je, jak postupovat v takovém případě?

Většinou asi budeme chtít ukázat, že nově navržený algoritmus je lepší, a tedy je „sympatická“ hodnota $\nu(n) > 1$. To je i důvod k výše uvedené formulaci $\nu(n)$, což v případě časového porovnání znamená vlastně urychlení.

Obecně se při porovnávání algoritmů postupuje tak, že za referenční algoritmus se vezme algoritmus, jehož chování je stabilní, tj. pro dané n jeho paměťová a časová náročnost nezávisí na vlastních datech, např. na geometrickém rozložení bodů apod. Toto většinou splňují algoritmy, které řeší daný problém „brutální silou“ (Brute Force - BF). U dříve zmíněného algoritmu testu, zda bod je uvnitř konvexního n-úhelníka, je to BF algoritmus se složitostí $O(N)$. Tím získáme chování nového algoritmu vůči algoritmu BF řešící daný problém. Algoritmus BF je nyní vlastně referenčním algoritmem. Pokud je obdobně i ohodnocena implementace, se kterou nový algoritmus porovnáváme, pak $\nu(n)$ je definován „inverzně“ (ale stále ukazuje na zrychlení nového algoritmu), a to:

$$\nu(n) = \frac{\nu(n)}{{}^1\nu(n)} = \frac{{}^1t_{BT}(n)/t_{new}(n)}{{}^1t_{BT}(n)/{}^1t_{new}(n)} = q \frac{{}^1t_{new}(n)}{t_{new}(n)} \quad (2.4)$$

kde: ${}^1\nu(n)$ je chování „konkurenčního“ algoritmu, vůči kterému nový algoritmus porovnáváme, $q = {}^1t_{BT}(n)/{}^1t_{new}(n)$ je faktor, který určuje vliv technických parametrů použitých výpočetních

systémů. Pokud je implementace algoritmu na stejných systémech, pak ideálně $q = 1$. Nicméně je otázkou, jak postupovat, pokud použité systémy jsou rozdílné, tj. $q \neq 1$. V tomto případě lze hodnotu q odhadnout na základě porovnání výpočetní výkonnosti z odpovídajících „benchmarků“ apod.

Poznámka

Při realizaci algoritmů je nanejvýš vhodné:

- vyhnout se dvojímu, resp. trojímu indexování, i když popis algoritmu tyto indexace používá, např. $T[i, j]$, resp. $T[i, j, k]$. Je nutné si uvědomit, že se indexy přepočítávají na adresu v lineární paměti, např. $addr = (i * N + j) * M + k$. Navíc se v mnoha případech indexy mění pouze o ± 1

- 1 • nepožívat extenzivně objekty – určitě bude neúnosné říci, že bod (x, y) je objekt a generovat
2 n objektů pro $n \in \langle 10^6, 10^{12} \rangle$, nebo implementovat matici hodnot jako matici objektů, kde
3 každý objekt má hodnotu atd.

4

5 **Poznámka**

6 Při testech se doporučuje volit hodnoty n podle mocninné řady R5, resp. R10 a pro několik dekád,
7 např. $n \in \langle 10^6, 10^{12} \rangle$. Výhodou je, že na ose pro n pak dostaváme rovnoměrné rozložení hodnot, což
8 může usnadnit grafické zobrazení závislostí.

R10	1	1.25	1.6	2	2.5	3.15	4	5	6.3	8
R5	1		1.6		2.5		4		6.3	

9 (http://en.wikipedia.org/wiki/Preferred_number)

10

11 Při experimentálním vyhodnocování složitosti je vhodné naměřená data proložit regresní křivkou
12 s popisem, např. s použitím metody nejmenších čtverců, viz např. kap.10.5 (Metoda nejmenších
13 čtverců).

14

15 Je tedy zřejmé, že „poctivé“ porovnání algoritmů není až tak jednoduchou záležitostí, jak by se na
16 první pohled mohlo zdát.

17

18

3. Matematický aparát počítačové grafiky

Počítačová grafika v užším slova smyslu je „generativní“, tj. na základě algoritmické a matematické specifikace se generuje příslušný grafický výstup, ne nezbytně jen 2D obraz. Tento proces se často označuje výrazem „rendrování“ a příslušný modul je označován jako „Renderer“.

V následujícím textu je předpokládána základní znalost lineární algebry, zejména pak základní operace s vektory a maticemi. Je nutné zdůraznit, že předpokládané znalosti matematiky jsou v zásadě na středoškolské úrovni se znalostí vektorové a maticové notace. Kompletnější souhrn operací je pak uveden v přílohách. V dalším textu budeme používat následující notace:

- a – skalární hodnota
- $\mathbf{a} = [a_1, \dots, a_n]^T$ – sloupcový vektor (tučné malé písmeno)
- \mathbf{A} – matice (tučné velké písmeno)

Uveďme alespoň základní operace (předpokládejme správné rozměry, existenční podmínky apod.):

- *skalární součin* (inner product) $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v}$ výsledkem je skalární hodnota

$$\|\mathbf{u}\| = \sqrt{u_x^2 + u_y^2 + u_z^2} \quad \mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos \varphi \quad (3.1)$$

- *vektorový součin* (cross product) $\mathbf{u} \times \mathbf{v}$ - výsledek se označuje většinou vektorem, ale přesně vzato je to orientovaná plocha a v E^3 je to bi-vektor. Vektorový součin je určen:

$$\mathbf{u} \times \mathbf{v} = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix} \quad \text{resp.} \quad \mathbf{u} \times \mathbf{v} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{T} \mathbf{v} \quad (3.2)$$

a dále platí:

$$\|\mathbf{u} \times \mathbf{v}\| = \|\mathbf{u}\| \|\mathbf{v}\| \sin \varphi \quad (3.3)$$

- *tenzorový součin* (tensor product):

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_x v_x & u_x v_y & u_x v_z \\ u_y v_x & u_y v_y & u_y v_z \\ u_z v_x & u_z v_y & u_z v_z \end{bmatrix} \quad (3.4)$$

Jde tedy vlastně o operaci sloupec krát řádek, kde výsledek je matice $\mathbf{Q} = \mathbf{u} \mathbf{v}^T$.

Pro tenzorový součin pak platí vztah:

$$(\mathbf{u} \otimes \mathbf{v}) \hat{\mathbf{n}} = (\mathbf{v} \cdot \hat{\mathbf{n}}) \mathbf{u} \quad (3.5)$$

který v dalším použijeme.

Dále platí:

$$\mathbf{u} \otimes \mathbf{v} = (\mathbf{v} \otimes \mathbf{u})^T \quad \mathbf{u} \otimes (a\mathbf{v} + b\mathbf{w}) = a(\mathbf{u} \otimes \mathbf{v}) + b(\mathbf{u} \otimes \mathbf{w}) \quad (3.6)$$

Lze dokázat, že pro libovolné vektory \mathbf{c}, \mathbf{d} a \mathbf{e} platí:

$$(\mathbf{c} \cdot \mathbf{d})\mathbf{e} = (\mathbf{e} \otimes \mathbf{c})\mathbf{d} = (\mathbf{e} \otimes \mathbf{d})\mathbf{c} \quad (3.7)$$

- *transpozice vektoru* \mathbf{a}^T , resp. matice \mathbf{A}^T
- *násobení matic* $\mathbf{C} = \mathbf{AB}$
- *inverze matic* $\mathbf{C} = \mathbf{A}^{-1}$, ($\det(\mathbf{A}) \neq 0$)
- *řešení soustav lineárních rovnic* $\mathbf{Ax} = \mathbf{b}$, resp. $\mathbf{Ax} = \mathbf{0}$

Upozornění

Operace s maticemi *nejsou obecně komutativní*, tj.

- pro součin matic $\mathbf{AB} \neq \mathbf{BA}$
- pro transpozici $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- pro inverzi $(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$

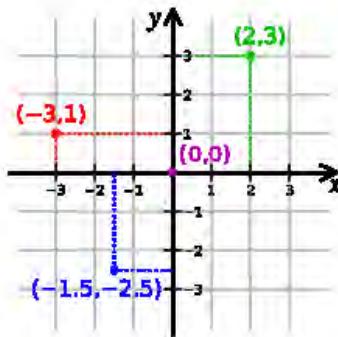
3.1. Souřadné systémy v počítačové grafice

V následujícím výkladu uvedeme nejběžnější souřadné systémy používané v praxi, a to jak v E^2 , tak i v E^3 . Je nutné zdůraznit, že souřadný systém je primárně dán aplikační oblastí. Pro aplikace v oblasti zpracování dat z pozemních radarů rotačního typu půjde zřejmě o válcové souřadnice, např. pro řízení letového provozu apod.

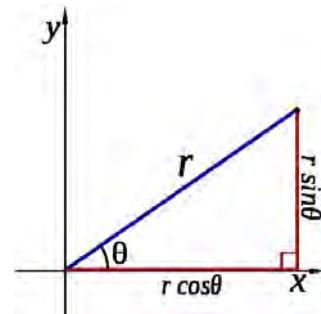
6

3.1.1. Souřadné systémy v E^2

Souřadné systémy v E^2 , tedy v rovině, jsou většinou charakterizovány proměnnými x, y v Eukleidovském prostoru, v parametrickém prostoru se pak většinou používá t , resp. u, v



Obr.3.1: Kartézský souřadný systém



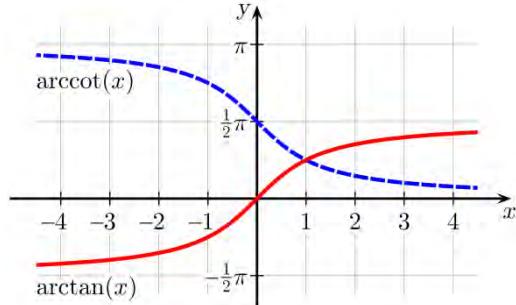
Obr.3.2: Polární souřadný systém

10

Vzájemný převod polárních a kartézských souřadnic je dán rovnicemi:

$$\begin{aligned} x &= r \cos \theta \\ y &= r \sin \theta \end{aligned}$$

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \arctg(\frac{y}{x}) \end{aligned}$$



Obr.3.3: Cyklometrické funkce

Z uvedených vztahů vidíme, že převod není úplně jednoduchý. Funkce $\arctg(x)$ je cyklometrická a výsledkem funkce není úhel $\varphi \in (0, 2\pi)$, neboť funkce $\tan x$ je definována na intervalu $(-\pi/2, \pi/2)$.

Navíc operace y/x vede k numerické nestabilitě, pokud $|x| \rightarrow 0$.

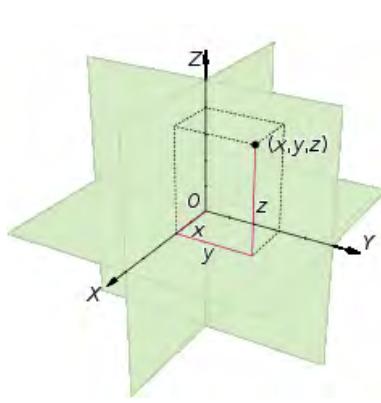
15

Je tedy zřejmě, že i takto jednoduchým vztahům je nutné věnovat náležitou pozornost také z hlediska numerické stability, podrobněji viz kap.3.6 (Numerická reprezentace a stabilita výpočtů).

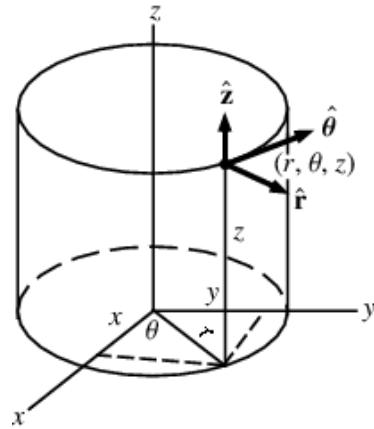
18

1 **3.1.2. Souřadné systémy v E^3**

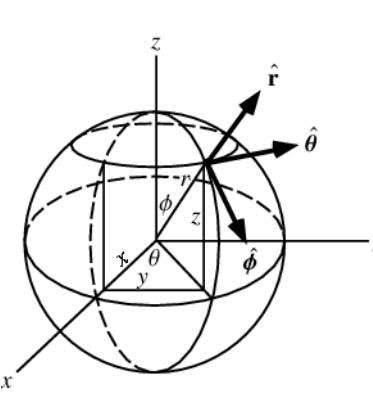
2 Nejčastěji používané souřadné systémy v E^3 jsou většinou:



Obr.3.4: Kartézský systém



Obr.3.5: Válcový systém



Obr.3.6: Sférický systém

3

4 Vzájemný převod do/z je určen rovnicemi:

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$z = z$$

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \arctg(y/x)$$

$$z = z$$

$$x = r \cos \theta \sin \phi$$

$$y = r \sin \theta \sin \phi$$

$$z = r \cos \phi$$

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arctg(y/x)$$

$$\phi = \arccos(z/r)$$

Válcový souřadný systém

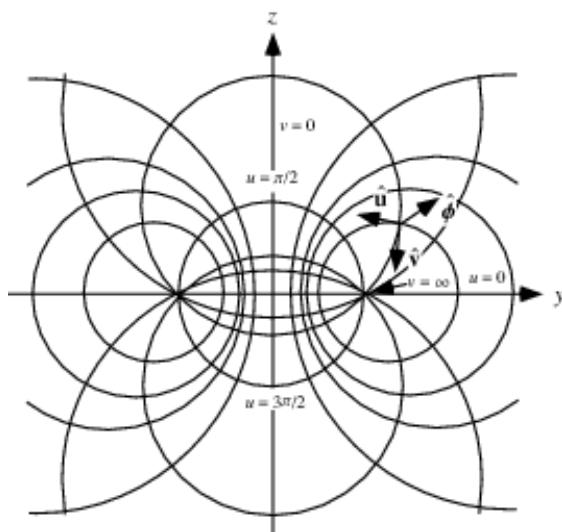
Sférický souřadný systém

5

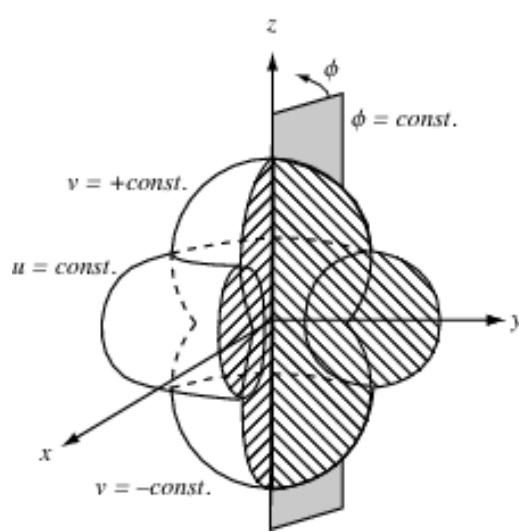
6 Výše uvedené souřadné systémy jsou běžně známý, používány a lze je považovat za standardní. Jsou
7 zajisté součástí standardního středoškolského vzdělání, zde jsou uvedeny jen pro úplnost.

8

9 Vedle výše uvedených souřadných systémů existuje celá řada dalších souřadných systémů
10 používaných v praxi. Pro názornost uvedeme jen toroidální souřadný systém:



Obr.3.7: Ortogonální souřadné systémy



11 Přehled mnoha dalších souřadných systémů lze nalézt např. na

12 <http://mathworld.wolfram.com/topics/CoordinateGeometry.html>

1 **3.1.3. Datové typy**

2 Používané datové typy vycházejí z datových standardů, dnes převážně ze standardu IEEE 578-2008.
 3 Tyto datové typy jsou podporovány současnými programovými prostředky, a to buď přímo pomocí
 4 hardwarové podpory, nebo nepřímo pomocí softwarových knihoven.

	Name	Digits	E min	E max
B 16	Half	10+1	-14	15
B 32	Single	23+1	-126	127
B 64	Double	52+1	-1022	1023
B 128	Quad	112+1	-16382	16383

5 IEEE 578-2008 Floating point reprezentace

6

7 Zde je nutné zdůraznit, že součástí standardu IEEE 578-2008 je definice datových typů o základu 10
 8 (http://en.wikipedia.org/wiki/IEEE_floating_point).

Name	Base	Digits	E min	E max	Decimal digits	Decimal E max
decimal32	10	7	-95	+96	7	96
decimal64	10	16	-383	+384	16	384
decimal128	10	34	-6143	+6144	34	6144

9

10 Pokud budeme více konkrétní, pak datové typy, které jsou obvykle k dispozici, jsou následující:
 11 **char, byte**
 12 **integer, long integer**
 13 **float, double, extended**
 14 **complex**, tj. $c = a + ib$ reprezentace komplexních čísel. Tento datový typ je k dispozici např.
 15 v jazyku FORTRAN
 16 **decimal** - reprezentace dekadických hodnot používaná zejména u ekonomických výpočtů
 17
 18 Při reprezentaci hodnot a manipulaci s nimi musíme mít na paměti zejména spolehlivost (reliability)
 19 a robustnost (robustness), a to jak pro výpočty, tak i pro vizualizaci dat.
 20
 21

3.2. Homogenní souřadnice a jejich geometrická interpretace

Mezi základní geometrické transformace řadíme známé operace, jako je posun (translation), rotace (rotation), změna měřítka (scaling), zkosení (shearing) jsou známé operace, a to i včetně perspektivní projekce (perspective projection), která se obvykle používá.

V následujícím textu zavedeme pojem projektivního rozšíření Eukleidovského prostoru (projektivní prostor) a homogenní souřadnice. V dalším výkladu se budeme snažit dodržovat značení pro:

- souřadnice v Eukleidovském prostoru $\mathbf{X} = (X, Y)^T$, tedy velká písmena
- souřadnice v projektivním prostoru $\mathbf{x} = [x, y: w]^T$, tedy malá písmena.

3.2.1. Operace v Eukleidovském prostoru

Je známo, že operace posunu bodu (X, Y) o vzdálenost (A, B) , je dána:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} A \\ B \end{bmatrix} \quad (3.9)$$

a operace rotace bodu (X, Y) okolo počátku je určena:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \quad (3.10)$$

Ostatní geometrické operace pro body lze definovat obdobně. Nicméně je zřejmé, že bude nutné kombinovat operace posunu a rotace, dělat též operace inverzní atd., což při Eukleidovské reprezentaci jednoduše nejde. V případě projektivního rozšíření Eukleidovského prostoru lze ukázat, že všechny základní geometrické transformace lze realizovat pomocí násobení matic. Toto je velmi výhodné, nejen z hlediska jednoduché reprezentace inverzních operací, neboť:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad (3.11)$$

ale i z důvodu hardwarové podpory geometrických transformací, neboť jsou všechny převedeny na jednotný tvar:

$$\mathbf{x}' = \mathbf{Ax} \quad (3.12)$$

kde $\mathbf{x} = [x, y: w]^T$, resp. $\mathbf{x}' = [x', y': w']^T$ jsou souřadnice v homogenních souřadnicích bodu X , resp. X' . Hodnota w se nazývá homogenní složka vektoru \mathbf{x} a je bezrozměrná, na rozdíl od hodnot x, y , které mají fyzikální rozměr, např. [m]. Proto hodnoty w budou oddělovány znakem ":".

Zde je vhodné krátce uvést projektivní rozšíření Eukleidovského prostoru, tedy vlastně projektivní rozšíření kartézského souřadného systému, kde jsou hodnoty reprezentovány v homogenních souřadnicích. Jejich vzájemné převody jsou uvedeny v Tab.QQQ a z hlediska matematického popisu jde tedy o poměrně jednoduchou záležitost.

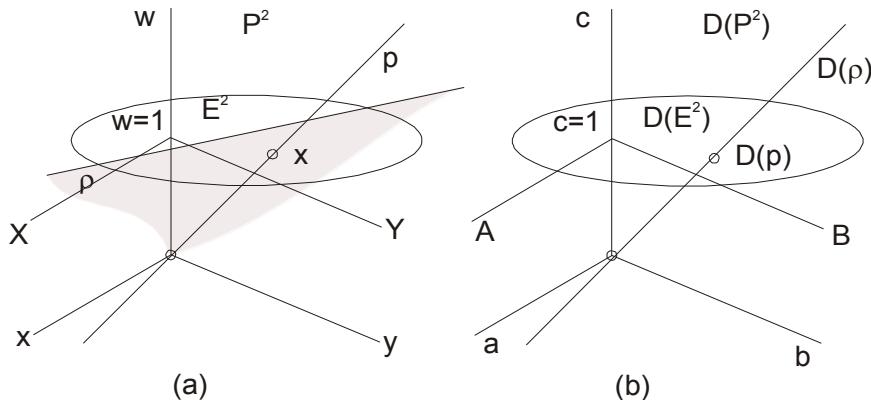
	Dimenze = 2	Dimenze = 3
Eukleidovský prostor	$\mathbf{X} = (X, Y) \in E^2$	$\mathbf{X} = (X, Y, Z) \in E^3$
Projektivní prostor	$\mathbf{x} = [x, y: w]^T \quad w \neq 0$	$\mathbf{x} = [x, y, z: w]^T \quad w \neq 0$
Vzájemný převod	$X = \frac{x}{w} \quad Y = \frac{y}{w} \quad w \neq 0$	$X = \frac{x}{w} \quad Y = \frac{y}{w} \quad Z = \frac{z}{w}$
Alternativní popisy	$\tilde{\mathbf{x}} = [w: x, y, z]^T = [a_0: a_1, \dots, a_n]^T \quad \text{resp. } \mathbf{x} = [a_0: a_1, \dots, a_n]^T$	

Tab.QQQ: Vzájemné převody základních souřadních systémů

Alternativní notace, tj. když homogenní složka je na první pozici, se většinou používá v matematickém textu:

$$\mathbf{x} = [w: x, y]^T \quad \text{obecně pak pro } E^n \quad \mathbf{x} = [x_0: x_1, \dots, x_n]^T \quad (3.13)$$

- 1 Výhodou tohoto alternativního zápisu je, že homogenní souřadnice w je vždy na první pozici nehledě
 2 na dimenzionalitu řešeného problému.
 3 Podívejme se však, jaká je geometrická interpretace v případě E^2 .



Obr.3.8: Projektivní rozšíření Eukleidovského prostoru a duální reprezentace

- 4
 5 Eukleidovský prostor E^2 je vlastně rovinou v projektivním prostoru P^2 pro $w = 1$. Z Obr.3.8 vidíme,
 6 že bod X je vlastně reprezentován přímkou p v projektivním prostoru P^2 a všechny body kromě
 7 počátku ležící na přímce p jsou vlastně jednoparametrickou reprezentací tohoto bodu.
 8 V homogenních souřadnicích obecně nemusí být hodnota $w = 1$.
 9 Pokud $w = 0$, pak jde o bod v nekonečnu, tzv. ideální bod (an ideal point nebo point in infinity), tedy
 10 o směr. Vidíme tedy, že pomocí projektivní reprezentace můžeme reprezentovat též body
 11 v nekonečnu.

12

13 Ukažme nyní výhodnost projektivní reprezentace na případě operací posuvu a rotace.

14

15 Operace posuvu

16 Pokud nyní použijeme projektivní reprezentaci pro operaci posuvu bodů, pak dostáváme:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x + Aw \\ y + Bw \\ w \end{bmatrix} \triangleq \begin{bmatrix} x/w + A \\ y/w + B \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + B \\ 1 \end{bmatrix} \quad (3.14)$$

17 kde \triangleq značí projektivní ekvivalence.

18 Je tedy zřejmé, že se podařilo převést operaci posuvu bodu na operaci maticového násobení, tj.
 19 operaci stejného typu jako je rotace a změna měřítka.

20

21 Operace rotace

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (3.15)$$

22 Jednotlivé geometrické operace v projektivním prostoru budou vysvětleny později, viz kap.4
 23 (Základní geometrické transformace).

24

25 Je nutné upozornit, že transformace pro implicitní vyjádření přímek, tj. koeficientů $[a, b: c]^T$, resp.
 26 rovin, tj. koeficientů $[a, b, c: d]^T$, nejsou totožné s transformacemi pro body.

27

28

3.3. Dualita a její aplikace

Použití projektivního rozšíření Eukleidovského prostoru je výhodné nejen pro „elegantní“ reprezentaci geometrických transformací, ale navíc poskytuje některé nové možnosti výpočtu geometrických entit, např. přímek nebo rovin, manipulace s nimi a též i elegantní řešení některých vybraných geometrických úloh, např. průsečík přímek v E^2 , průsečík rovin v E^3 apod.

3.3.1. Princip duality

Jednou ze základních vlastností reprezentace v projektivním prostoru je existence principu duality. Princip duality v P^2 říká, že:

- jakýkoliv teorém v E^2 zůstává platný, pokud zaměníme slova „bod“ a „přímka“, „leží na“ a „prochází“, „spojuje“ a „protíná“ apod.
- pokud byl dokázán jakýkoliv teorém, pak duální teorém dostaneme jak výše uvedeno.

To znamená, že **jedním výpočetním postupem můžeme řešit jak primární úlohu, tak i úlohu duální.**

V dalším výkladu se nebudeme zabývat teoretickými vlastnostmi principu duality, ale tím, jak tento princip můžeme s výhodou využít pro řešení úloh počítačové grafiky.

Pro jednoduchost uvažme velmi jednoduchý případ v E^2 , a to přímku p , která je dána v implicitní formě:

$$aX + bY + c = 0 \quad (3.16)$$

Bez újmy na obecnosti můžeme rovnici vynásobit $w \neq 0$ a dostaneme:

$$awX + bwY + cw = 0 \quad (3.17)$$

a protože $x = wX$ a $y = wY$ můžeme psát:

$$ax + by + cw = 0 \quad (3.18)$$

Ve vektorové notaci pak:

$$\mathbf{p}^T \mathbf{x} = 0 \quad (3.19)$$

kde: $\mathbf{p} = [a, b; c]^T$, $\mathbf{x} = [x, y; w]^T = [wX, wY; w]^T$.

Přímka p v E^2 je vlastně rovinou ρ v projektivním prostoru, přičemž bod $\mathbf{x} = [0, 0; 0]^T$ je vyloučen, viz Obr.3.8.

Z notace $\mathbf{p}^T \mathbf{x} = 0$ nelze určit význam jednotlivých symbolů, tj. určit, zda \mathbf{p} reprezentuje přímku a \mathbf{x} reprezentuje bod nebo naopak v případě P^2 . To znamená, že bod a přímka v E^2 jsou duální pojmy.

V případě P^3 lze ukázat, že bod a rovina jsou pojmy duální. Pochopitelně vzniká otázka, jak vlastně vypadá duální souřadný systém? Jeho geometrická reprezentace viz Obr.3.8.

Je nutné upozornit, že:

- duální reprezentací se v principu nemění výpočetní složitost
- dualita umožňuje převést problém do duální reprezentace, následně se duální problém vyřeší a výsledek se pak převede zpět z duální reprezentace do reprezentace původní.

Použití duality umožňuje se na daný problém „podívat z jiné strany“, což může vést k inovativnímu způsobu řešení daného problému. Aplikace principu duality však nevede ke snížení algoritmické složitosti jako takové.

1 Příklad duálních primitiv a operátorů:

	Primitivum	Duální primitivum
P^2	bod přímka	přímka bod
P^3	bod rovina	rovina bod
	Operátor	Duální operátor
	sjednocení průsečík	průsečík sjednocení

2 Důležitým výsledkem je, že jednou programovou sekvencí můžeme řešit jak primární, tak i duální úlohu současně.

4 3.3.2. Vektorový součin a řešení soustav lineárních rovnic

5 Mnoho geometrických úloh vede k řešení soustavy lineárních rovnic, ať už s pravou nenulovou stranou, tj. $\mathbf{Ax} = \mathbf{b}$, nebo s nulovou pravou stranou, tj. $\mathbf{Ax} = \mathbf{0}$. K řešení soustavy $\mathbf{Ax} = \mathbf{b}$ se používají různé metody, přímé nebo iterační. Soustavy $\mathbf{Ax} = \mathbf{0}$ většinou představují trochu problém, neboť pokud existuje netriviální řešení, pak řešení je alespoň jedno-parametrické. V každém případě většinou prezentované postupy nejsou vhodné např. pro použití na GPU. V následujícím výkladu bude ukázána aplikace vektorového součinu pro řešení soustavy lineárních rovnic.

11

12 Vektorový součin

13 Vektorový součin dvou vektorů v E^3 je definován:

$$\mathbf{X}_1 \times \mathbf{X}_2 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \end{bmatrix} \quad (3.20)$$

14 kde: $\mathbf{i} = [1,0,0]^T$ $\mathbf{j} = [0,1,0]^T$ $\mathbf{k} = [0,0,1]^T$.

15 Vektorový součin lze vyjádřit i jako násobení matice vektorem, a to:

$$\mathbf{X}_1 \times \mathbf{X}_2 = \begin{bmatrix} 0 & -Z_1 & Y_1 \\ Z_1 & 0 & -X_1 \\ -Y_1 & X_1 & 0 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{T}\mathbf{X}_2 \quad (3.21)$$

16 Podívejme se nyní, jak lze použít vektorový součin v projektivním prostoru.

17

18 Při použití vektorového součinu v projektivním prostoru P^2 dostáváme obdobné schéma, ale vektory \mathbf{x}_1 a \mathbf{x}_2 mají jiný význam, neboť reprezentují vlastně Eukleidovskou dvourozměrnou entitu. Nyní:

$$\mathbf{x}_1 = [x_1, y_1 : w_1]^T \qquad \mathbf{x}_2 = [x_2, y_2 : w_2]^T$$

20 Vektorový součin v P^2 je definován:

$$\mathbf{x}_1 \times \mathbf{x}_2 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \qquad \mathbf{x}_1 \times \mathbf{x}_2 = \begin{bmatrix} 0 & -w_1 & y_1 \\ w_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ w_2 \end{bmatrix} = \mathbf{T}\mathbf{x}_2 \quad (3.22)$$

21

22 kde: $\mathbf{i} = [1,0:0]^T$ $\mathbf{j} = [0,1:0]^T$ $\mathbf{k} = [0,0:1]^T$. Vektorový součin lze tedy opět vyjádřit i jako násobení matice vektorem.

24

25 Je nutné upozornit, že nyní používáme reprezentaci v homogenních souřadnicích
26 a jeho geometrický význam je odlišný.

1 **Průsečík dvou přímek**

2 Uvažme dvě přímky, které jsou určeny vektory:

$$\mathbf{p}_1 = [a_1, b_1 : c_1]^T$$

$$\mathbf{p}_2 = [a_2, b_2 : c_2]^T$$

3 Jejich průsečík je dán řešením soustavy rovnic:

$$a_1 X + b_1 Y + c_1 = 0$$

$$a_2 X + b_2 Y + c_2 = 0 \quad (3.23)$$

4 tedy soustavy rovnic tvaru $\mathbf{Ax} = \mathbf{b}$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix} \quad (3.24)$$

5 Nyní se obvykle použije formule pro výpočet průsečíku $\mathbf{X} = (X, Y)$:

$$X = \frac{\text{Det}_X}{\text{Det}} = \frac{\det \begin{bmatrix} q_1 & b_1 \\ q_2 & b_2 \end{bmatrix}}{\det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}} \quad Y = \frac{\text{Det}_Y}{\text{Det}} = \frac{\det \begin{bmatrix} a_1 & q_1 \\ a_2 & q_2 \end{bmatrix}}{\det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}} \quad (3.25)$$

6 Programátor se nyní musí rozhodnout, zda a za jakých podmínek jde již o singulární řešení, tj. přímky
7 jsou „již rovnoběžné“, a kdy ještě ne (jde vlastně o koncept řešení i pro složitější soustavy rovnic).

8

9 Takže programátor „*nadaný intuici*“ rozhodne, kdy hodnota Det je již příliš malá a způsobila by
10 přetečení (overflow) v pohyblivé řádové čárce a obvykle použije sekvenci:

11 ***if*** $\text{abs}(\text{det}(\dots)) \leq \text{eps}$ ***then*** ,

12 což je primárně zdroj nestability výpočtu.

13 Zkusme se nyní podívat na problém z druhé strany.

14

15 **Věta**

16 Nechť jsou dány dvě přímky \mathbf{p}_1 a \mathbf{p}_2 . Pak souřadnice bodu \mathbf{x} v projektivním prostoru, který je
17 průsečíkem těchto dvou přímek, jsou určeny vektorovým součinem homogenních souřadnic těchto
18 dvou přímek, a to:

$$\mathbf{x} = \mathbf{p}_1 \times \mathbf{p}_2 \quad \mathbf{x} = [x, y: w]^T \quad (3.26)$$

19 přičemž

$$\mathbf{p}_1 = [a_1, b_1 : c_1]^T$$

$$\mathbf{p}_2 = [a_2, b_2 : c_2]^T$$

20

21 **Důkaz**

22 Hledáme vlastně řešení následujících dvou rovnic

$$\mathbf{x}^T \mathbf{p}_1 = 0 \quad \mathbf{x}^T \mathbf{p}_2 = 0 \quad (3.27)$$

23 **Poznámka:** obvykle přímka v implicitní formě je dána jako $aX + bY = q$ místo $aX + bY + c = 0$,
24 nebo v explicitní formě $Y = kX + q$.

25

26 Na výše uvedeném příkladě bylo ukázáno, že k výpočtu průsečíku dvou přímek:

- 27 • lze použít vektorový součin
- 28 • není zapotřebí operace dělení, neboť se hodnota jmenovatele v dělení vlastně „uloží“ do
29 homogenní složky w souřadnice průsečíku \mathbf{x}
- 30 • pokud jsou přímky téměř rovnoběžné, pak hodnota $w \rightarrow 0$.

31 Zde je vhodné připomenout, že z důvodů limitované přesnosti výpočtů v pohyblivé řádové čárce, jsou
32 některé testy nesmyslné, např. zda 3 body leží na přímce, zda 4 body leží na rovině, zda se 2 přímky
33 v prostoru protínají, apod.

34 *i když by se z čistě matematického hlediska měly!*

35 Nyní se podívejme na řešení obdobné úlohy, a to určení přímky v E^2 , která je určena dvěma body.

1 **Přímka daná dvěma body**

2 Nechť jsou dány dva body X_1 a X_2 a chceme určit koeficienty přímky p , která je určena těmito body.

3 To znamená, že musíme určit 3 hodnoty, a to a, b, c ze dvou hodnot X_1, X_2 , tj. musíme řešit rovnice:

$$aX_1 + bY_1 + c = 0 \quad aX_2 + bY_2 + c = 0 \quad (3.28)$$

4 Dostáváme tedy homogenní soustavu rovnic, tj. s nulovou pravou stranou. Je zřejmé, že jde o jednoparametrické řešení (pokud nejde o singulární případ, tj. dva body totožné). V maticové formě pak:

$$\begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad Ax = \mathbf{0} \quad (3.29)$$

7 Jak řešit danou soustavu rovnic? Toto je obvykle problém a programátoři se obvykle snaží zvolit jeden parametr, např. $c = 1$, ale co když přímka prochází počátkem? Nebo $a = 1$ nebo $b = 1$. Také je možné řešení s použitím dodatečné podmínky $a + b = 1$ a pak:

$$\begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad Ax = b \quad (3.30)$$

10 Toto je ale principiálně špatný přístup.

11 **Je tedy otázkou, jak daný problém řešit?**

12 **Elegantní a jednoduché řešení**

13 Z principu duality víme, že přímka v E^2 je duální bodu a naopak. Díky principu duality tedy dostáváme řešení:

Průsečík	\Leftrightarrow dualita \Rightarrow	Spojení
$x = p_1 \times p_2$	\Leftrightarrow dualita \Rightarrow	$p = x_1 \times x_2$
$Ax = b$	\Leftrightarrow ale proč různé? \Rightarrow	$Ax = \mathbf{0}$

15 Jinými slovy: proč pro duální problém musím řešit rozdílným způsobem?

16

17 **Věta** (plynoucí z principu duality)

18 Nechť jsou dány dva body x_1 and x_2 v projektivním prostoru P^2 . Pak koeficienty přímky p , která je určena těmito body, jsou určeny vektorovým součinem takto:

$$p = x_1 \times x_2 = [a, b: c]^T \quad (3.31)$$

20 **Důkaz**

21 Nechť přímka p je v projektivním prostoru P^2 určena jako:

$$ax + by + cw = 0$$

22 pak rovnice přímky musí platit pro oba dva body, tj. musí platit rovnice:

$$p^T x_1 = 0 \quad p^T x_2 = 0 \quad (3.32)$$

23 kde: $p = [a, b: c]^T$

24

25 **Poznámka**

26 Koeficient c „reprezentuje“ vzdálenost (je to vlastně násobek vzdálenosti) přímky od počátku souřadného systému, zatímco koeficienty a, b reprezentují normálu přímky.

28

29 To znamená, že jakýkoliv bod x , který leží na přímce p , musí vyhovovat oběma rovnicím výše a rovnici $p^T x = 0$, a tedy vektor p je definován:

$$p = x_1 \times x_2 = \det \begin{bmatrix} i & j & k \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \quad (3.33)$$

31

1 Pak pro bod x můžeme psát:

$$(x_1 \times x_2)^T x = 0 \quad \det \begin{vmatrix} x & y & w \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{vmatrix} = 0 \quad (3.34)$$

2 Poznamenejme, že skalární a vektorový součin jsou instrukcemi v Cg/HLSL na GPU. Vyhodnocením determinantu:

$$\det \begin{vmatrix} a & b & c \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{vmatrix} = 0 \quad (3.35)$$

4 pak dostáváme rovnici přímky p takto:

$$a = \det \begin{vmatrix} y_1 & w_1 \\ y_2 & w_2 \end{vmatrix} \quad b = -\det \begin{vmatrix} x_1 & w_1 \\ x_2 & w_2 \end{vmatrix} \quad c = \det \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \quad (3.36)$$

5

6 Obdobně jako v P^2 je princip duality platný i v prostoru P^3 , ale je nutné na definovat rozšíření
7 vektorového součinu.

8

9 3.3.3. Aplikace duality v prostoru E^3

10 V prostoru P^3 jsou duální primitiva bod a rovina. Rovina je určena třemi body a bod je průsečíkem tří
11 rovin.

12 Parametry roviny $\rho = [a, b, c: d]^T$ určené body x_1, x_2, x_3 a souřadnice průsečíku $x = [x, y, z: w]^T$
13 určeného rovinami ρ_1, ρ_2, ρ_3 lze určit takto:

$$\begin{aligned} \rho &= x_1 \times x_2 \times x_3 & x &= \rho_1 \times \rho_2 \times \rho_3 \\ &= \begin{vmatrix} i & j & k & l \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix} & &= \begin{vmatrix} i & j & k & l \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{vmatrix} \end{aligned} \quad (3.37)$$

14 Je tedy zřejmé, že výpočet je opět velmi jednoduchý a není zapotřebí žádné dělení ani žádná volba
15 hodnoty nějakého parametru apod.

16

17 V předchozím bylo ukázáno, že řešení obou typů soustav rovnic, tj.

$$Ax = b \quad Ax = 0 \quad (3.38)$$

18 je speciálním případem při použití vektorového součinu.

19

20 Závěr

- 21 • vektorový součin je ekvivalentní řešení obou typů soustav lineárních rovnic
- 22 • pro řešení soustavy rovnic není zapotřebí operace dělení, pokud výsledek může být
23 ponechán v projektivní reprezentaci

24

25

26 Je důležité poznamenat, že souřadnice bodů x , které mají obecně hodnotu homogenní složky $w \neq 1$,
27 není při výpočtech zapotřebí převádět do Eukleidovského prostoru, čímž se ušetří 2, resp. 3 operace
28 dělení na 1 bod v E^2 , resp. v E^3 .

29

1 **3.3.4. Vzdálenost v projektivním prostoru**

2 Geometrické úlohy jsou úzce spojeny s pojmem vzdálenosti a jejím měřením. V Eukleidovské
3 geometrii je vzdálenost určena:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad , \text{ resp.} \quad d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad (3.39)$$

5 Jedním z častých argumentů proti používání projektivní reprezentace je, že není jednoduše
6 definována metrika. Toto sice eliminuje konformní geometrie, kterou se však nebudeme zabývat.

8 V případě projektivní reprezentace, vzdálenost dvou bodů je určena vztahem:

$$dist = \sqrt{\xi^2 + \eta^2} / (w_1 w_2) \quad (3.40)$$

9 kde: $\xi = w_1 x_2 - w_2 x_1$ $\eta = w_1 y_2 - w_2 y_1$

11 Vzdálenost bodu x_0 od přímky v P^2 je určena vztahem:

$$dist = \frac{\mathbf{a}^T \mathbf{x}_0}{w_0 \sqrt{a^2 + b^2}} \quad (3.41)$$

12 kde: $\mathbf{x}_0 = [x_0, y_0 : w_0]^T$ $\mathbf{a} = [a, b : c]^T$

14 Extenze do E^3/P^3 je jednoduchá a vzdálenost bodu x_0 od roviny je dána:

$$dist = \frac{\mathbf{a}^T \mathbf{x}_0}{w_0 \sqrt{a^2 + b^2 + c^2}} \quad (3.42)$$

15 kde: $\mathbf{x}_0 = [x_0, y_0, z_0 : w_0]^T$ $\mathbf{a} = [a, b, c : d]^T$.

17 V mnoha případech vlastně nepotřebujeme vzdálenost určující, který bod je blíže. Proto nám ale stačí
18 monotónně rostoucí funkce vzdálenosti bodu x_i od přímky p , např. kvadrát vzdálenosti $dist_i^2$. Pak
19 pro případ E^2 dostaváme:

$$dist_i^2 = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2 (a^2 + b^2)} = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2 \mathbf{n}^T \mathbf{n}} \quad (3.43)$$

20 kde: $\mathbf{a} = [a, b : c]^T = [\mathbf{n}^T : c]^T$ a normálový vektor $\mathbf{n} = [a, b]^T$ není normalizován. Tím můžeme
21 podstatně snížit nároky na výpočetní výkon.

23 Pokud porovnáváme vzdálenosti více bodů x_i , $i = 1, \dots, n$ od stejné dané přímky p , můžeme použít
24 pro porovnání "pseudodistance":

$$(pseudo_dist_i)^2 = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2} \quad (3.44)$$

25 Analogicky pro případ roviny ρ v E^3 :

$$dist_i^2 = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2 (a^2 + b^2 + c^2)} = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2 \mathbf{n}^T \mathbf{n}} \quad (pseudo_dist_i)^2 = \frac{(\mathbf{a}^T \mathbf{x}_i)^2}{w_i^2} \quad (3.45)$$

26 kde: $\mathbf{a} = [a, b, c : d]^T = [\mathbf{n}^T : d]^T$ a normálový vektor $\mathbf{n} = [a, b, c]^T$ není normalizován.

27

3.4. Kvaterniony

Pro reprezentaci souřadnic bodů, resp. vektorů v rovině byla dosud používána reprezentace

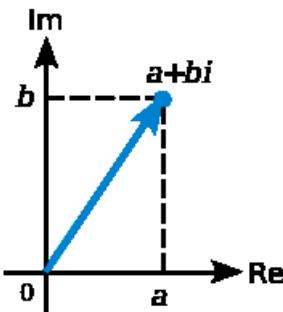
$\mathbf{x} = [x, y]^T$, resp. $\mathbf{s} = [s_x, s_y]^T$. Pro operace rotace pak byla používána maticová forma ve tvaru"

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.46)$$

Pro snadnější popis operací lze také použít notaci pro komplexní čísla. Je vhodné připomenout, že některé programovací jazyky mají standardní typ **complex** s odpovídajícími základními matematickými operacemi. Reprezentace pomocí komplexních čísel umožňuje elegantní řešení mnoha technických problémů a jejich výhodné použití lze nalézt v elektrotechnice, fyzice, statistice atd.

3.4.1. Komplexní čísla a jejich reprezentace

Pojem komplexního čísla byl pravděpodobně zaveden Gerolamo Cardano (1501-1576), který řešil problém výpočtu kubické rovnice. Komplexní číslo $\mathbf{z} = a + ib$ má vlastně dvě složky, a to reálnou složku a a imaginární složku b . Tyto složky se také někdy označují jako Re , resp. Im . Symbol i označuje imaginární složku komplexního čísla, nikoliv index, tj. $a = \text{Re}(\mathbf{z})$ a $b = \text{Im}(\mathbf{z})$.



Obr.3.9: Geometrická interpretace komplexního čísla

Pro komplexní čísla platí následující základní operace:

$$i^2 = -1 \quad (3.47)$$

neboť násobení i vlastně znamená rotaci o úhel $+\pi/2$

Komplexně sdruženým číslem k číslu $\mathbf{z} = a + ib$ je číslo $\bar{\mathbf{z}} = a - ib$.

Sčítání

$$(a + ib) \pm (c + id) = (a \pm c) + i(b \pm d) \quad (3.48)$$

Násobení

$$(a + ib) * (c + id) = (ac - bd) + i(bc + ad) \quad (3.49)$$

Dělení

$$\frac{(a + ib)}{(c + id)} = \frac{(a + ib)(c - id)}{(c + id)(c - id)} = \frac{(ac + bd) + i(bc - ad)}{c^2 + d^2} \quad (3.50)$$

a tedy:

$$\frac{1}{\mathbf{z}} = \frac{\bar{\mathbf{z}}}{\mathbf{z}\bar{\mathbf{z}}} \quad (3.51)$$

Absolutní hodnota r komplexního čísla je pak určena:

$$r^2 = \|\mathbf{z}\|^2 = \mathbf{z}\bar{\mathbf{z}} = a^2 + b^2 \quad (3.52)$$

Protože platí Eulerova rovnost:

$$e^{i\varphi} = \cos \varphi + i \sin \varphi \quad (3.53)$$

a označíme-li:

$$\mathbf{z} = x + iy \qquad \mathbf{z}' = x' + iy' \quad (3.54)$$

1 pak operaci rotace ve 2D o úhel φ můžeme přepsat do tvaru:

$$\mathbf{z}' = \mathbf{ze}^\varphi = (x \cos \varphi - y \sin \varphi) + i(x \sin \varphi + y \cos \varphi) \quad (3.55)$$

2
3 Bohužel takto jednoduché řešení pro třírozměrný případ není možné. Nicméně pro operace rotace
4 v třírozměrném prostoru je možné využít koncept kvaternionů.

5
6 **3.4.2. Kvaterniony**
7 V 19. stol. přišel s konceptem kvaternionů (quaternions) Sir William Rowan Hamilton (1805-1865),
8 kterému trvalo 15 let formulovat koncept 4D notace pro operace rotace v třírozměrném prostoru a
9 postulovat následující základní pravidla kvaternionů, jako vícerozměrného komplexního čísla, a to:

$$\begin{aligned} i^2 &= j^2 = k^2 = ijk = -1 \\ ij &= k & jk &= i & ki &= j \\ ji &= -k & kj &= -i & ik &= -j \end{aligned} \quad (3.56)$$

10
11 Nicméně mnoho matematiků mělo velkou nedvěru k imaginárním členům výrazu a koncept
12 kvaternionů nebyl ještě dlouhou dobu akceptován. Teprve Josiah Gibbs (1839-1903) přišel
13 s konceptem reprezentace komplexních členů kvaternionů, tj. $ib + jc + kd$, kde i, j, k označují
14 komplexní složky, jako 3D vektoru $bi + cj + dk$, kde i, j, k jsou jednotkové vektory Kartézského
15 souřadného systému. Jde tedy o zobrazení $R^4 \times R^4 \rightarrow R^4$.

16
17 V současné době se používají dvě notace kvaternionů, které se liší nepodstatným způsobem, a to:

$$\begin{aligned} \mathbf{q} &= [s, \mathbf{v}] = [s, xi + yj + zk] = \text{re}(\mathbf{q}) + \text{pu}(\mathbf{q}) \\ &\qquad \mathbf{q} = [s + \mathbf{v}] \end{aligned} \quad (3.57)$$

18 kde: s, x, y, z jsou reálná čísla, $\text{re}(\mathbf{q})$ značí reálnou složku kvaternionu a $\text{pu}(\mathbf{q})$ značí rýze komplexní
19 složku (*pure part*) kvaternionu. V následujícím budeme používat první notaci.

20
21 Kvaternion je tedy kombinací skalární hodnoty a 3D vektoru. Kvaterniony umožňují dobře popsat
22 operace rotace okolo osy, tj. např. kamery nebo 3D objektu.

24 **3.4.3. Základní operace s kvaterniony**

25 Pro pochopení, jak je možné kvaterniony použít, je nutné znát alespoň základní operace
26 s kvaterniony. Uvažme dva kvaterniony \mathbf{q}_1 a \mathbf{q}_2 :

$$\begin{aligned} \mathbf{q}_1 &= [s_1, \mathbf{v}_1] = [s_1, x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}] \\ \mathbf{q}_2 &= [s_2, \mathbf{v}_2] = [s_2, x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}] \end{aligned} \quad (3.58)$$

27 **Sčítání a odečítání kvaternionů**

28 Vzhledem k tomu, že kvaternion je vlastně definován jako vektor ve 4D, je sčítání a odečítání
29 kvaternionů definováno takto:

$$\mathbf{q}_1 \pm \mathbf{q}_2 = [s_1 \pm s_2, \mathbf{v}_1 \pm \mathbf{v}_2] = [(s_1 \pm s_2), (x_1 \pm x_2)\mathbf{i} + (y_1 \pm y_2)\mathbf{j} + (z_1 \pm z_2)\mathbf{k}] \quad (3.59)$$

31
32 **Násobení kvaternionů**
33 Pro násobení kvaternionů je nutné respektovat následující pravidla:

$$\begin{aligned} i^2 &= j^2 = k^2 = ijk = -1 \\ ij &= k & jk &= i & ki &= j \\ ji &= -k & kj &= -i & ik &= -j \end{aligned} \quad (3.60)$$

34
35 Pak součin dvou kvaternionů \mathbf{q}_1 a \mathbf{q}_2 je určen:

$$\mathbf{q}_1 \mathbf{q}_2 = \begin{bmatrix} (s_1 s_2 - x_1 x_2 - y_1 y_2 - z_1 z_2), \\ (s_1 x_2 + s_2 x_1 + y_1 z_2 - y_2 z_1) \mathbf{i} \\ +(s_1 y_2 + s_2 y_1 + z_1 x_2 - z_2 x_1) \mathbf{j} \\ +(s_1 z_2 + s_2 z_1 + x_1 y_2 - x_2 y_1) \mathbf{k} \end{bmatrix} \quad (3.61)$$

1 a tedy ve vektorové notaci lze psát:

$$\mathbf{q}_1 \mathbf{q}_2 = [(s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2), (s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)] = [s, x\mathbf{i} + y\mathbf{j} + z\mathbf{k}] \quad (3.62)$$

2 kde: $(s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2)$ je skalární hodnota a $(s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$ je vektor.

3

4 Ryzí kvaterniony

5 Pokud je $s = 0$, pak se takový kvaternion nazývá „*ryzí kvaternion*“ (pure quaternion).

6 Pokud \mathbf{q}_1 a \mathbf{q}_2 jsou ryzí kvaterniony, tj.

$$\begin{aligned} \mathbf{q}_1 &= [0, \mathbf{v}_1] = [0, x_1 \mathbf{i} + y_1 \mathbf{j} + z_1 \mathbf{k}] \\ \mathbf{q}_2 &= [0, \mathbf{v}_2] = [0, x_2 \mathbf{i} + y_2 \mathbf{j} + z_2 \mathbf{k}] \end{aligned} \quad (3.63)$$

7 pak se některé operace zjednoduší.

8

9 Násobení ryzích kvaternionů

10 Lze snadno nahlédnout, že pro ryzí kvaterniony platí:

$$\mathbf{q}_1 \mathbf{q}_2 = [-\mathbf{v}_1 \cdot \mathbf{v}_2, \mathbf{v}_1 \times \mathbf{v}_2] = [s, x\mathbf{i} + y\mathbf{j} + z\mathbf{k}] \quad (3.64)$$

11

12 Je nutné zdůraznit, že součin kvaternionů není komutativní, tj.

$$\mathbf{q}_1 \mathbf{q}_2 \neq \mathbf{q}_2 \mathbf{q}_1 \quad (3.65)$$

13

14 Obdobě jako u komplexních čísel, kdy bylo k číslu $\mathbf{z} = a + ib$ definováno číslo komplexně sdružené
15 $\bar{\mathbf{z}} = a - ib$, je definován také **kvaternion sdružený**, a to:

$$\bar{\mathbf{q}} = [s, -x\mathbf{i} - y\mathbf{j} - z\mathbf{k}] \quad (3.66)$$

16 a je tedy zřejmé, že:

$$\mathbf{q} \cdot \bar{\mathbf{q}} = \mathbf{q}\bar{\mathbf{q}} = s^2 + x^2 + y^2 + z^2 \quad (3.67)$$

17 kde „·“ značí skalární součin.

18

19 Inverzní kvaternion

20 Pro daný kvaternion $\mathbf{q} = [s, \mathbf{v}] = [s, x\mathbf{i} + y\mathbf{j} + z\mathbf{k}]$ je definován pro $\mathbf{q} \neq 0$ kvaternion inverzní, a to:

$$\mathbf{q}^{-1} = \frac{[s, -x\mathbf{i} - y\mathbf{j} - z\mathbf{k}]}{\|\mathbf{q}\|^2} \quad (3.68)$$

21 kde $\|\mathbf{q}\|$ je velikost (magnitude, modulus) kvaternionu \mathbf{q} , která je určena:

$$\|\mathbf{q}\|^2 = s^2 + x^2 + y^2 + z^2 \quad (3.69)$$

22 Je zřejmé, že platí:

$$\mathbf{q}\mathbf{q}^{-1} = \mathbf{q}^{-1}\mathbf{q} = 1 \quad (3.70)$$

23

24 Jednotkový kvaternion

25 Kvaternion se nazývá jednotkovým, pokud $\|\mathbf{q}\| = 1$, tj. $\mathbf{q}\bar{\mathbf{q}} = 1$

26

27 3.4.4. Kvaterniony a rotace

28 Rotace ve 2D je dána vztahem:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{R} \mathbf{x} \quad (3.71)$$

29 přičemž platí:

$$\det(\mathbf{R}) = A^2 + B^2 = 1 \quad (3.72)$$

30 pak také platí:

$$\mathbf{R}\mathbf{R}^T = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} A & B \\ -B & A \end{bmatrix} = \begin{bmatrix} A^2 + B^2 & 0 \\ 0 & A^2 + B^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I} \quad (3.73)$$

1 Předpokládejme nyní, že:

$$2 \quad A = a^2 - b^2 \quad B = 2ab \quad (3.74)$$

3 Lze ukázat, že pokud $a = \cos(\varphi/2)$ a $b = \sin(\varphi/2)$, pak

$$A = \cos^2(\varphi/2) - \sin^2(\varphi/2) = \cos \varphi \\ B = 2 \cos(\varphi/2) \sin(\varphi/2) = \sin \varphi \quad (3.75)$$

4 Je tedy zřejmé, že matice rotace nebyla změněna, neboť:

$$5 \quad \mathbf{R} = \begin{bmatrix} a^2 - b^2 & 2ab \\ 2ab & a^2 - b^2 \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \quad (3.76)$$

a dále pak platí:

$$6 \quad \det(\mathbf{R}) = (a^2 - b^2)^2 + (2ab)^2 = (a^2 + b^2)^2 = 1 \quad (3.77)$$

V komplexní formě můžeme psát:

$$7 \quad e^{i\varphi/2} = \cos(\varphi/2) + i \sin(\varphi/2) = a + ib \quad (3.78)$$

8 Kvaterniony jsou v zásadě operace s vektory. Souřadnice bodu jsou proto reprezentovány pozičním vektorem v souřadním systému. Pak bod $P(x, y, z)$ je reprezentován kvaternionem \mathbf{p} :

$$9 \quad \mathbf{p} = [0, \mathbf{v}] = [0, xi + yj + zk] \quad (3.79)$$

10 Rotace podle obecné osy v E^3 je určena směrovým vektorem osy rotace \mathbf{a} o úhel φ a je dána kvaternionem:

$$11 \quad \mathbf{q} = [\cos(\varphi/2), \mathbf{a} \sin(\varphi/2)] \quad (3.80)$$

a vlastní rotace je určena:

$$12 \quad \mathbf{x}' = \mathbf{q} \mathbf{x} \mathbf{q}^{-1} \quad (3.81)$$

13 Pro jednotkový kvaternion lze použít místo inverzního kvaternionu kvaternion sdružený:

$$14 \quad \mathbf{x}' = \mathbf{q} \mathbf{x} \bar{\mathbf{q}} \quad (3.82)$$

15 Zde je vhodné zdůraznit, že výpočet bude pomalejší než výpočet využívající postupu uvedeného v kap. 4.7 (Transformace rotace v E^3 okolo osy založená na rotaci vektorů).

16
17 Vzhledem k tomu, že hardware podporuje operace typu násobení matice vektor, používá se maticový ekvivalent, kdy matice \mathbf{R} je dána prvky jednotkového kvaternionu \mathbf{q} takto:

$$18 \quad \mathbf{R} = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2(xy - sz) & 2(xz + sy) & 0 \\ 2(xy + sz) & 1 - 2z^2 - 2x^2 & 2(yz - sx) & 0 \\ 2(xz - sy) & 2(yz + sx) & 1 - 2x^2 - 2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

19 Podrobnější informace lze nalézt např. v:

- 20 • Gortler,S.J.: Foundations of 3D Computer Graphics, MIT Press, ISBN 978-0-262-01735-0, 2012
- 21 • Salomon,D.: The Computer Graphics manual, Springer, ISBN 978-0-85729-885-0, 2011
- 22 • Salomon,D.: Transformations and Projections in Computer Graphic, Springer,
- 23 ISBN 978-1-84628-392-5, 2006
- 24 • Hanson,A.J.: Visualizing Quaternions, Morgan Kaufmann Publ., ISBN 978-0-12-088400-1,
- 25 2006
- 26 • Vince,J.: Matrix Transforms for Computer Games and Animation, Springer,
- 27 ISBN 978-1-4471-4320-8, 2012

3.5. Plückerovy souřadnice

Plückerovy souřadnice se nepoužívají příliš často, ale zejména při použití projektivní reprezentace, tj. homogenních souřadnic, lze řešit některé problémy efektivně.

Přímka v E^3 v parametrickém tvaru je určena rovnicí:

$$\mathbf{X}(t) = \mathbf{X}_1 + (\mathbf{X}_2 - \mathbf{X}_1) t \quad (3.83)$$

kde: $\mathbf{X} = (X, Y, Z)$.

V projektivním prostoru P^3 pak:

$$\mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1) t \quad (3.84)$$

kde: $\mathbf{x} = [x, y, z : w]^T$.

Přímka v E^3 je dána body $\mathbf{x}_1 = [x_1, y_1, z_1 : w_1]^T$ a $\mathbf{x}_2 = [x_2, y_2, z_2 : w_2]^T$. Plückerovy souřadnice l_{ij} jsou definovány:

$$\begin{aligned} l_{41} &= w_1 x_2 - w_2 x_1 & l_{23} &= y_1 z_2 - y_2 z_1 \\ l_{42} &= w_1 y_2 - w_2 y_1 & l_{31} &= z_1 x_2 - z_2 x_1 \\ l_{43} &= w_1 z_2 - w_2 z_1 & l_{12} &= x_1 y_2 - x_2 y_1 \end{aligned} \quad (3.85)$$

Alternativně pomocí matice \mathbf{L} :

$$\mathbf{L} = \mathbf{x}_1 \mathbf{x}_2^T - \mathbf{x}_2 \mathbf{x}_1^T \quad (3.86)$$

kde: $l_{ij} = -l_{ji}$ a $l_{ii} = 0$.

Definujme vektory $\boldsymbol{\omega}$ a \boldsymbol{v} :

$$\boldsymbol{\omega} = (l_{41}, l_{42}, l_{43}) \quad \boldsymbol{v} = (l_{23}, l_{31}, l_{12}) \quad (3.87)$$

Pokud souřadnice bodů jsou v Eukleidovských souřadnicích, tj. $w = 1$, pak je zřejmé, že vektor $\boldsymbol{\omega}$ reprezentuje "směrový" vektor a vektor \boldsymbol{v} reprezentuje "poziční" vektor. Pro $w = 1$ pak dosáváme:

$$\boldsymbol{\omega} = \mathbf{X}_2 - \mathbf{X}_1 \quad (3.88)$$

Pokud $w \neq 0$ a obecně $w \neq 1$ dostáváme:

$$\mathbf{X}_2 - \mathbf{X}_1 = \left(\frac{x_2}{w_2} - \frac{x_1}{w_1}, \frac{y_2}{w_2} - \frac{y_1}{w_1}, \frac{z_2}{w_2} - \frac{z_1}{w_1} \right) \quad (3.89)$$

Lze snadno nahlédnout, že pro projektivní prostor obecně dostáváme:

$$\begin{aligned} \boldsymbol{\omega} &= w_1 w_2 (\mathbf{X}_2 - \mathbf{X}_1) = (w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1, w_1 z_2 - w_2 z_1) \\ &= (l_{41}, l_{42}, l_{43}) \end{aligned} \quad (3.90)$$

a

$$\boldsymbol{v} = w_1 w_2 (\mathbf{X}_1 \times \mathbf{X}_2) = (y_1 z_2 - y_2 z_1, z_1 x_2 - z_2 x_1, x_1 y_2 - x_2 y_1) = (l_{23}, l_{31}, l_{12}) \quad (3.91)$$

V roce 1871 Klein dokázal, že:

$$\boldsymbol{\omega}^T \boldsymbol{v} = 0 \quad l_{23} * l_{41} + l_{31} * l_{42} + l_{12} * l_{43} = 0 \quad (3.92)$$

Gibbs dokázal, že výše uvedená rovnice je ve skutečnosti rovnice 2. stupně, a proto řešení je na 4 –dimensionálním kvadratickém hyperpovrchu a pokud \mathbf{q} je bod na přímce $\mathbf{q}(t) = \mathbf{q}_1 + \boldsymbol{\omega} t$

Pak musí platit podmínka:

$$\boldsymbol{\omega} \times \mathbf{q} = \boldsymbol{v} \quad (3.93)$$

a bod na přímce $\mathbf{q}(t)$ je dán:

$$\mathbf{q}(t) = \frac{\boldsymbol{v} \times \boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega} t \triangleq [\boldsymbol{v} \times \boldsymbol{\omega} + \boldsymbol{\omega} \|\boldsymbol{\omega}\|^2 t : \|\boldsymbol{\omega}\|^2]^T \quad (3.94)$$

Pokud $\|\boldsymbol{\omega}\| = 0$ pak body \mathbf{x}_1 a \mathbf{x}_2 jsou totožné.

Protože bod a rovina jsou duální pojmy v E^3 dostáváme automaticky i formuli pro výpočet průsečnice dvou rovin, stačí zamenit jen pojmy bod a rovina.

28

1 Podrobné odvození viz:

- 2 • Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates,
3 Int.Journal on Image and Graphics, DOI No: [10.1142/S021946780800326X](https://doi.org/10.1142/S021946780800326X), ISSN 0219-4678,
4 Vol.8, No.4, pp.615-628, 2008
5 ([CLICK](#) off-line)

6

7 Další podrobné informace lze nalézt:

- 8 • F. Klein: Notiz Betreffend dem Zusammenhang der Liniengeometrie mit der Mechanik starrer
9 Körper., Math. Ann. 4 , pp.403- 415, 1871
10 • Gibson,C.,G. and Hunt,K.,H. (1992) Geometry of Screw Systems. Mech. Machine Theory,
11 Vol.12, pp.1-27, 1992
12 • Blinn,J.F.: A Homogeneous Formulation for Lines the in E3 Space, ACM SIGGRAPH, Vol.11,
13 No.2., pp.237-241, 1997.

14

15 Plückerovy souřadnice lze s výhodou např. použít pro test, zda orientovaná přímka (paprsek) protíná
16 daný trojúhelník v E^3 .

17

18

19

20

3.6.Numerická reprezentace a stabilita výpočtu

V současném technologicky orientovaném světě je zapotřebí posuzovat data, a tedy i informace, v širších souvislostech, kdy hlavními faktory jsou bezesporu:

- velký objem dat, který je nutné posoudit podle jednoho či více kritérií
- data jsou vícerozměrná, tj. multidimenzionální
- velká výpočetní náročnost potřebná nejen pro vlastní řešení, ale i pro generování výstupu např. pro vizualizaci dat
- inovační technologický cyklus je kratší než 6 měsíců

	Decimal usage	Binary usage
GigaByte [GB]	10^9	2^{30}
TeraByte [TB]	10^{12}	2^{40}
PetaByte [PB]	10^{15}	2^{50}
ExaByte [EB]	10^{18}	2^{60}
ZettaByte [ZB]	10^{21}	2^{70}
YottaByte [YB]	10^{24}	2^{80}
????	??	$2^{??}$

Obr.3.10: Zkratky pro kapacity pamětí

Dalšími faktory pak jsou také, že:

- každých 18 měsíců se zdvojnásobí výpočetní kapacita systémů, tj. za 15 let budou k dispozici pravděpodobně systémy s výpočetní i paměťovou kapacitou $2^{10} = 1024$ násobně vyšší než dnes
- limitujícím faktorem zřejmě v dohledné době zůstane i přesnost výpočtů v pohyblivé řádové čárce daná standardem IEEE 578-2008 apod.
- objem získaných nebo generovaných dat, která jsou zpracovávána a zobrazována, bude narůstat rychleji než lineárně
- lidská schopnost vyhodnocovat data zůstává víceméně stejná, neboť je dána fyziologií člověka

Je tedy více než oprávněná otázka, jaké problémy budeme za 10 let řešit, jak je budeme zpracovávat, jaká data a jakým způsobem je budeme zobrazovat, resp. vizualizovat, a pomocí jakých zařízeních.

Bohužel i v současnosti se lze setkat s typickou ignorancí problémů spojených s konečnou přesností reprezentace dat a lze běžně nalézt konstrukce, které typicky vedou k numerickým problémům.

Typickými ukázkami špatného kódu jsou:

- Testy `A ? B` pro hodnoty v pohyblivé řádové čárce, např.
`if A = B then else ; if A ≠ 0 then else`
- Takové konstrukce by měly být zakázány v programovacích jazycích.
- nekonečné cykly v důsledku hodnot v pohyblivé řádové čárce, např.
`double x = -1; double p =;`
`while (x < +1)`
`{ if (x == p) Console.Out.WriteLine(" *** "); x += p; }`
`/* if p = 0.1 then no output, if p = 0.25 then expected output */`

1 **3.6.1. Vybrané příklady a ukázky numerických problémů**

2 Z praxe je k dispozici celá řada příkladů, kdy numerické výpočty selhávají. Uvedeme alespoň ty
3 nejzákladnější, které by měl mít uživatel na paměti (pro názornost je použita jednoduchá přesnost).

5 **Výpočet prosté sumy hodnot:**

$$\sum_{i=1}^{10^3} 10^{-3} = 0.999990701675415 \quad \sum_{i=1}^{10^4} 10^{-4} = 1.000053524971008 \quad (3.95)$$

6 Výsledek by měl být vždy roven hodnotě 1.

9 **Závislost na pořadí výpočtu** – výsledek také závisí na pořadí výpočtu:

$$\sum_{n=1}^{10^6} \frac{1}{n} = 14.357357 \quad \sum_{n=10^6}^1 \frac{1}{n} = 14.392651 \quad (3.96)$$

10 Na výše uvedených příkladech vidíme, že výpočet, byť i „primitivních“ matematických formulí,
11 nemusí vést nutně ke správnému numerickému výsledku.

14 **Konverze mezi datovými typy**, které nemusí programový systém kontrolovat s následnou fatální
15 ztrátou přesnosti výpočtu, např. **float → integer**, viz např. Explosivní raketový experiment Ariane, kap.3.7.

17 **Neplatnost „matematických“ identit** - i když se zdají být výše uvedené příklady „umělými“, je nutné
18 si uvědomit, že **matematické identity nejsou obecně platné** při použití reprezentace čísel s omezenou
19 přesností, např. identity:

$$\cos^2 \alpha + \sin^2 \alpha = 1 \quad x^2 - y^2 = (x - y)(x + y) \quad (3.97)$$

20 v důsledku omezené numerické reprezentace obecně *neplatí*.

22 Také řešení obyčejné kvadratické rovnice nemusí být korektní, pokud použijeme „standardní“ řešení.

$$at^2 + 2bt + c = 0 \quad t_{1,2} = \frac{-b \pm \sqrt{b^2 - ac}}{a} \quad (3.98)$$

23 Pro výpočet diskriminantu se musí použít dvojnásobná přesnost, než je reprezentace ostatních
24 proměnných.

26 Duální formule pro $t \neq 0$ pak vydělením:

$$a + 2\frac{b}{t} + \frac{c}{t^2} = 0 = a + c\left(\frac{1}{t^2} + \frac{b}{ct}\right) \quad (3.99)$$

27 Protože:

$$c\left(\frac{1}{t} + \frac{b}{c}\right)^2 = c\left(\frac{b}{c}\right)^2 - a = \frac{b^2}{c} - \frac{ac}{c} = \frac{b^2 - ac}{c} \quad (3.100)$$

28 pak:

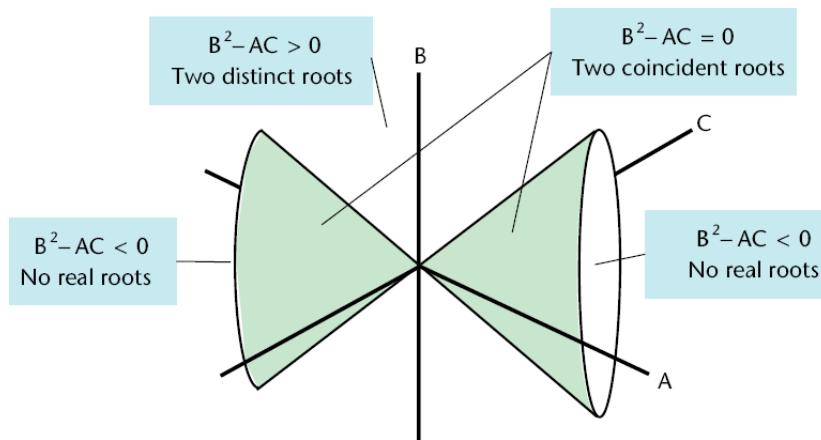
$$\frac{1}{t} + \frac{b}{c} = \pm \frac{\sqrt{b^2 - ac}}{c} \quad (3.101)$$

29 a tedy:

$$t = \frac{c}{-b \pm \sqrt{b^2 - ac}} \quad (3.102)$$

30 Tato formule je vhodná pokud $b^2 \gg ac$.

31



Obr.3.11: Řešení kvadratické rovnice - problém stability

Pokud $b^2 \ll ac$, pak by měla být použita „duální“ formule:

$$q = -(b + \text{sign}(b)\sqrt{b^2 - ac}) \quad t_1 = q/a \quad t_2 = c/q$$

Výše uvedená formule vychází z Vietovy věty, kde:

$$t_1 + t_2 = -b/a \quad t_1 t_2 = c/q \quad (3.103)$$

$$t_1 t_2 = \left(\frac{-b - \sqrt{b^2 - ac}}{a} \right) \left(\frac{-b + \sqrt{b^2 - ac}}{a} \right) = \frac{c}{a} \quad (3.104)$$

Příklad

Zajímavá situace nastává pro $\sqrt{b^2 - ac} \cong |b|$. Pro jednoduchost uvažme např. rovnici:

$$f(t) = t^2 + 2 \times 10^6 t + 1 = 0 \quad (3.105)$$

Jaké výsledky dostaneme pro jednotlivé reprezentace v pohyblivé řádové čárce?

Podrobněji viz:

- Blinn,J.F.: How to Solve a Quadratic Equation, IEEE Computer Graphics&Application, Vol.25, No.6, pp.76-79, 2005 ([CLICK](#) off-line).
- Blinn,J.F.: How to Solve a Quadratic Equation – Part 2, IEEE Computer Graphics&Application, Vol.26, No.2, pp.82-87, 2005 ([CLICK](#) off-line).

Úloha

Předpokládejme kvadratickou rovnici:

$$f(t) = t^2 + 2 \times 10^6 t + 1 = 0 \quad (3.106)$$

3

4

5 a určete řešení v jednoduché přesnosti podle vzorců

$$t_{1,2} = \frac{-b \pm \sqrt{b^2 - ac}}{a} \quad (3.107)$$

$$q = -(b + sign(b)\sqrt{b^2 - ac}) \quad t_1 = q/a \quad t_2 = c/a \quad (3.108)$$

6

Výpočet funkčních hodnot

8 Uvedeme zde, sice trochu umělý, ale zajímavý problém, a to výpočet hodnoty funkce:

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y) \quad (3.109)$$

9 v bodě $x = 77617$, $y = 33096$.

10

11 Výsledek, který obdržíme standardním výpočtem, je pak:

- 12 • $f = 6.33835 \cdot 10^{29}$ jednoduchá přesnost
- 13 • $f = 1,1726039400532$ dvojnásobná přesnost
- 14 • $f = 1,1726039400531786318588349045201838$ „extended“ přesnost

15

16 avšak **správný výsledek** při použití intervalové aritmetiky:

$$[-0,82739605994682136814116509547981629\mathbf{2005}, \\ -0,82739605994682136814116509547981629\mathbf{1986}]$$

17

18 **Přesné řešení** je pak dáno výrazem:

$$f(x, y) = -2 + \frac{x}{2y} = \frac{54767}{66192} \quad (3.110)$$

19

20 **Existují tedy případy, kdy prostým prodlužováním mantisy nemusíme dojít ke korektnímu výsledku.**

21

1 **3.6.2. Rekurze**

2 Rekurze je dalším potencionálním zdrojem chyb. Rekurze je vlastně volání funkce sebe sama, ať už
3 přímo nebo zprostředkovaně, tj. tzv. vzájemnou rekurzí.

4 Nebezpečnými faktory zejména jsou:

- 5 • rekurze, kdy hloubka rekurze i při středně rozsáhlém výpočtu nemusí být výpočetním
6 systémem zvládnutelná. Typickou úlohou je např. problém „Hanojských věží“, který je
7 definován takto:

8
9 MOVE (A, C, n);
10 { **if** n > 1 **then**
11 { MOVE (A, B, n-1); MOVE (A, C, 1); MOVE (B, C, n-1) }
12 } # MOVE (from, to, number) #
13

14 Úkolem je přesunout n disků z tyče A na tyče C s použitím tyče B tak, že v žádném okamžiku
15 nesmí být větší disk na menším disku.

- 16 • rekurze, při níž datový typ není schopen reprezentovat hodnotu pro její velikost, tj. při
17 „přetečení“ bez detekce chybného stavu. Toto je spojeno zejména s reprezentací celých čísel,
18 kdy přetečení není detekováno současným hardwarem.

19
20 Typickou ukázkou je Ackermannova funkce, která je definována:

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases} \quad (3.111)$$

21 Hodnoty Ackermannovy funkce rostou extrémně rychle, např.

$$A(4, 4) = 2^{2^{2^{65536}}} = 2^{2^{10^{197296}}}$$

22 Podrobněji viz http://en.wikipedia.org/wiki/Ackermann_function
23

24 **3.6.3. Další možnosti zvýšení přesnosti**

25 Pro zvýšení přesnosti je možné použít např.:

- 26 • intervalové aritmetiky, kdy se předpokládá, že uložená hodnota vlastně reprezentuje celý
27 interval hodnot. Uvedeme základní pravidla pro intervalovou aritmetiku:

- 28 ○ $x + y = [a + c, b + d]$ $x = [a, b]$
- 29 ○ $x - y = [a - d, b - c]$ $y = [c, d]$
- 30 ○ $x \times y = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
- 31 ○ $x / y = [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)]$ if $y \neq 0$

- 32 • speciální reprezentace hodnot, např. pomocí řetězových zlomků apod. Pro názornost
33 uveďme vyjádření hodnoty $\pi = [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1 \dots]$, což je reprezentace
34 řetězového zlomku:

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{\dots}}} \quad (3.112)}$$

35 viz http://en.wikipedia.org/wiki/Generalized_continued_fraction
36

1 **3.6.4. Příklad překvapivé nestability výpočtu**

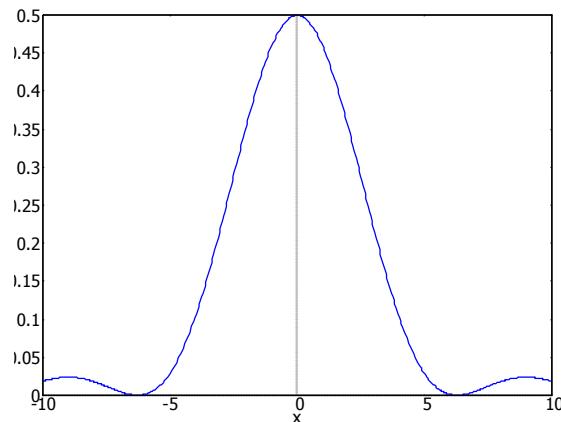
2 Další ukázkou nepřesnosti výpočtu je funkce:

$$f(x) = \frac{1 - \cos x}{x^2} \quad (3.113)$$

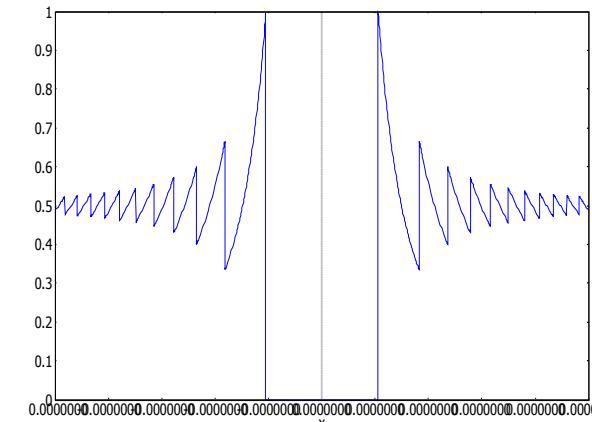
3 Graf funkce je uveden na Obr.3.12.a. Lze ukázat, že $f(0) = 0,5$. Nicméně pro hodnoty okolo počátku
4 se funkce chová „zvláštně“, viz Obr.3.12.b vlivem numerické nepřesnosti a přesný průběh závisí silně
5 na realizaci výpočtu vzorce a použitém hardwaru.

6 Vypočtená hodnota funkce je v intervalu $(-\varepsilon, \varepsilon)$ dokonce **nulová**.

7



a) Průběh funkce $f(x) = \frac{1 - \cos x}{x^2}$



b) Průběh funkce okolo počátku $(-10^{-8}, 10^{-8})$

Obr.3.12: Numerická nepřesnost výpočtu funkce

8

9 **Poznámka**

10 Je nutné si uvědomit, že hodnota funkce $f(0)$ není definována pouze pro $x = 0$, neboť jde o výraz
11 $0/0$, avšak numerické problémy vznikají již v poměrně velkém okolí tohoto bodu.

12 Hodnotu $f(0)$ lze určit pomocí L'Hospitalova pravidla:

$$f(0) = \lim_{x \rightarrow 0} \frac{1 - \cos x}{x^2} = \lim_{x \rightarrow 0} \frac{g(x)}{h(x)} = \lim_{x \rightarrow 0} \frac{g'(x)}{h'(x)} = \lim_{x \rightarrow 0} \frac{\sin x}{2x} \quad (3.114)$$

13 což je opět výraz $0/0$. Opětovným použitím L'Hospitalova pravidla dostaváme:

$$f(0) = \lim_{x \rightarrow 0} \frac{\sin x}{2x} = \lim_{x \rightarrow 0} \frac{\cos x}{2} = 0,5 \quad (3.115)$$

14

15 Zde je nanejvýš vhodné zdůraznit, že existuje celá řada dnes méně známých, ale stále používaných
16 programovacích jazyků, které poskytují specifické datové typy a zajímavé datové konstrukce, např.
17 programovací jazyk **Algol68** má konstrukci **long**, která prodlužuje délku reprezentace, a tedy i
18 přesnost datového typu. Programátor pak může mít i velmi dlouhou mantisu použitím konstrukce
19 **longlong real A**.

20

21 Jak je vidět, někdy i „koncepčně zastaralé“ programovací jazyky nabízejí více než ty „nové“.

22

23 V praxi lze nalézt mnoho výpočetních postupů, které se snaží eliminovat vliv konečné reprezentace
24 různými způsoby, např. pro geometrické úlohy se používá knihovna CGAL.

25 Výše uvedený přehled snad poslouží k náhledu, že tato problematika není rozhodně jednoduchá a že
26 v praxi se často vyskytuje.

3.7.Příklady katastrof způsobených výpočetními chybami

Je známa celá řada katastrof způsobených špatným numerickým výpočtem. Po přečtení původní zprávy o příčinách, následné analýzy a doprovodných komentářů každý musí být překvapen, jaké elementární chyby se staly, jak se podcenilo testování správnosti a korektnosti výpočtů. Bohužel se zřejmě stále častěji bude stávat, že takové chyby nebudou eliminovány a vzhledem ke vztřístající složitosti systémů budou pravděpodobně i důsledky takových katastrof větší. Uvedeme alespoň pro ilustraci několik nejznámějších případů.

Explose rakety Ariane 5

Evropská agentura ESA (European Space Agency) vypustila 4. června 1996 raketu Ariane 5. Její vývoj stál přes 7 miliard US\$. Cena vlastního nákladu a rakety byla přes 500 milionů US\$. Raketa explodovala asi 40 [s] po startu. Příčinou bylo selhání programového vybavení inerciálního naváděcího systému SRI (Inertial Reference System), které bylo zapříčiněno konverzí 64 bitového čísla v pohyblivé řádové čárce do 16 bitového čísla v celočíselné reprezentaci a hodnota byla větší než hodnota reprezentovatelná na 16 bitech.

Podrobné informace viz: <http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html>



Obr.3.13: Kolaps rakety Ariane Courtesy CNN

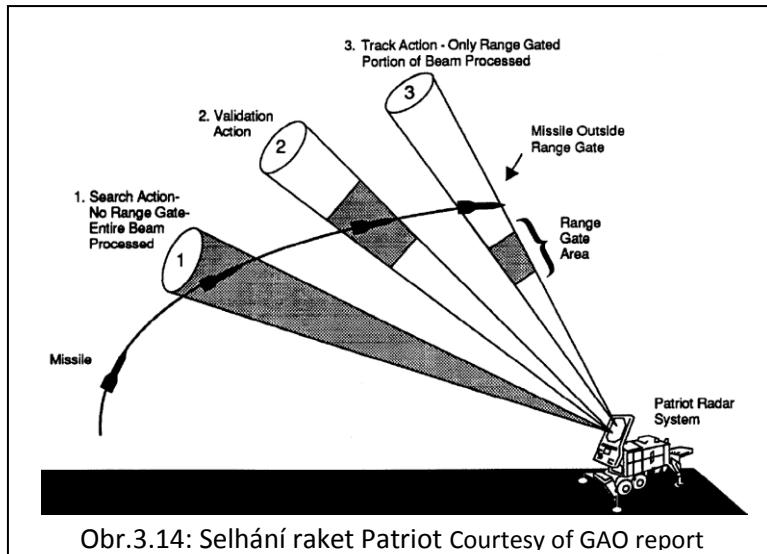
Selhání antirakety Patriot

Systém protivzdušné raketové obrany Patriot byl původně navržen pro krátké a operativní nasazení po roce 1960. Byl navržen pro sestřelení raket s rychlostí MACH 2. Systém byl při praktickém nasazení v pohotovostním režimu přes 100 hod. a byl použit pro sestřelení rakety typu SCUD letící rychlostí MACH 5. Výpočet pro detekci a určení pozice byl založen na celočíselné reprezentaci o délce 24 bitů a s časovým taktem 1/10 [s] a výpočtem rychlosti v pohyblivé řádové čárce. Nastavení časového taktu 1/10 [s] bylo kritickým bodem, neboť není ani postačující pro sportovní účely.

Bohužel $1/10 = 1/2^4 + 1/2^5 + 1/2^8 + 1/2^9 + 1/2^{12} + \dots$ nemá konečnou reprezentaci a chyba na 42 bitech byla asi 0.000000095. V průběhu 100 hod tak dosáhla hodnoty 0.34. Protože rakety typu SCUD mají rychlosť MACH 5, aktuální chyba byla 687[m] a raketa byla mimo „okno“ pro samonavádění. O daném problému se vědělo, příslušný software korigující problém byl k dispozici, ale pracovníci neprovědli příslušný update programového vybavení.

Důsledkem špatných předpokladů, nekorektní technické a programové realizace a nezodpovědného přístupu bylo, že 25. února 1991 při útoku v Dhahranu iráckou raketou SCUD bylo 28 US vojáků zabito, 100 lidí na základně zraněno.

Podrobnosti lze nalézt v reportu GAO: <http://www.fas.org/spp/starwars/gao/im92026.htm>

1
2
3
4

Dalším velmi známým případem je potopení těžební plošiny Sleipner A v Severním moři.

5
6 **Potopení těžební plošiny Sleipner A**
 7 V roce 1991 se potopila těžební plošina Sleipner A. Jen pro ilustraci uveďme, že samotná těžební
 8 plošina vážila přes 57 000 tun, nástroje a zařízení pak 40 000 tun a na plošině pracuje průběžně přes
 9 200 osob. Struktura těžební plošiny byly „optimalizována“ s použitím systému založeného na
 10 konečných prvcích a smyková napětí byla podhodnocena téměř o 50%. Toto vedlo k porušení
 11 struktury a trhlinám s následným průnikem vody, který pumpy nebyly schopny zvládnout.
 12 Odhadovaná cena potopené těžební plošiny činí cca 700 miliónů US\$.

13
14

15

1 Až dosud jsme se zabývali elementárními datovými typy. Nicméně je asi vhodné se podívat na
 2 problém i z jiné strany. V převážné většině dnes běžně zpracovávaných dat lze najít dvě hlavní datové
 3 množiny, které jsou obvykle viděny samostatně, a to geometrická data (včetně obrazových) a data
 4 textová.

5 Zkusme se tedy podívat, jaké jsou rozdíly mezi textovými data a daty geometrickými.

		Dimenzionalita	
		Malá	„Nekonečná“
Interval hodnot	Malý	Obrazová data Dimenze 2, 3 Interval hodnot $\langle 0,255 \rangle$ apod.	Textová data Dimenze, tj. délka řetězce dlouhá Interval hodnot, např. $\langle 0,255 \rangle$ pro ASCII
	„Nekonečný“	Geometrická data Dimenze 2, 3 Interval hodnot $(-\infty, +\infty)$	Harmonická analýza apod.

6 Tab.xx: Rozdílnost textových a geometrických dat

- 7 • *textová data* – rozsah hodnot je dán použitou alfabetou, např. 256 pro ASCII, zatímco
 8 dimenzionalita je „nekonečná“, neboť řetězce mohou být velmi dlouhé, např.:
 - 9 ○ protein titin je popsán 189 819 znaky
 - 10 ○ železniční stanice *Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch* ve
 11 Walesu atd.
 12 a počet slov, tj. zpracovávaných elementů je např. pro český slovník cca 2-3 mil. slov
- 13 • *geometrická data* – obvykle mají jen 2, resp. 3 souřadnice $\langle x, y \rangle$, resp. $\langle x, y, z \rangle$, ale
 14 „nekonečný“ interval hodnot $(-\infty, +\infty)$, přičemž běžně zpracováváme $10^6 - 10^{15}$ hodnot.

15

16

4. Základní geometrické transformace

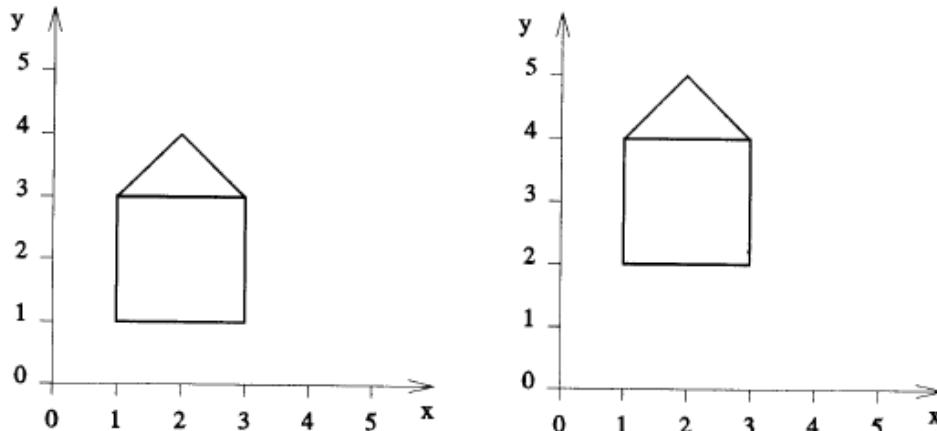
Geometrické transformace v počítačové grafice jsou využívány ke geometrickým manipulacím s objekty, resp. jejich částí, při vytváření objektů, tj. modelování, při animaci, kdy potřebujeme „rozpohybovat“ jednotlivé části objektu, v počítačových hrách atd. Mezi základní operace patří zejména posuv, rotace, změna měřítka a také rovinné projekce. Jednotlivé základní operace, které jsou reprezentovány jednotlivými maticemi geometrických transformací, se pak řetězí a vzniká pak jedna matice, která reprezentuje celkovou geometrickou transformaci. Zde je vidět výhoda použití projektivní reprezentace, viz kap.3.2 (Homogenní souřadnice a jejich geometrická interpretace). Jednotlivé transformace jsou popsány jako transformace pro jeden bod, je však nutné si uvědomit, že daná transformace se aplikuje na všechny body daného objektu, resp. té části, která se transformuje. U reálných úloh je takto transformováno běžně $10^5 - 10^{10}$ a i více bodů. Navíc v mnoha případech se vyžaduje, aby celý proces včetně generování výstupního obrazu probíhal v reálném čase.

Všechny geometrické transformace budou popsány rovnicí $\mathbf{x}' = \mathbf{Q}\mathbf{x}$ se sloupcovou notací vektorů. Je nutné zdůraznit, že mnoho publikací používá řádkovou notaci. Výše uvedený vztah v řádkové notaci má tvar $\mathbf{x}'^T = \mathbf{x}^T \mathbf{Q}^T$, tj. matice transformace je transponovaná. To v případě E^2 znamená např. rotaci o úhel opačný.

4.1. Základní transformace v E^2

Pro jednoduchost uveďme nejdříve základní operace v rovině, tj. v E^2 .

4.1.1. Posuv (translation)



Obr.4.1: Transformace posunu

Geometrická transformace posudu je reprezentována vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \mathbf{x}' = \mathbf{T}(A, B)\mathbf{x} \quad (4.1)$$

V dalším výkladu budeme používat zkráceného zápisu $\mathbf{x}' = \mathbf{T}(A, B)\mathbf{x}$. Všimněme si, že $\det(\mathbf{T}) = 1$. Rozepsáním a použitím formálních úprav je zřejmé, že výše uvedený vztah reprezentuje posuv o vektor (A, B) :

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x + Aw \\ y + Bw \\ w \end{bmatrix} \triangleq \begin{bmatrix} x/w + A \\ y/w + B \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + B \\ 1 \end{bmatrix} \quad (4.2)$$

kde \triangleq značí projektivní ekvivalence.

- 1 Zde je vhodné připomenout, že homogenní w souřadnice transformovaných bodů může být $w \neq 1$.
 2 Tedy není nutná jejich konverze do Eukleidovského prostoru.

3

4 **Inverzní operace posuvu**

- 5 Inverzní operace je jednoduchá, neboť jde vlastně o posuv daný vektorem
- $(-A, -B)$
- a tedy:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & -A \\ 0 & 1 & -B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad x' = \mathbf{T}(-A, -B)x = \mathbf{T}^{-1}(A, B) \quad (4.3)$$

6

- 7 Lze ukázat, že pokud vektor posuvu $[a, b: c]^T \triangleq (A, B)$ je v homogenních souřadnicích, tj. obecně
 8 $c \neq 1$, lze operaci posuvu realizovat bez použití operace dělení. V tomto případě je pak transformace
 9 určena:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} c & 0 & a \\ 0 & c & b \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} cx + aw \\ cy + bw \\ cw \end{bmatrix} \triangleq \begin{bmatrix} (cx + aw)/(cw) \\ (cy + bw)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} x/w + a/c \\ y/w + b/c \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + B \\ 1 \end{bmatrix} \quad (4.4)$$

- 10 Nyní je
- $\det(\mathbf{T}) = c^3$
- , což však nevadí, neboť operace jsou v projektivním prostoru.

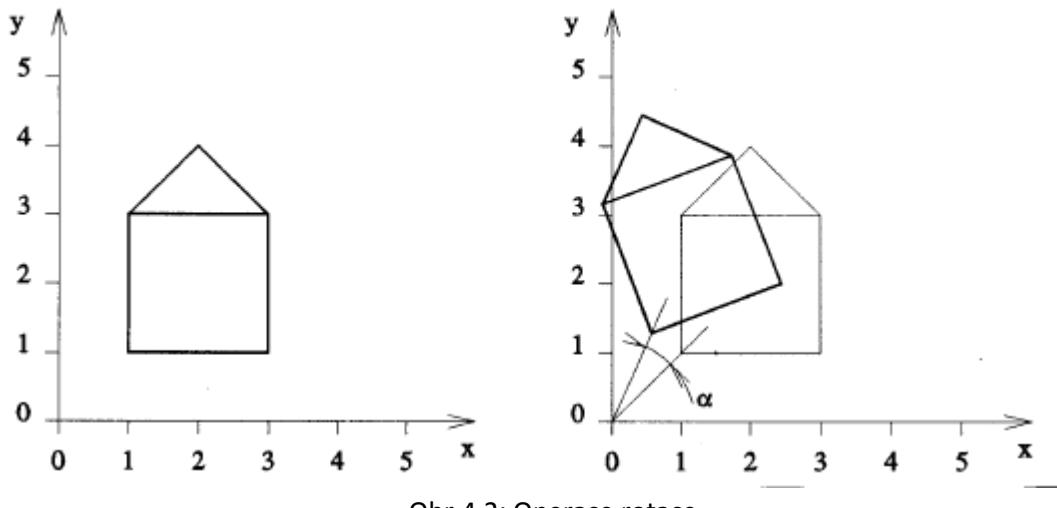
11

- 12 Další základní transformací je transformace rotace.

13

14 **4.1.2. Rotace (rotation)**

- 15 Operace rotace je operace, u které je nutné zdůraznit, že jde o rotaci *okolo počátku souřadného systému*. Pokud střed rotace není v počátku souřadného systému, pak jde o složenou transformaci, podrobněji viz kap.4.2 (Řetězení transformací).



Obr.4.2: Operace rotace

18

- 19 Rotace bodu o úhel
- φ
- proti směru hodinových ručiček je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad x' = \mathbf{R}(\varphi)x \quad (4.5)$$

22

23

1 **Inverzní operace rotace**2 Inverzní operace je opět jednoduchá, neboť jde vlastně o rotaci s úhlem opačným, tj. $x' = R(-\varphi)x$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos(-\varphi) & -\sin(-\varphi) & 0 \\ \sin(-\varphi) & \cos(-\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (4.6)$$

3 Opět platí, že $\det(R) = 1$, tj. matice R je ortonormální a $R^{-1}(\varphi) = R(-\varphi) = R^T(\varphi)$.

4

5 Lze ukázat, že pokud je rotace určena vektorem, který projektivně reprezentuje rotaci ve tvaru

6 $[a, b: c]^T \triangleq \left(\frac{a}{c}, \frac{b}{c} \right) = (\cos\varphi, \sin\varphi)$, pak transformace je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad x' = R'(a, b: c)x \quad (4.7)$$

7

8 Rozepsáním pak lze ukázat korektnost vztahu:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} ax - by \\ bx + ay \\ cw \end{bmatrix} \triangleq \begin{bmatrix} (ax - by)/(cw) \\ (bx + ay)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{w}a - \frac{y}{w}b \\ \frac{x}{w}b + \frac{y}{w}a \\ 1 \end{bmatrix} = \begin{bmatrix} X\cos\varphi - Y\sin\varphi \\ X\sin\varphi + Y\cos\varphi \\ 1 \end{bmatrix} \quad (4.8)$$

9

10 Je nutné podotknout, že $\det(R') = (a^2 + b^2)c = c^3$, neboť $c^2 = a^2 + b^2$. Toto ale nevadí, neboť
11 operace jsou realizovány v projektivním prostoru. Obecně není tedy nutné dělení pro převod
12 parametrů rotace do Eukleidovského prostoru.

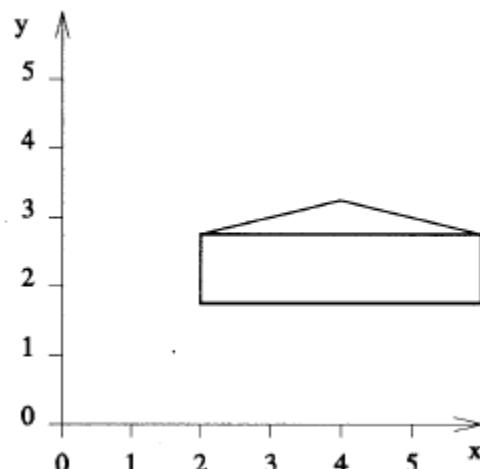
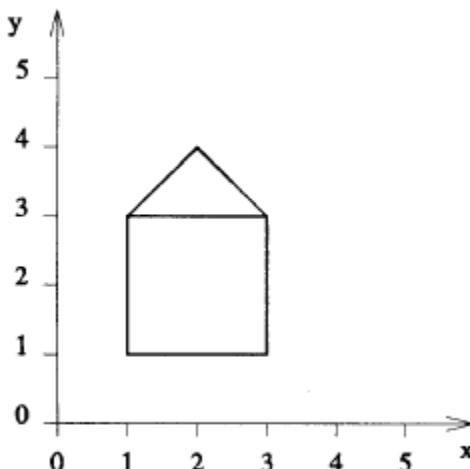
13

14 Další základní geometrickou operací je změna měřítka.

15

16 **4.1.3. Změna měřítka (scaling)**17 Operace změny měřítka je operací, která umožňuje zvětšení či zmenšení, ať už proporcionální v každé
18 ose nebo neproporcionální. Je opět nutné zdůraznit, že operace má počátek souřadného systému
19 jako referenční bod.

20



21

Obr.4.3: Operace změny měřítka

22

23

1 Operace změny měřítka je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \mathbf{x}' = \mathbf{S}(S_x, S_y)\mathbf{x} \quad (4.9)$$

2 kde S_x , resp. S_y určují změnu měřítka v ose x , resp. y .

3 Obecně $\det(\mathbf{S}) \neq 1$, neboť $\det(\mathbf{S}) = S_x S_y$.

4

5 Inverzní operace změny měřítka

6 Inverzní operace změny měřítka je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \mathbf{x}' = \mathbf{S}^{-1}(S_x, S_y)\mathbf{x} = \mathbf{S}\left(\frac{1}{S_x}, \frac{1}{S_y}\right)\mathbf{x} \quad (4.10)$$

7 Lze ukázat, že pokud je změna měřítka určena vektorem, který projektivně reprezentuje změnu

8 měřítka ve tvaru $[S_x, S_y : w_s]^T \triangleq (\frac{s_x}{w_s}, \frac{s_y}{w_s}) = (S_x, S_y)$, pak transformace je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & w_s \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

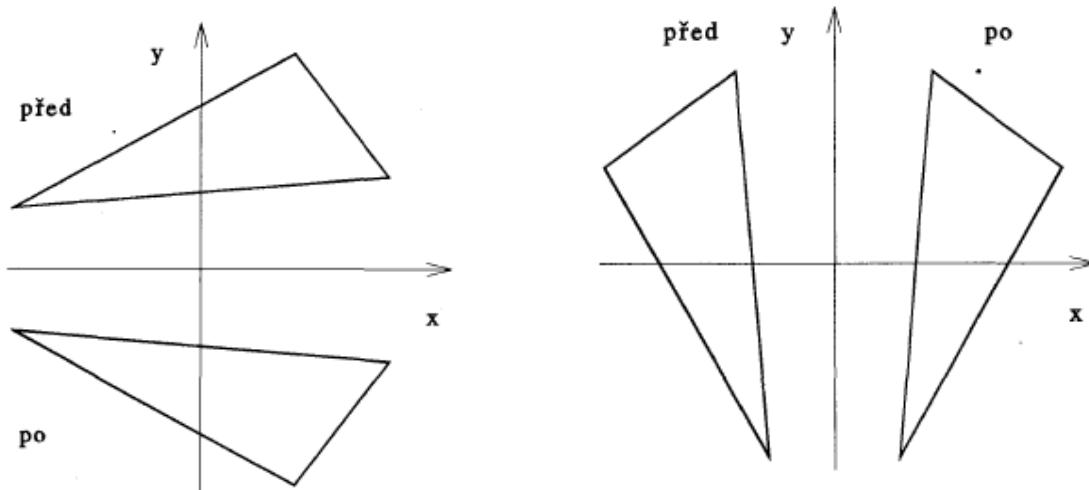
9 přičemž $\det(\mathbf{S}') = s_x s_y w_s$.

10

11 4.1.4. Zrcadlení (mirroring)

12 Zrcadlení podle nějaké osy je opět častou operací. Nejjednodušším případem je ten, kdy osa zrcadlení je totožná s osou souřadného systému.

14



Obr.4.4: Operace zrcadlení podle základní osy

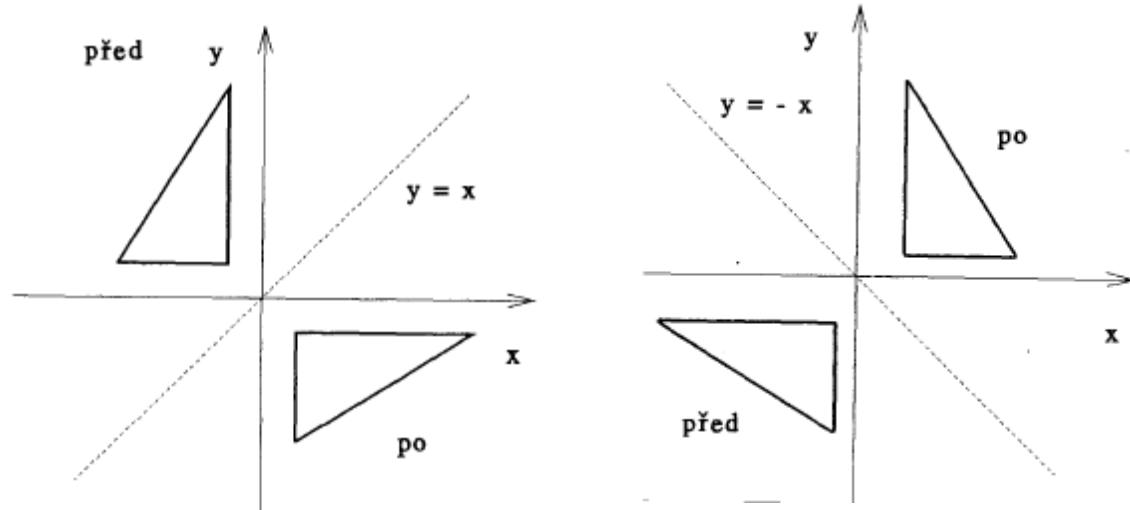
15

16 Transformační vztahy jsou pak určeny takto:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (4.11)$$

17

- 1 Osou zrcadlení může být i obecná osa. Uvažme opět jednoduchý případ, kdy osou je přímka $y = x$,
 2 resp. $y = -x$.
 3



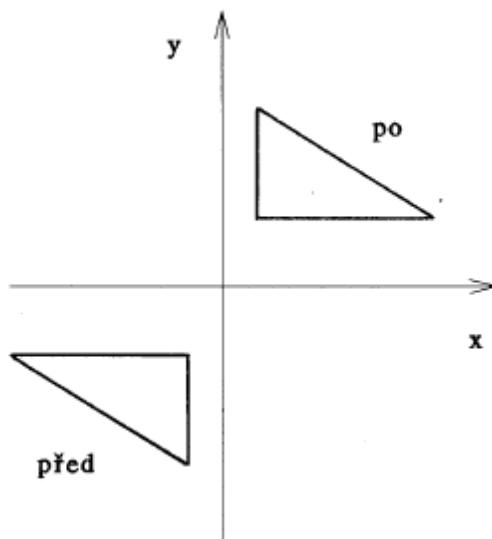
Obr.4.5: Operace zrcadlení podle přímky

- 4 Transformační vztahy jsou pak určeny takto:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (4.12)$$

5

- 6 Posledním typem zrcadlení je zrcadlení vzhledem k počátku, viz Obr.4.6.



Obr.4.6: Operace zrcadlení podle počátku

7

8

9

- 10 Transformační vztahy jsou pak určeny takto:

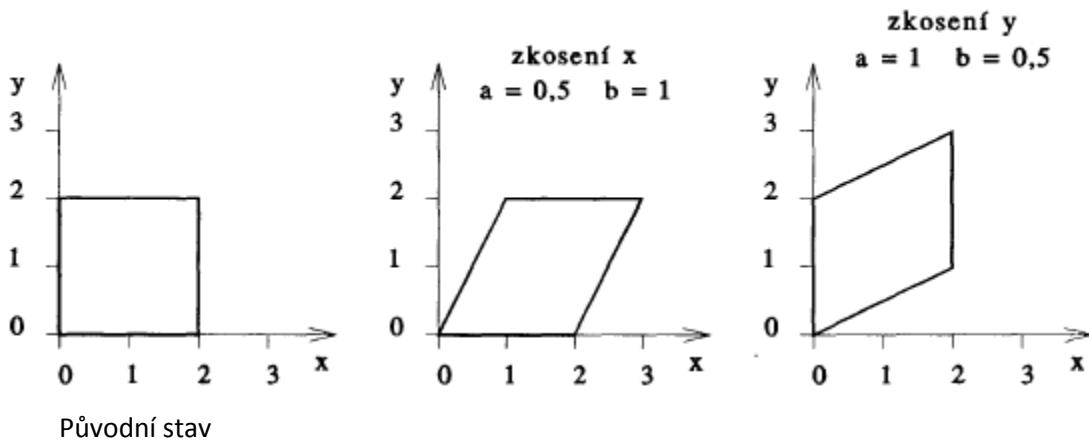
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (4.13)$$

11

- 12 Poslední „základní“ geometrickou transformací, kterou zde uvedeme, je zkosení.

1 **4.1.5. Zkosení (shearing)**

2



Obr.4.7: Operace zkosení

3 Transformační vztahy jsou pak určeny:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (4.14)$$

4

5

6 **Příklad**

7 Odvoděte vztahy pro geometrickou transformaci, která je dána maticí:

$$H_{xy}(a, b) = \begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.15)$$

8

9 a výsledek nakreslete pro $a = 0.5$ a $b = 0.7$ pro situaci z Obr.4.7.a.

10

11 **Příklad**12 Jak bude vypadat matice transformace pro zrcadlení pro osu $y + 2x = 0$?

13

14

15

16

17 Až dosud byly ukázány základní geometrické operace. Vedle těchto operací jsou ještě složitější
18 operace, které jsou vyjádřitelné jako kompozice základních operací. Tento postup se označuje
19 pojmou „řetězení operací“ (concatenation).

20

21

4.2. Řetězení transformací

Řetězení geometrických transformací slouží k realizaci složitějších transformací. Představme si, že máme n po sobě jdoucích transformací popsaných transformačními maticemi \mathbf{Q}_i , $i = 1, \dots, n$. Jednotlivé transformační kroky pak jsou:

$$\mathbf{x}_1 = \mathbf{Q}_1 \mathbf{x} \quad \mathbf{x}_2 = \mathbf{Q}_2 \mathbf{x}_1 \quad \dots \quad \mathbf{x}_n = \mathbf{Q}_n \mathbf{x}_{n-1} \quad (4.16)$$

Celková transformace je pak dána:

$$\mathbf{x}_n = \mathbf{Q}_n \dots \mathbf{Q}_2 \mathbf{Q}_1 \mathbf{x} = \mathbf{Q} \mathbf{x} \quad (4.17)$$

přičemž $\mathbf{Q} = \mathbf{Q}_n \dots \mathbf{Q}_2 \mathbf{Q}_1$.

Připomeňme, že *násobení matic není komutativní*, tj. obecně platí že $\mathbf{AB} \neq \mathbf{BA}$.

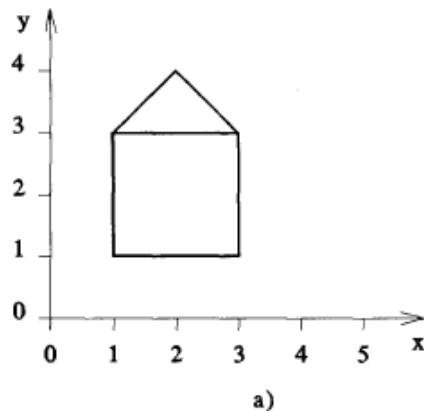
V praxi se všechny elementární transformace \mathbf{Q}_i realizující danou složitější transformaci akumuluje do výsledné matice \mathbf{Q} , která pak reprezentuje celkovou transformaci.

Pro názornost uvedeme jednoduché příklady, a to:

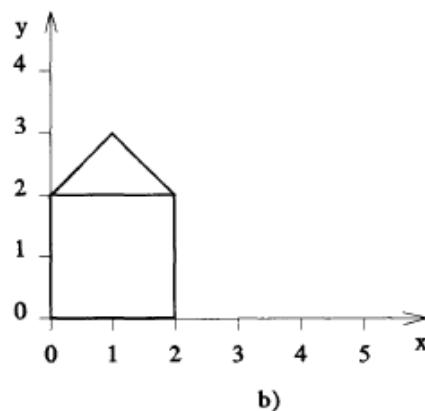
- otočení objektu okolo daného bodu, který je obecně odlišný od počátku souřadného systému
- relativní změna měřítka

Řetězení transformací se též využívá v transformaci „Okno-pohled“, viz kap.4.3 (Window - Viewport transformace).

Příklad 1 - Otočení objektu okolo daného bodu

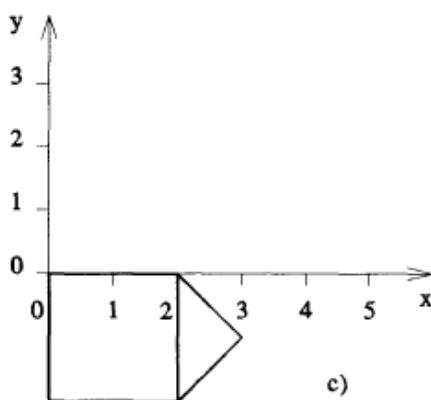


Původní stav



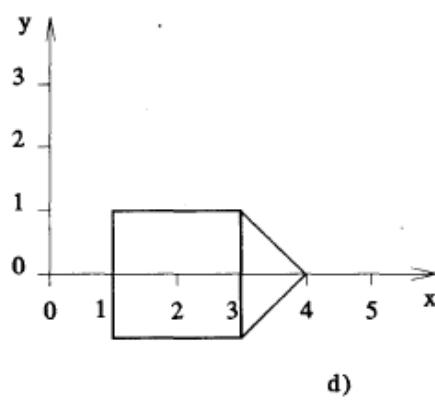
b)

Po posunutí referenčního bodu do počátku



c)

Po otočení o požadovaný úhel



d)

Po posunutí referenčního bodu do původní pozice

Obr.4.8: Řetězení operací

- 1 Daná úloha se sestává z následujících kroků, které jsou aplikovány na všechny body objektu:
- 2 • posuv tak, aby referenční bod rotace (a, b) , tj. v našem případě levý dolní roh objektu, byl
3 v počátku
- 4 • pootočení o požadovaný úhel φ , v našem případě o úhel $\varphi = -\pi/2$
- 5 • posunutí referenčního bodu do původní pozice

6 Nyní můžeme specifikovat jednotlivé transformační kroky:

$$\mathbf{x}' = \mathbf{T}(-a, -b)\mathbf{x} \quad \mathbf{x}'' = \mathbf{R}(\varphi)\mathbf{x}' \quad \mathbf{x}''' = \mathbf{T}(a, b)\mathbf{x}'' \quad (4.18)$$

- 7
- 8 Výsledná transformační matice \mathbf{Q} je dána vztahem:

$$\mathbf{Q} = \mathbf{T}(a, b) \mathbf{R}(\varphi) \mathbf{T}(-a, -b) = \begin{bmatrix} \cos\varphi & -\sin\varphi & -a\cos\varphi + b\sin\varphi + a \\ \sin\varphi & \cos\varphi & -a\sin\varphi - b\cos\varphi + b \\ 0 & 0 & 1 \end{bmatrix} \quad (4.19)$$

9 Po dosazení hodnot pro úhel $\varphi = -\pi/2$ dostáváme:

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 & -b + a \\ -1 & 0 & -a + b \\ 0 & 0 & 1 \end{bmatrix} \quad (4.20)$$

10 Vidíme, že výsledná matice \mathbf{Q} má vlastně strukturu se submaticemi, resp. vektory, a to:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_R & \mathbf{Q}_T \\ \mathbf{Q}_P & 1 \end{bmatrix} \quad (4.21)$$

11 přičemž:

- 12 • \mathbf{Q}_R reprezentuje vlastně rotaci,
- 13 • \mathbf{Q}_T reprezentuje posuv (translaci)
- 14 • \mathbf{Q}_P , která je zatím nulová, reprezentuje např. perspektivní projekci z E^3 do E^2 , viz kap. 5
15 (Projekce).

16 .

17

18 Poznámka

19 V případě řádkové vektorové notace, tj. $\xi = [x, y, z: w]$, je nutné mít na paměti, že zřetězená
20 transformace je definována:

$$\xi' = \xi \Omega_1 \dots \Omega_n \quad (4.22)$$

21 kde: $\Omega_i = \mathbf{Q}_i^T$, $i = 1 \dots n$. To znamená, že pořadí matic je obrácené a matice jsou transponované.

22

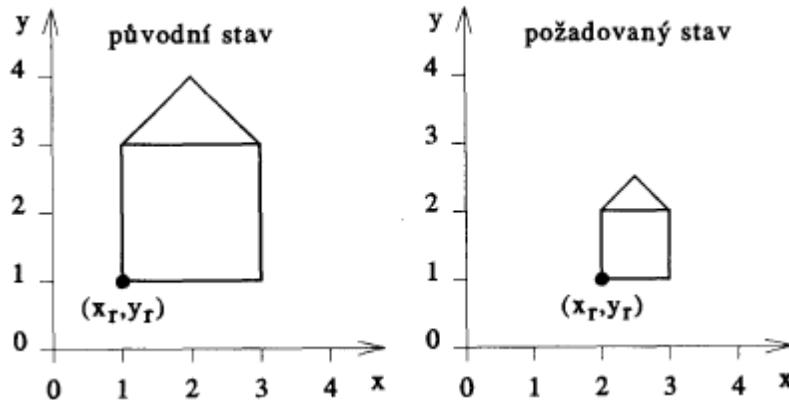
23 Dalším příkladem je relativní změna měřítka

24

1 **Příklad 2 - Relativní změna měřítka**

2 V některých případech může být i změna měřítka záludnou operací. Uvažme jednoduchý objekt, který
 3 chceme zmenšit v poměru 1:2 a referenční bod (x_r, y_r) posunout do nové pozice. Je zřejmé, že
 4 pokud vynásobíme souřadnice vrcholů transformační maticí $S\left(\frac{1}{2}, \frac{1}{2}\right)$, dostaneme výsledek nesprávný,
 5 neboť se posune i referenční bod do pozice $\left(\frac{1}{2}, \frac{1}{2}\right)$.

6



7 Obr.4.9: Změna měřítka s referenčním bodem

8

9 Takže nyní můžeme specifikovat jednotlivé transformační kroky:

- 10 • posuv referenčního bodu do počátku
 11 • změna měřítka
 12 • posuv referenčního bodu do nové pozice

$$x' = T(-a, -b)x \quad x'' = S(S_x, S_y)x' \quad x''' = T(a', b')x'' \quad (4.23)$$

13

14 Výsledná transformační matice Q je dána vztahem:

$$Q = T(a', b') S(S_x, S_y) T(-a, -b) \quad (4.24)$$

15 kde (a, b) je původní pozice referenčního bodu, (a', b') je nová pozice referenčního bodu.

16

17 V předchozím bylo ukázáno, jak se jednotlivé geometrické transformace spojují tak, aby bylo možné
 18 realizovat složité geometrické transformace.

19

20

21 Jednou z významných transformací je transformace Window – Viewport (okno – pohled).

22

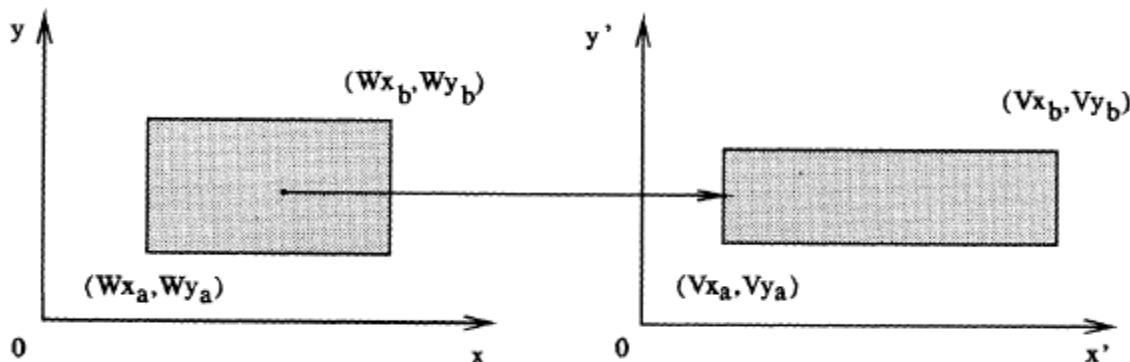
4.3.Window - Viewport transformace

Transformace Window – Viewport (okno – pohled) je velmi specifickou a velmi často používanou.

Tato transformace se obecně sestává z více kroků, a to:

- změna měřítka a změna pozice referenčního bodu
- a volitelným odříznutím částí, které nejsou ve specifikované oblasti. Odříznutí části objektů, které nejsou v této oblasti, se nazývá „ořezávání“, resp. „clipping“ a tato operace je popsána v kap.6 (Metody ořezávání v E^2 a E^3 a operace s n-úhelníky)

Tato transformace zajišťuje ať už přímo, nebo nepřímo vlastně transformaci ze souřadného systému ve kterém je zpracovávaný objekt definován do souřadného systému daného výstupního média.



Obr.4.10: Transformace Window-Viewport

Transformace je složena takto:

- posuv bodů okna tak, že referenční bod Wx_a je v počátku – transformace T_1
- změna měřítka tak, aby se interval (Wx_a, Wx_b) transformoval na interval (Vx_a, Vx_b) – transformace S
- posuv bodů pohledu tak, že referenční bod je v pozici Vx_a – transformace T_2

Celková transformace je pak určena transformační maticí:

$$Q = T_2 S T_1 \quad (4.25)$$

Jednotlivé transformace pak jsou určeny takto:

$$T_1 = \begin{bmatrix} 1 & 0 & -Wx_a \\ 0 & 1 & -Wy_a \\ 0 & 0 & 1 \end{bmatrix} \quad T_2 = \begin{bmatrix} 1 & 0 & Vx_a \\ 0 & 1 & Vy_a \\ 0 & 0 & 1 \end{bmatrix} \quad (4.26)$$

$$S = \begin{bmatrix} (Vx_b - Vx_a)/(Wx_b - Wx_a) & 0 & 0 \\ 0 & (Vy_b - Vy_a)/(Wy_b - Wy_a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.27)$$

a celková matice Q je určena:

$$Q = \begin{bmatrix} \alpha & 0 & \gamma \\ 0 & \beta & \delta \\ 0 & 0 & 1 \end{bmatrix} \quad (4.28)$$

kde:

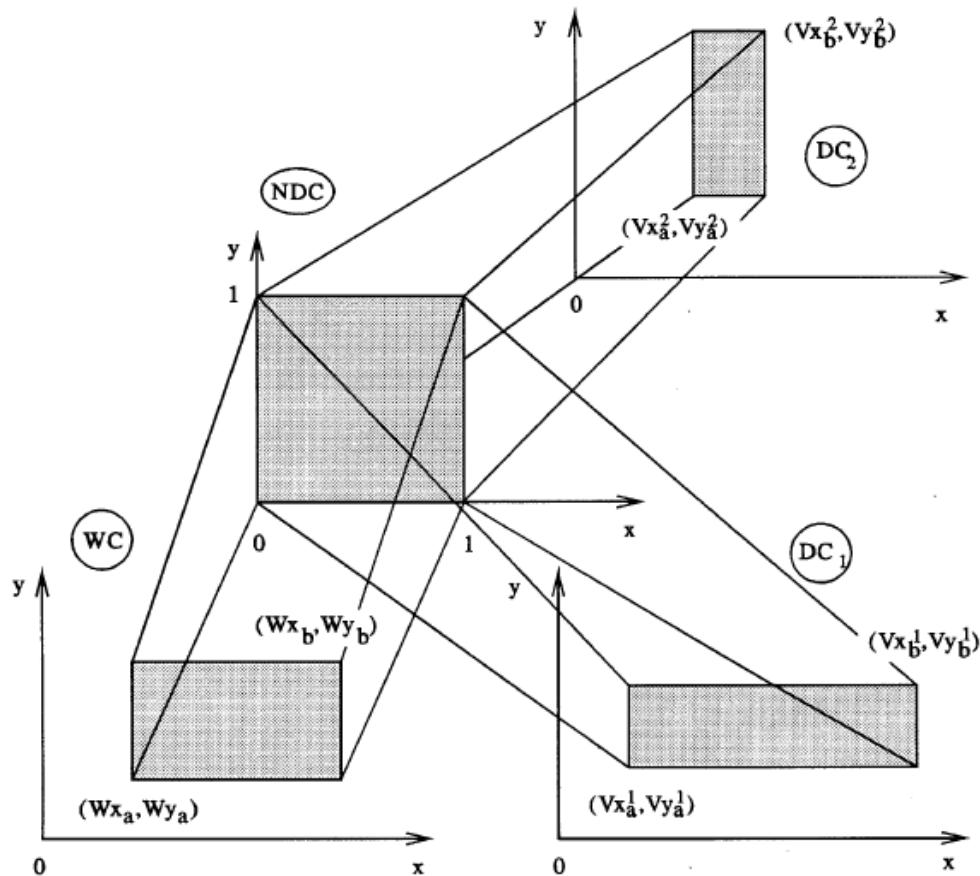
$$\begin{aligned} \alpha &= (Vx_b - Vx_a)/(Wx_b - Wx_a) & \gamma &= Vx_a - \alpha Wx_a \\ \beta &= (Vy_b - Vy_a)/(Wy_b - Wy_a) & \delta &= Vy_a - \beta Wy_a \end{aligned}$$

Je zřejmé, že pokud některé části vykreslované scény přesahují velikost výstupního okna (Vx_a, Vx_b) , resp. média apod., je nutné, resp. vhodné, tyto části odříznout.

Důležitým konceptem je „Normalizovaný souřadný systém“.

1 4.4. Normalizovaný souřadný systém

2 Koncept normalizovaného souřadného systému NDC (Normalized Device Coordinates) je klíčový
 3 nejen pro oblast počítačové grafiky.



4 Obr.4.11: Koncept Normalizovaného souřadného systému

5 V systémech počítačové grafiky jsou používány různé souřadné systémy v závislosti na kontextu:

- 6 • souřadný systém zařízení **DC** (Device Coordinates) – tento souřadný systém pracuje většinou
 7 ve fyzických jednotkách daného konkrétního zařízení. Typickým příkladem je souřadný systém
 8 obrazovky, kdy jednotkou je pixel a vertikální osa je orientována směrem dolů
- 9 • souřadný systém **WC** (World Coordinates) – v tomto souřadném systému je reprezentován
 10 objekt. Pozice bodu může při stejně numerické hodnotě znamenat různě velké hodnoty,
 11 vzdálenosti apod. Navíc hodnoty mohou být v různé reprezentaci
- 12 • souřadný systém normalizovaných souřadnic **NDC** (Normalized Device Coordinates) -
 13 umožňuje efektivní transformaci mezi WC a DC souřadnými systémy. Za mapování mezi WC a
 14 NDC je zodpovědná aplikace, např. GIS systém, zatímco za mapování mezi NDC a DC je pak
 15 zodpovědný ovladač daného fyzického zařízení

16 Je tedy zřejmé, že jde o vícenásobnou aplikaci transformace Window – Viewport, která byla již
 17 uvedena dříve. Zavedením konceptu NDC se umožnilo snadné připojování vstupních a výstupních
 18 periférií k aplikačním programům.

20 Až dosud byly předloženy geometrické transformace v rovině, tj. v prostoru E^2 . Nicméně prostor, ve
 21 kterém se pohybujeme, je třírozměrný, tj. E^3 .

22 *Je tedy otázkou, jak jsou definovány geometrické transformace v E^3 .*

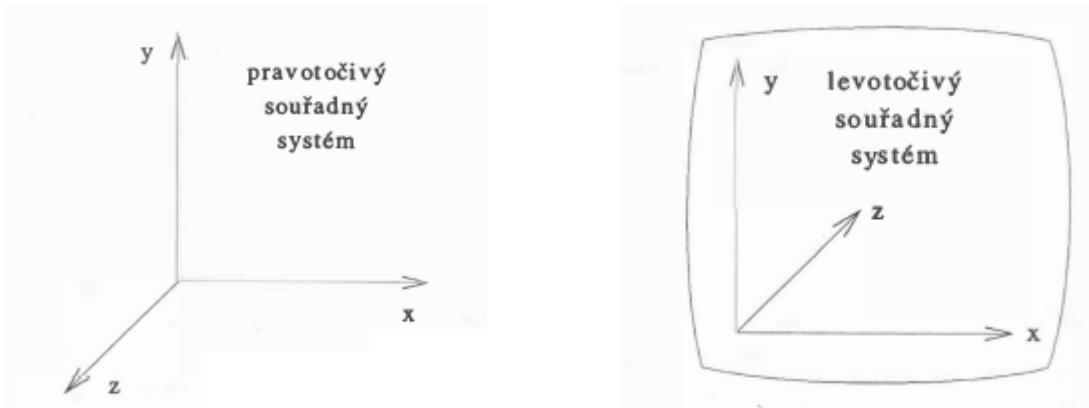
4.5. Základní transformace v E^3

V případě dvourozměrného systému je orientace os systému vžitá, tj. osa x je horizontální s orientací doprava, osa y je vertikální s orientací nahoru, v případě obrazových aplikací je orientace osy y směrem dolu. V případě třírozměrného kartézského souřadného systému je situace poněkud komplikovanější, neboť jsou dvě orientace souřadného systému, a to:

- *pravotočivá*, která bude implicitně používána pro geometrické operace v WC souřadném systému
- *levotočivá*, která bude použita pro reprezentaci pozice pozorovatele, resp. kamery.

Určení orientace souřadného systému pomocí pravé dlaně – pokud osa x protíná dlaň a prsty mají stejný směr jako osa y a palec ukazuje ve směru osy z , pak daný souřadný systém je pravotočivý. V opačném případě je levotočivý.

Poznamenejme, že některé publikace, zejména pokud používají řádkovou notaci pro vektory, používají implicitně levotočivý souřadný systém. Dále uvedené vztahy pak používají transponovanou notaci.



Obr.4.12: Pravotočivý a levotočivý souřadný systém

Nyní uveďme základní geometrické transformace v E^3 , které jsou vlastně jen přímým prostým rozšířením rovinných operací dříve definovaných. Pochopitelně vektor x nyní obsahuje navíc složku pro osu z :

$$\mathbf{x} = [x, y, z; w]^T$$

Operace posunutí

Operace posunutí o (A, B, C) je určena vztahem:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & A \\ 0 & 1 & 0 & B \\ 0 & 0 & 1 & C \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \quad \mathbf{x}' = \mathbf{T}(A, B, C)\mathbf{x} \quad (4.29)$$

Operace posuvu je nyní reprezentována maticí 4×4 , místo matice 3×3 .

Operace změny měřítka

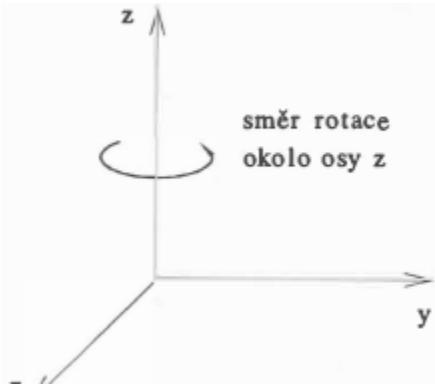
Operace změny měřítka je obdobná, a to:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \quad \mathbf{x}' = \mathbf{S}(S_x, S_y, S_z)\mathbf{x} \quad (4.30)$$

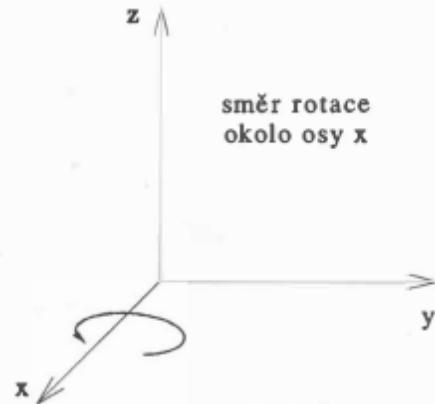
1 **Operace rotace**

2 Operace rotace je poněkud složitější, neboť v rovinném případě jsme rotovali vlastně v rovině xy , tj.
 3 okolo „virtuální“ osy z , která směřovala vzhůru, tj. k pozorovateli. V případě třírozměrném máme 3
 4 základní osy, okolo kterých rotaci můžeme definovat, resp. 3 základní roviny, ve kterých můžeme
 5 rotaci realizovat.

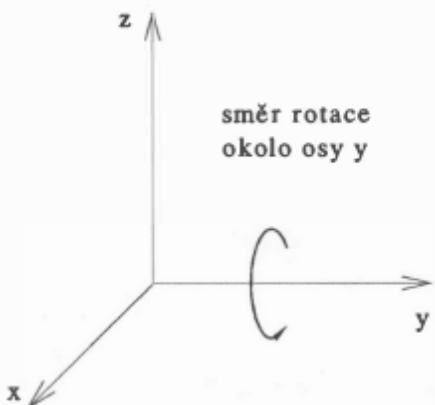
6 Z faktického hlediska je jedno, zda mluvíme o rotaci v rovině nebo okolo osy. Nicméně po formální
 7 stránce je vhodnější mluvit o rotaci v rovině xy , yz nebo zx , neboť znaménko *minus* v matici bude
 8 vždy u řádku, který odpovídá 1. písmenku v notaci rotace. Pořadí písmenek v notaci není libovolné,
 9 neboť je dánou pravotočivostí souřadného systému. Např. rotace v rovině xz místo zx by vlastně byla
 10 rotací v levotočivém souřadném systému, tedy vlastně rotací s opačným úhlem.

Rotace v rovině xy , tj. okolo osy z

$$\mathbf{R}(\varphi)_{xy} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.31)$$

Rotace v rovině yz , tj. okolo osy x

$$\mathbf{R}(\varphi)_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.32)$$

Rotace v rovině zx , tj. okolo osy y

$$\mathbf{R}(\varphi)_{zx} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.33)$$

1 Inverzní operace

2 Konstrukce inverzních operací v E^3 je analogická konstrukci inverzních operací v E^2 .

3

4

5 Upozornění

6 Mnoho publikací používá řádkovou notaci pro vektory nebo levotočivý souřadný systém jako základní
7 souřadný systém. V tomto případě jsou transformační matice transponované a pořadí transformací je
8 opačné, tj. nikoliv zprava doleva, ale zleva doprava. Nelze tedy „míchat“ transformace ze zdrojů
9 s řádkovou notací a se sloupcovou notací jednoduše dohromady.

10

11 Rotace je vlastně jednou z nejdůležitějších operací, neboť zobrazovaný objekt chceme prohlížet na
12 daném výstupním zařízení, např. obrazovce, a pohyb v E^3 budeme zřejmě ovládat nějakým
13 zařízením, např. myší, které je vlastně zařízením pracujícím v E^2 . Takže vedle rotace budou zapotřebí
14 ještě další transformace apod.

15

16 V praxi se však vyskytuje obecnější případ, a to *rotace okolo dané osy v prostoru E^3* . Toto je už
17 složitější geometrická transformace, která je dána jako kompozice základních geometrických
18 transformací. Stejně jako u rovinných geometrických transformací lze použít řetězení geometrických
19 transformací k získání složitých geometrických transformací.

20

21 Vidíme, že základní geometrické transformace jsou poměrně jednoduché z hlediska matematického
22 popisu. Pro realizaci složitějších geometrických transformací se opět používá zřetězení geometrických
23 transformací, které bylo uvedeno v kap.4.2 (Řetězení transformací).

24

25

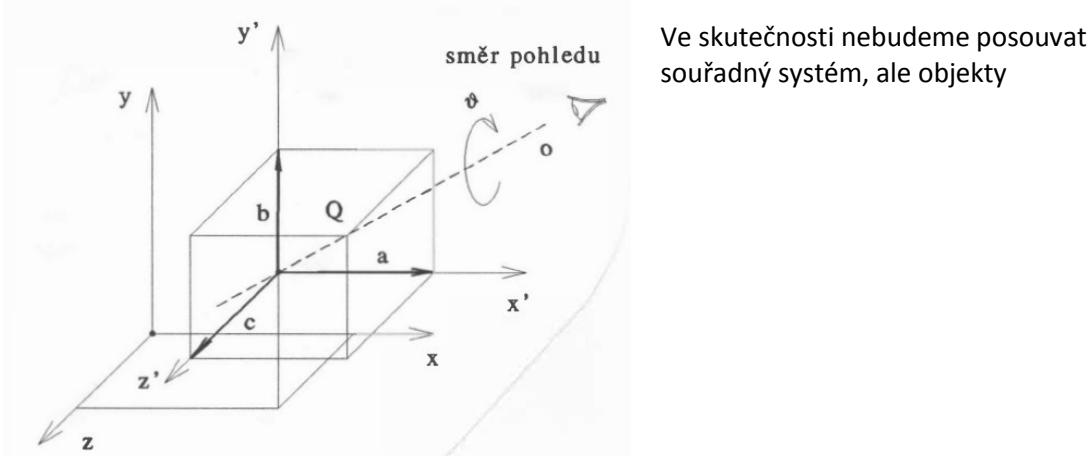
4.6. Rotace okolo dané osy v E^3

Rotace okolo dané osy v E^3 již není triviální geometrickou transformací. Uvažme jednoduchou situaci, a to:

- je dána osa rotace o bodem a směrovým vektorem v E^3 , tj.

$$\mathbf{X}(t) = \mathbf{X}_A + \mathbf{S}t \quad \mathbf{S} = [a, b, c]^T \quad (4.34)$$

- všechny body ve scéně se mají pootočit o úhel ϑ v naznačeném směru



Obr.4.14: Rotace podle obecné osy v prostoru

Tato složená transformace má několik základních kroků (uvedená posloupnost transformace rotací není jedinou možnou), a to:

- posuv souřadného systému do bodu \mathbf{X}_A , resp. všechny body se posunou odpovídajícím způsobem do geometricky ekvivalentní pozice. Toto je nutné, neboť rotace má počátek souřadného systému za referenční bod
- rotace tak, aby osa rotace o ležela v nějaké základní rovině, např. v rovině xy , tj. uděláme rotaci v rovině zx
- rotace tak, aby osa rotace o byla totožná s nějakou osou, např. s osou x
- rotace o úhel ϑ
- inverzní operace provedené před rotací o úhel ϑ v opačném pořadí

Takže dostáváme matici \mathbf{Q} kumulované transformace:

$$\mathbf{Q} = \mathbf{T}^{-1} \mathbf{R}_{zx}^{-1} \mathbf{R}_{xy}^{-1} \mathbf{R}(\varphi) \mathbf{R}_{xy} \mathbf{R}_{zx} \mathbf{T}$$

Podíváme-li se však pozorněji, např. na matici \mathbf{R}_{yz}

$$\mathbf{R}_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{\sqrt{b^2 + c^2}} & \frac{-b}{\sqrt{b^2 + c^2}} & 0 \\ 0 & \frac{b}{\sqrt{b^2 + c^2}} & \frac{c}{\sqrt{b^2 + c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.35)$$

Vidíme, že transformace není stabilní, pokud $\sqrt{b^2 + c^2} \rightarrow 0$, nebo je nepřesná pokud $b^2 \gg c^2$. Navíc další nepřesnosti vznikají násobením matic. V následujícím výkladu si ukážeme, jak transformace může být elegantně řešena s podstatně lepší výpočetní přesností a stabilitou.

1 4.7.Transformace rotace v E³ okolo osy založená na rotaci vektorů

2 Uveďme nyní zcela jiný přístup ke geometrickým transformacím, který je založen na transformaci
3 vektorů místo bodů.

5 Je dána osa rotace procházející počátkem souřadného systému a úkolem je pootočení scény o úhel
6 φ . Pro daný bod X jde tedy o rotaci na rovině v obecné poloze, která má normálový vektor n ,
7 přičemž normálový vektor n je normalizován, tj. $\|n\| = 1$

9 Transformace je pak dána vztahem:

$$10 \quad X' = X \cos\varphi + (1 - \cos\varphi)(n^T X) \cdot n + (n \times X) \sin\varphi \quad (4.36)$$

$$11 \quad Q = I \cos\varphi + (1 - \cos\varphi)(n \otimes n) + W \sin\varphi \quad (4.37)$$

12 kde: $n \otimes n = n \cdot n^T$ je matice

13 (operace \otimes se nazývá tenzorový součin vektorů).

14 V Eukleidovském prostoru vektor n musí být normalizován

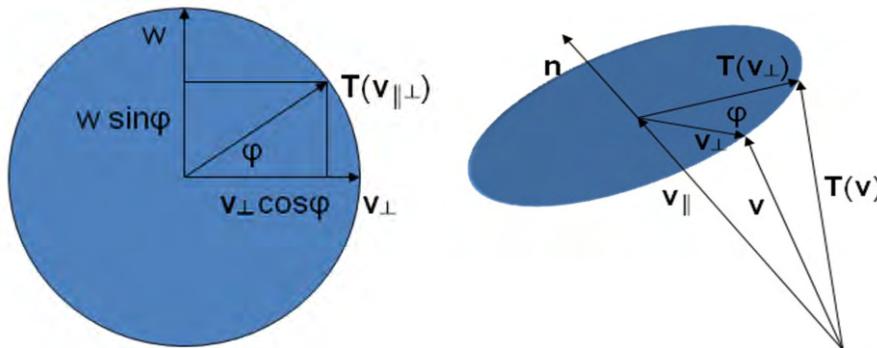
15 a matice W je definována jako: $Wv = w \times v$, tj.:

$$16 \quad W = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} \quad (4.38)$$

17 Pro odvození se použije „trik“, a to:

$$18 \quad v = v_{||} + v_{\perp} \quad (4.39)$$

19 tedy „rozdelení“ vektoru v na dva vektory navzájem na sebe kolmé.



Obr.4.15: Rotace podle obecné osy

20 Podrobněji viz 4.11 (Geometrické transformace - vektorově orientované formulace) nebo:

- 21 • Miller,J.R.: Vector Geometry for Computer Graphics, IEEE Computer Graphics and Applications, pp.66-73, May/June 1999 ([CLICK PDF](#) off-line)
- 22 • Miller,J.R.: Applications of Vector Geometry for Robustness and Speed, IEEE Computer Graphics and Applications, pp.68-73, July/August/May/June 1999 ([CLICK PDF](#) off-line)

4.8.Transformace přímek a rovin

Dosud předložené geometrické transformace se týkaly transformace bodů. Nicméně počítačová grafika používá také přímky a roviny. Často vzniká otázka, jak se změní rovnice přímky nebo roviny, pokud se body, které je definují, transformují. Typickou úlohou je problém stínování plochy, pokud se s danou plochou manipuluje, např. otáčí, mění se měřítko, které je v různých směrech různé atd. V předešlém výkladu bylo ukázáno, že *normála není vlastně vektor, ale bivektor*, neboť výsledek vektorového součinu dvou vektorů je orientovaná plocha.

Z předchozího je známo, že duální primitiva jsou:

	E^2	E^3
	$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2$	$\boldsymbol{\rho} = \mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3$
Duální problém	$\mathbf{x} = \mathbf{p}_1 \times \mathbf{p}_2$	$\mathbf{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$

Tab xxx

Pokud se body definující přímku nebo rovinu transformují podle vztahu:

$$\mathbf{x}' = \mathbf{T}\mathbf{x} \quad (4.40)$$

vzniká otázka, jaká bude implicitní reprezentace přímky, resp. roviny. Lze ukázat, že přímka p po transformaci bodů, které ji definují, bude určena vektorem \mathbf{p}' takto:

$$\mathbf{p}' = (\mathbf{T}\mathbf{x}_1) \times (\mathbf{T}\mathbf{x}_2) = \det(\mathbf{T})(\mathbf{T}^{-1})^T \mathbf{p} \triangleq (\mathbf{T}^{-1})^T \mathbf{p} \quad (4.41)$$

kde \triangleq znamená projektivní ekvivalence, neboť rovnice přímky je implicitní a je ji možné násobit libovolnou nenulovou hodnotou a pozice přímky se nezmění. Takže transformovaná přímka p je určena vektorem:

$$\mathbf{p}' = (\mathbf{T}^{-1})^T \mathbf{p} = [a', b': c']^T \quad (4.42)$$

Obdobně pro případ roviny, jejíž definiční body byly transformovány, dostáváme obdobný vztah:

$$\boldsymbol{\rho}' = (\mathbf{T}\mathbf{x}_1) \times (\mathbf{T}\mathbf{x}_2) \times (\mathbf{T}\mathbf{x}_3) = \det(\mathbf{T})(\mathbf{T}^{-1})^T \boldsymbol{\rho} \triangleq (\mathbf{T}^{-1})^T \boldsymbol{\rho} \quad (4.43)$$

a transformovaná rovina ρ je určena vektorem:

$$\boldsymbol{\rho}' = (\mathbf{T}^{-1})^T \boldsymbol{\rho} = [a', b', c': d']^T \quad (4.44)$$

Závěr

Z výše uvedeného je podstatné to, že:

- při geometrických transformacích přímky nebo roviny je nutné normálu násobit maticí $(\mathbf{T}^{-1})^T$, což je matice obecně různá od matice \mathbf{T} , tj. obecně $(\mathbf{T}^{-1})^T \neq \mathbf{T}$
- není nutné transformovat body a následně počítat koeficienty transformované přímky, či roviny, lze určit koeficienty transformované přímky, či roviny přímo, pokud již byly dříve určeny

Geometrické transformace při použití vektorové reprezentace jsou uvedeny v kap.4.11 (Geometrické transformace - vektorově orientované formulace)

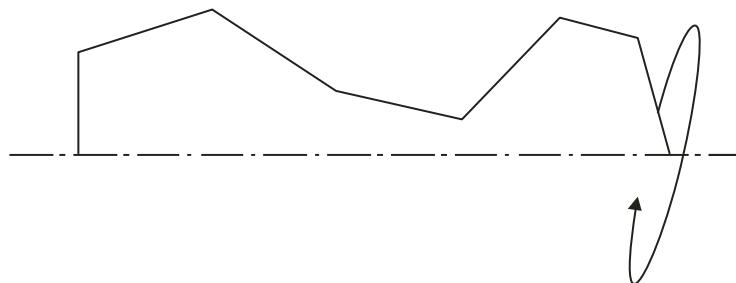
V předchozí části byly ukázány základní principy „klasických“ geometrických transformací se základními geometrickými primitivy.

4.9. Generace rotačního tělesa

Jednou z častých úloh je generace povrchu rotačního objektu daného obrysem, např. lomenou čárou, a jeho následné zpracování a zobrazení.

Úkolem je vytvořit plášť rotačního objektu s daným profilem a s osou rotace x . Tato úloha už není zcela triviální, neboť vedle generace vlastních bodů a jejich transformací, je nutné ještě zvážit:

- datovou strukturu, která se použije
- způsob reprezentace povrchu pomocí trojúhelníků, které grafické akcelerátory zpracovávají
- otázkou vykreslování, tj. drátěný model, stínovaný s konstantním stínováním, resp. stínováním Gouraud, viz kap.14.2 (Metody stínování – lokální metody). Je nutné určit normály generovaných trojúhelníků, resp. normály ve vrcholech generovaných trojúhelníků



Obr.4.16:Generování rotačně souměrného tělesa

Rotační tělesa jsou velmi častá. Rotační symetrie totiž umožnuje v mnoha inženýrských aplikacích podstatné snížení výpočetní složitosti, neboť umožnuje snadnější řešení rotačně symetrického problému, např. elektro-magnetického pole, v E^2 místo E^3 , tj. dojde ke snížení dimenze.

Transformace pro rotaci okolo osy x , tj. v rovině yz , pak je definována:

$$\begin{aligned} x' &= x \\ y' &= y \cos\varphi - z \sin\varphi \\ z' &= y \sin\varphi + z \cos\varphi \end{aligned} \tag{4.45}$$

Poznámka

V tomto příkladě je jednodušší určovat normálu ve vrcholech, než normálu generovaných trojúhelníků. Je vhodné tento příklad „dotáhnout“ do konce včetně implementace a zobrazení výsledného tělesa.

4.10. Generace povrchu koule

Dalším často generovaným povrchem je povrch koule. V mnoha aplikacích se použije přímá aplikace sférických souřadnic.

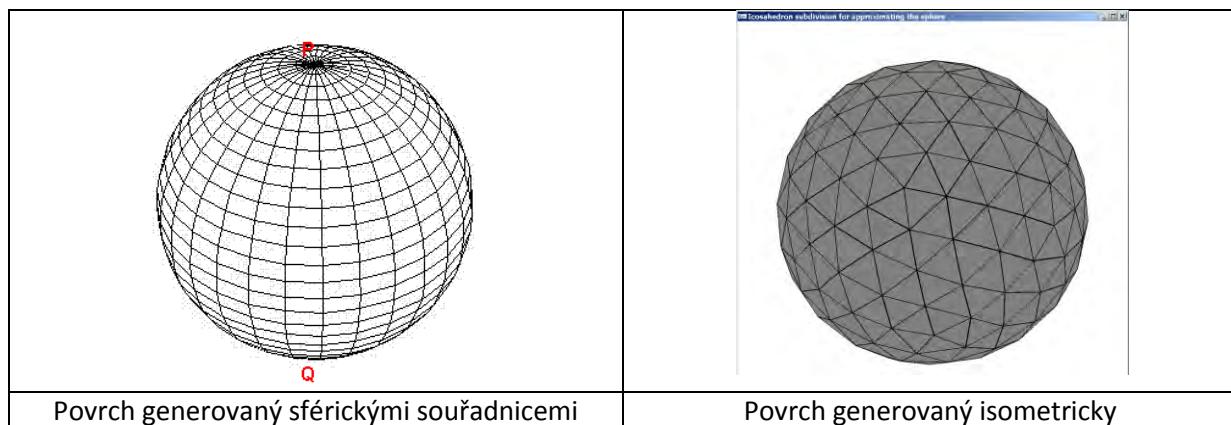
4.10.1. Generování pomocí sférických souřadnic

Body sítě se generují v pravidelné síti parametrů θ a ϕ . Pro generaci bodů použijeme vztahy:

$$x = r \cos \theta \sin \phi \quad y = r \sin \theta \sin \phi \quad z = r \cos \phi \quad (4.46)$$

Daný způsob je jednoduchý, ale je několik vlastností, které je nutno mít na paměti, a to zejména:

- generované čtyřúhelníky nejsou rovinné a „rozbíjejí“ se na trojúhelníky
- generované trojúhelníky nejsou stejně velké
- na „pólech“ jsou degenerované trojúhelníky, neboť pro $\phi = \pm \frac{\pi}{2}$ je generován pouze jeden bod pro všechny úhly θ , tj. generovaný čtyřúhelník degeneruje na trojúhelník
- body sítě, tj. vrcholy trojúhelníků, jsou jednoduše indexovatelné



Obr.4.17: Generace povrchu koule

Zásadní nevýhodou z hlediska aplikací je to, že trojúhelníky jsou různě velké. Existuje však postup, který zajistí, že generované trojúhelníky jsou stejně velké.

4.10.2. Povrch generovaný iso-metricky

Tento postup je založen na tom, že trojúhelník lze rozdělit, a to:

- na 3 trojúhelníky vložením bodu, např. do těžiště. Postupným dělením však vznikají stále štíhlzejší trojúhelníky
- na 4 trojúhelníky tak, že každou hranu rozdělíme na $\frac{1}{2}$ a takto vzniklé body spojíme. V případě, že původní trojúhelník byl *rovnostranný*, i nově vzniklé trojúhelníky budou také rovnostranné, resp. dělením vznikají trojúhelníky podobné. Toto je důležitá vlastnost vhodná v mnoha aplikacích.

Tento způsob podporuje přirozeně hierarchické dělení, resp. adaptivní zjemňování trojúhelníkové sítě, avšak indexace je složitější.

1 **Algoritmus**

2 Do koule vepřeme 2 pyramidy o hraně $r\sqrt{2}$ podstavou k sobě. Jednotlivé hrany pak rozdělíme na $\frac{1}{2}$ a
 3 souřadnice takto vzniklých bodů „dotáhneme“ na povrch koule. Tímto postupem postupně
 4 zjemňujeme trojúhelníkovou síť approximující povrch koule trojúhelníkovou sítí.



Obr.4.18: princip generace isometrického povrchu

7 **Výpočet nových bodů pro dělení hrany**

8 Při generování nových bodů, je postup výpočetně poměrně jednoduchý, neboť bod ξ je určen jako:

$$\xi = (\mathbf{x}_2 + \mathbf{x}_1)/2$$

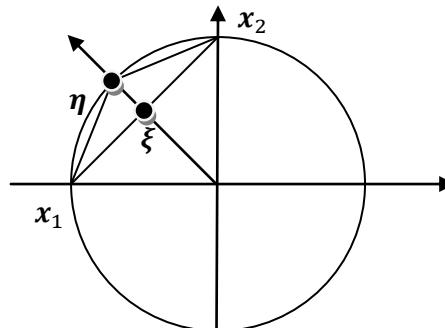
9 Nyní je nutné určit souřadnice bodu η , a to jako průsečík paprsku ξ s povrchem koule se středem
 10 v počátku souřadného systému, který se jeví v průmětu jako kružnice κ .

11 Paprsek z počátku do bodu ξ :
 12 $\mathbf{x}(t) = \xi t$
 13 a kružnice:
 14 $\mathbf{x}^T \mathbf{x} = R^2$

15 Pak řešením dostáváme:

$$\xi^T \xi t^2 - R^2 = 0$$

$$t = R / \sqrt{\xi^T \xi}$$



Obr.4.19: Geometrie řešení

17 Průsečík η je pak určen:

$$\eta = R \frac{\xi}{\sqrt{\xi^T \xi}} = R \frac{\xi}{\|\xi\|}$$

18 Tento postup se opakuje pro všechny hrany již generované trojúhelníkové sítě. Při dalším dělení body
 19 \mathbf{x}_1 a \mathbf{x}_2 již nejsou obecně na osách, ale jsou již v obecné poloze.

20

21 **Úloha 1**

22 Napište algoritmus v pseudokódu, který bude generovat isometricky povrch koule ve formě
 23 podprogramu SPHERE(R, N), kde: R je poloměr koule a N je počet dělení hran. Navrhnete vhodnou
 24 datovou strukturu pro reprezentaci výsledné trojúhelníkové sítě.

25

26 **Úloha 2**

27 Navrhnete datovou strukturu umožňující reprezentaci hierarchie generované trojúhelníkové sítě a
 28 najděte jednoduchý indexační (adresační) algoritmus pro generovanou hierarchickou trojúhelníkovou
 29 sítě.

4.11. Geometrické transformace - vektorově orientované formulace

Geometrické transformace dříve uvedené jsou založeny na transformaci bodů. Bod jako takový je vždy vázán k počátku souřadného systému. Lze ukázat, že dříve uvedené transformace mohou být v určitých případech numericky nestabilní. Alternativním způsobem definování geometrických trasformací je formulace v afinním prostoru, tj. s použitím vektorů (matematicky vzato), které nejsou vázány k počátku souřadného systému. Tento přístup není obecně rozšířen, avšak poskytuje podstatně větší stabilitu. V následujícím výkladu si ukážeme princip a základní geometrické transformace.

Je nutné si vědomit, že:

- *bod* (point) je vlastně pozice v prostoru v daném souřadném systému. Při realizování implementace používáme datovou strukturu jednorozměrné matice, kterou obvykle nazýváme vektorem.
- *vektor* (vector) nemá vlastní pozici, neboť má pouze směr a velikost, tj. není „ukotven“ v souřadném systému. Na rozdíl od bodu, vektory lze sčítat a odečítat.

Pro pochopení následujícího ještě uvedeme operace s vektory:

- *normalizace* vektoru: pokud $\|\mathbf{u}\| = \sqrt{u_x^2 + u_y^2 + u_z^2} = 1$, tj.

$$\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \quad \|\hat{\mathbf{u}}\| = 1 \quad (4.47)$$

- *skalární součin* (dot product): $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos \varphi$, kde φ je úhel vektory \mathbf{u} a \mathbf{v} sevřený
- *vektorový součin* (cross product, někdy též outer product): $\|\mathbf{u} \times \mathbf{v}\| = |\mathbf{u}| |\mathbf{v}| \sin \varphi$, přičemž vektor $\mathbf{u} \times \mathbf{v}$ je kolmý na původní vektory \mathbf{u} a \mathbf{v} . Jde vlastně o bivektor mající vlastnost orientované plochy vektory sevřené.

- *tenzorový součin* (tensor product): $\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_x v_x & u_x v_y & u_x v_z \\ u_y v_x & u_y v_y & u_y v_z \\ u_z v_x & u_z v_y & u_z v_z \end{bmatrix}$.

Jde vlastně o operaci sloupec krát řádek, kde výsledek je matici $\mathbf{Q} = \mathbf{u} \mathbf{v}^T$.

Pro tenzorový součin platí vztahy:

$$(\mathbf{u} \otimes \mathbf{v}) \hat{\mathbf{n}} = (\mathbf{v} \cdot \hat{\mathbf{n}}) \mathbf{u} \quad (4.48)$$

který v dalším použijeme.

$$\mathbf{u} \otimes \mathbf{v} = (\mathbf{v} \otimes \mathbf{u})^T \quad (4.49)$$

a

$$\mathbf{u} \otimes (a\mathbf{v} + b\mathbf{w}) = a(\mathbf{u} \otimes \mathbf{v}) + b(\mathbf{u} \otimes \mathbf{w}) \quad (4.50)$$

Lze dokázat, že pro libovolné vektory \mathbf{c} , \mathbf{d} a \mathbf{e} platí:

$$(\mathbf{c} \cdot \mathbf{d}) \mathbf{e} = (\mathbf{e} \otimes \mathbf{c}) \mathbf{d} = (\mathbf{e} \otimes \mathbf{d}) \mathbf{c} \quad (4.51)$$

Princip dále uvedených geometrických transformací je v zásadě velmi jednoduchý. Transformace jsou formulovány nezávisle na souřadném systému s použitím vektorů a přidáním „fixního“ bodu, pak dochází ke specifikaci transformace pro daný souřadný systém.

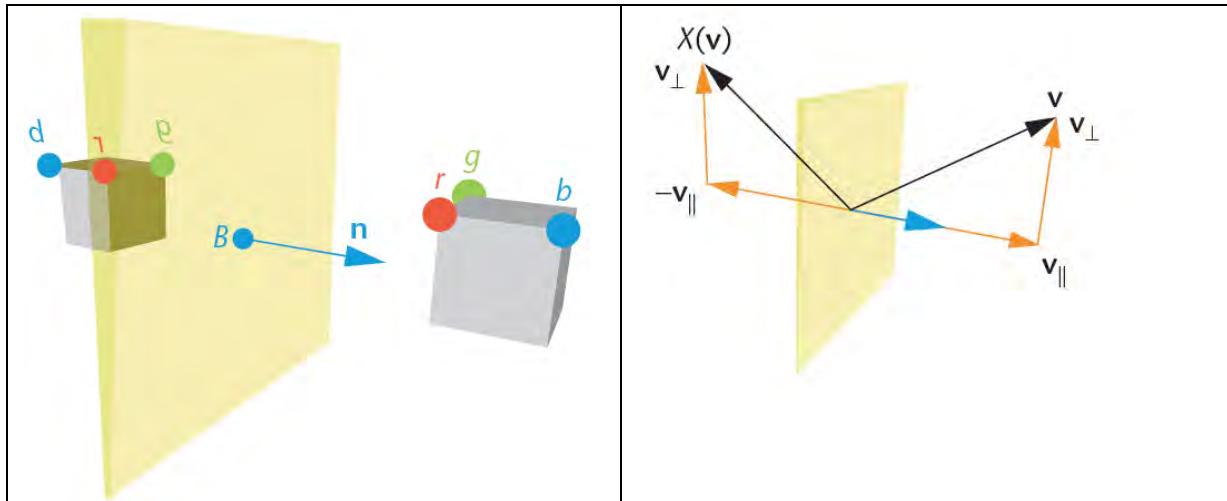
Obecně budeme transformace reprezentovat maticí \mathbf{Q} :

$$\mathbf{Q} = \begin{bmatrix} \mathbf{M}_{3 \times 3} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad (4.52)$$

33

1 **4.11.1. Operace zrcadlení podle roviny v obecné poloze v E^3**

2 Předpokládejme, že je dána rovina v E^3 v libovolné poloze a to jejím bodem x_0 a normalizovaným
3 vektorem \hat{n} , tj. je dána rovina $\rho(x_0, \hat{n})$ a chceme zrcadlit bod x .



4 Obr.4.20: Operace zrcadlení - Courtesy of Miller,J.R.

5 Vektor $v = x - x_0$ můžeme rozdělit na dva na sebe kolmé vektory v_{\parallel} a v_{\perp} tak, že $v = v_{\parallel} + v_{\perp}$, tj. na
6 složku rovnoběžnou a na složku kolmou k normálovému vektoru \hat{n} . Je zřejmé, že vektor v_{\perp} směr
7 nemění a vektor v_{\parallel} je opačného směru, tj. $-v_{\parallel}$, a tyto vektory jsou určeny takto:

$$v_{\parallel} = (\hat{n} \cdot v) \cdot \hat{n} \quad v_{\perp} = v - (\hat{n} \cdot v) \cdot \hat{n} \quad (4.53)$$

10 Pak transformace vektoru v :

$$\begin{aligned} Mv &= M(v_{\parallel} + v_{\perp}) = Mv_{\parallel} + Mv_{\perp} = v_{\perp} - v_{\parallel} = \\ &= v - (\hat{n} \cdot v) \cdot \hat{n} - (\hat{n} \cdot v) \cdot \hat{n} \end{aligned} \quad (4.54)$$

11 Úpravou a s použitím vztahu $(u \otimes v)w = (v \cdot w)u$ dostáváme:

$$Mv = v - 2[(\hat{n} \cdot v) \cdot \hat{n}] = v - 2[\hat{n} \otimes \hat{n} \cdot v] \quad (4.55)$$

12 Pak transformační matice M je určena:

$$M = I - 2 \hat{n} \otimes \hat{n} \quad (4.56)$$

13 Nyní je nutné určit translační složku, tj. vektor t a to dosazením bodu roviny. Protože pozice bodu
14 roviny x_0 se nemění, tj. je stejná před i po transformaci, pak vektor posuvu je určen:

$$t = x_0 - Mx_0 \quad (4.57)$$

16 Je tedy zřejmé, že výše uvedené řešení je:

- 17 • jednoduché a robustní i z hlediska numerického
- 18 • formulace je dostatečně obecná

19
20 **Úloha**
21 Odvodte matici zrcadlení podle libovolné roviny „standardním“ postupem používaným v kap.4.5
22 (Základní transformace v E^3). Porovnejte výsledné transformace jak z hlediska složitosti, tak i
23 robustnosti.

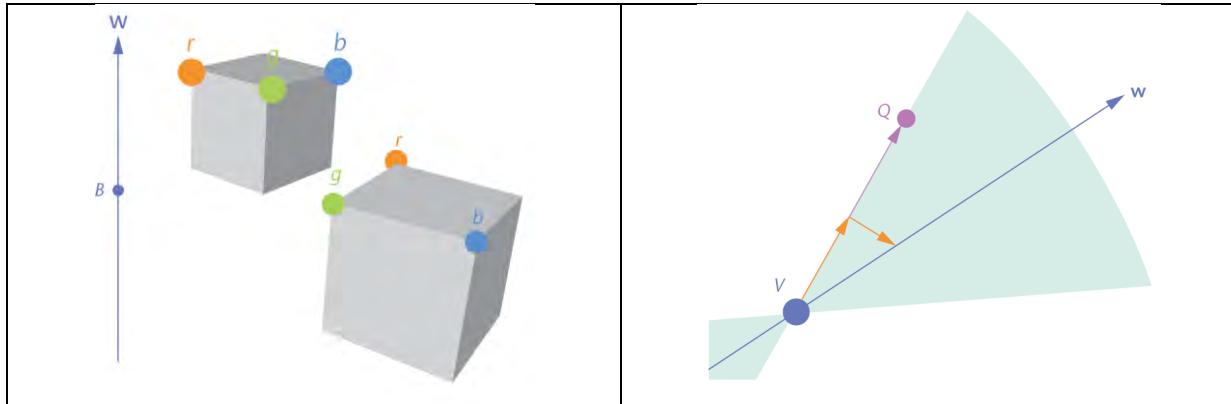
4.11.2. Rotace okolo obecné osy v E^3

Transformace rotace okolo obecné osy, založená na transformaci bodů, byla vysvětlena v kap. 4.6 (Rotace okolo dané osy v E^3) měla tvar:

$$Q = T^{-1} R_{zx}^{-1} R_{xy}^{-1} R(\theta) R_{xy} R_{zx} T \quad (4.58)$$

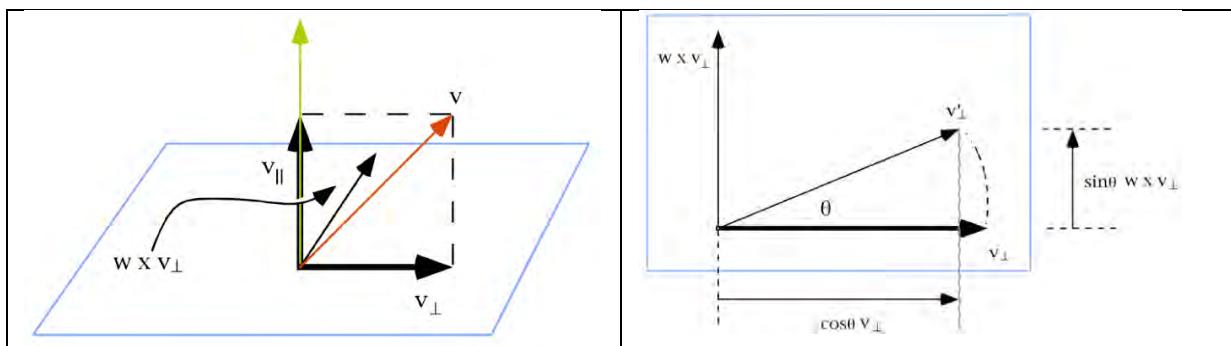
Zkusme nyní odvodit transformaci okolo obecné osy v E^3 jako rotaci vektorů s následným „uchycením“ pro daný souřadný systém.

Rotace je obecně dáná osou, tj. bodem x_0 , směrovým vektorem \hat{w} osy, okolo které rotujeme, a úhlem θ , o který budeme body otáčet. Úloha je vlastně úlohou, kdy máme rovinu rotace v bodě x_0 s normálovým vektorem \hat{w} .



Obr.4.21: Operace rotace - Courtesy of Miller,J.R.

Vektor $v = x - x_0$ opět můžeme rozdělit na vektory v_{\parallel} a v_{\perp} tak, že $v = v_{\parallel} + v_{\perp}$, tj. na složku rovnoběžnou a na složku kolmou k vektoru \hat{w} . Nyní však vektor v_{\perp} mění pozici na v'_{\perp} protože dochází k rotaci o úhel θ .



Obr.4.22: Operace rotace – použité vektory - Courtesy of Miller,J.R.

Takže

$$v_{\perp} = v - (\hat{w} \cdot v) \cdot \hat{w} \quad (4.59)$$

a

$$Mv_{\perp} = v_{\perp} \cos \theta + (\hat{w} \times v_{\perp}) \sin \theta \quad (4.60)$$

Neboť v_{\parallel} se rotací nemění, pak opět můžeme psát:

$$Mv = Mv_{\perp} + Mv_{\parallel} \quad (4.61)$$

a dosazením a úpravami dostáváme:

$$\begin{aligned} Mv &= (\hat{w} \cdot v) \cdot \hat{w} + (v - (\hat{w} \cdot v) \cdot \hat{w}) \cos \theta + (\hat{w} \times (v - (\hat{w} \cdot v) \cdot \hat{w})) \sin \theta \\ &= (\hat{w} \cdot v) \cdot \hat{w} + v \cos \theta - (\hat{w} \cdot v) \cdot \hat{w} \cos \theta \\ &\quad + (\hat{w} \times v) \sin \theta - \hat{w} \times (\hat{w} \cdot v) \cdot \hat{w} \sin \theta \end{aligned} \quad (4.62)$$

20

1 Protože vektor $\hat{w} \times (\hat{w} \times v)$ je kolmý na vektor \hat{w} je poslední člen výrazu nulový, tj.:
 2
$$\hat{w} \times (\hat{w} \cdot v) \cdot \hat{w} = 0 \quad (4.63)$$

2 Pak lze psát s použitím vztahu $(u \otimes v) \cdot \hat{w} = (v \cdot \hat{w})u$:

$$\begin{aligned} Mv &= (\hat{w} \cdot v) \cdot \hat{w} + v \cos \theta - (\hat{w} \cdot v) \cdot \hat{w} \cos \theta + (\hat{w} \times v) \sin \theta \\ &= (\hat{w} \otimes \hat{w}) \cdot v + \cos \theta v - \cos \theta (\hat{w} \otimes \hat{w}) \cdot v + \sin \theta (\hat{w} \times v) \end{aligned} \quad (4.64)$$

3 Abychom mohli vytknout vektor v , použijeme maticového vyjádření pro vektorový součin, a to:

$$\hat{w} \times v = Wv \quad (4.65)$$

4 kde matice W obsahuje složky normalizovaného vektoru \hat{w} a je definována takto:

$$W = \begin{bmatrix} 0 & -\hat{w}_z & \hat{w}_y \\ \hat{w}_z & 0 & -\hat{w}_x \\ -\hat{w}_y & \hat{w}_x & 0 \end{bmatrix} \quad (4.66)$$

5 Nyní lze psát:

$$\begin{aligned} Mv &= (\hat{w} \cdot v) \cdot \hat{w} + v \cos \theta - (\hat{w} \cdot v) \cdot \hat{w} \cos \theta + (\hat{w} \times v) \sin \theta \\ &= (\hat{w} \otimes \hat{w}) \cdot v + \cos \theta v - \cos \theta (\hat{w} \otimes \hat{w}) \cdot v + \sin \theta Wv \end{aligned} \quad (4.67)$$

6 a tedy:

$$\begin{aligned} Mv &= (\hat{w} \otimes \hat{w}) \cdot v + \cos \theta v - \cos \theta (\hat{w} \otimes \hat{w}) \cdot v + \sin \theta Wv \\ &= ((\hat{w} \otimes \hat{w}) + I \cos \theta - (\hat{w} \otimes \hat{w}) \cos \theta + \sin \theta W)v \end{aligned} \quad (4.68)$$

7 Úpravou pak dostáváme:

$$Mv = [\cos \theta I + (1 - \cos \theta)(\hat{w} \otimes \hat{w}) + \sin \theta W]v \quad (4.69)$$

8 Takéž matice rotace je dána

$$M = \cos \theta I + (1 - \cos \theta)(\hat{w} \otimes \hat{w}) + \sin \theta W \quad (4.70)$$

9
 10 Nyní je nutné opět určit translační složku, tj. vektor t a to dosazením bodu roviny. Protože pozice
 11 bodu roviny x_0 se nemění, tj. je stejná před i po transformaci, pak vektor posuvu je určen:

$$t = x_0 - Mx_0 \quad (4.71)$$

12
 13 Tím je celková matice Q pro transformaci rotace okolo obecné osy určena.
 14

15 Podrobněji viz:

- 16 • Miller,J.R.: The Mathematics of Graphical Transformations: Vector Geometric and
 17 Coordinate-Based Approaches, DesignLab, 1997 ([CLICK PDF](#) off-line)
- 18 • Miller,J.R.: Vector Geometry for Computer Graphics, IEEE Computer Graphics and
 19 Applications, pp.66-73, May/June 1999 ([CLICK PDF](#) off-line)
- 20 • Miller,J.R.: Applications of Vector Geoemtry for Robustness and Speed, IEEE Computer
 21 Graphics and Applications, pp.68-73, July/AugustMay/June 1999 ([CLICK PDF](#) off-line)

22 Úloha

23 Pokuste se odvodit vztahy pro body zadané v homogenních souřadnicích a využijte projektivní
 24 ekvivalence ke zjednodušení operace.

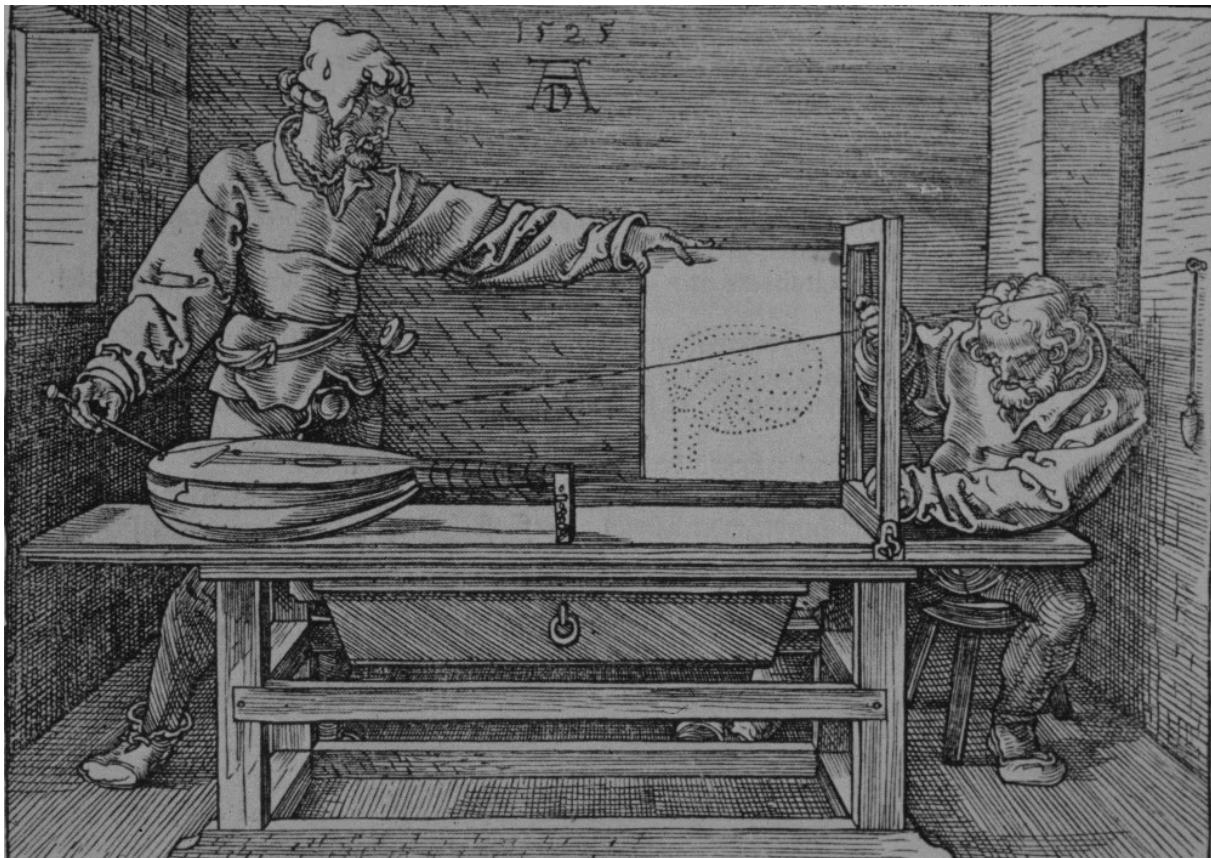
25
 26 **Dodatečné zdroje pro transformace:**

- 27 • Vince,J.: Mathematics for Computer Graphics, Springer, 2006
- 28 • Vince,J.: Matrix Transforms for Computer Games and Animation, Springer 2012

29

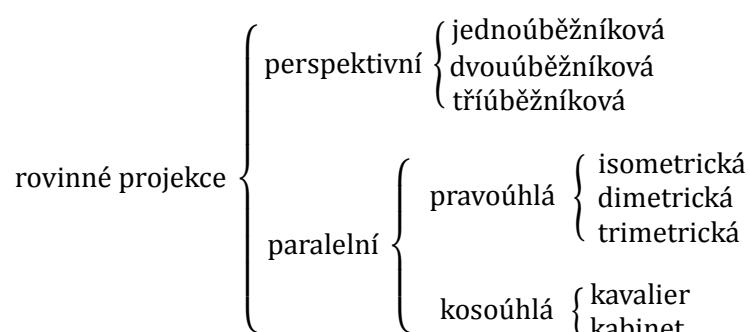
1 5. Projekce

Až dosud jsme se zabývali geometrickými transformacemi v E^2 a E^3 . Při zobrazování třírozměrných dat obvykle dochází k promítání 3D dat na 2D průmětnu, a to i v případě virtuální reality, kdy dochází ke generování stereoskopických obrazů, které pouze vytvářejí vjem 3D prostoru. Promítání je tedy obecně proces, kdy se 3D objekty zobrazují na nějakou plochu, která však nemusí být nutně rovinná. Jedním typem projekce je projekce perspektivní na rovinou plochu. Asi nejznámější kresba znázorňující perspektivní projekci pochází od Albrechta Dürera (1471-1528).



Obr.5.1: Znázornění perspektivní projekce

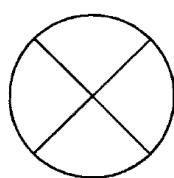
V počítačové grafice se převážně používají projekce rovinné, a ty lze klasifikovat takto:



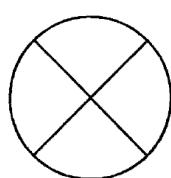
13 Paralelní projekce je vlastně speciálním případem projekce perspektivní, kdy pozorovatel je
14 v nekonečnu. Paralelní projekce se především používá v technické praxi, zatímco víceúběžníkové
15 perspektivy se používají hlavně v architektuře.

1 Aplikace projekce je nedílnou součástí mnoha aktivit lidské činnosti. Na základě 2D projekcí uživatel
2 „dostává“ informaci, jak 3D objekt vlastně vypadá. V oblasti strojírenství je velmi častá operace
3 rekonstrukce 3D tvaru z 2D pohledů. Tato úloha je poměrně komplikovaná, obecně nejednoznačná, a
4 to i v případě, že jsou vyznačeny neviditelné hrany obvykle pomocí čárkovaných hran. Na Obr.5.2.a je
5 těleso, které má všechny kolmé průměty stejné. Na Obr.5.2.b je pak fotografie vlastního tělesa.

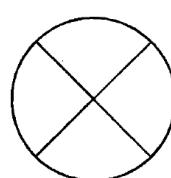
6



nárys



bokorys
pohled zleva



půdorys
pohled ze spodu



Obr.5.2: Průměty tělesa a vlastní těleso

7
8 Bohužel, v průběhu učení člověka jsme odkázáni na základní „technologii“, a to: papír, tužka, guma,
9 statický obraz.
10 To zajisté podstatně limituje představivost v průběhu učení člověka. Také staticnost zobrazení je
11 velmi limitujícím faktorem.

12

13 Pro ilustraci si představme:

- 14 • krychli, která je zavěšena na niti za vrchol a zkusme ji nakreslit přímo tak, jak opravu visí.
15 *Zhruba 40% lidí nakreslí takto zavěšený objekt špatně!*
16 • nyní krychli roztočme okolo svislé osy. Co uvidíme jako obálku takového rotačního tělesa?
17 *Zkusme takový objekt nakreslit a zdůvodnit, co vlastně vidíme. Zhruba 40 - 60% lidí nakreslí
18 takto zavěšený rotující objekt špatně!*

19

20 Odpověď je uvedena v kap. 5.7 (Projekce a faktor dynamiky).

21

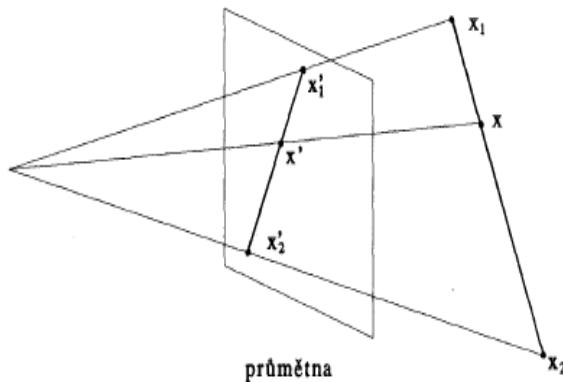
22

23

5.1.Perspektivní projekce

Perspektivní projekce reprezentuje velmi dobře vjem prostoru člověkem. Je známo, že paralelní přímky směřující od pozorovatele se pozorovateli jeví jako přímky, které se protínají. Takový bod se nazývá úběžník (vanishing point). Podle počtu takových bodů mluvíme o jednoúběžníkové, dvouúběžníkové nebo tříúběžníkové perspektivě. Úběžník si lze tedy představit jako zobrazení bodu v nekonečnu. Je tedy zřejmé, že pokud budeme zobrazovat krychli různě natočenou v prostoru, pak přímky, na kterých její hrany leží, se budou po projekci protínat maximálně ve třech bodech.

Při perspektivní projekci je nutné si uvědomit, že dochází ke ztrátě informace o vzdálenosti, neboť všechny body po průmětu, tj. po perspektivní transformaci, jsou na průmětně.



Obr.5.3: Jednoúběžníková perspektiva

Pokud parametricky vyjádříme úsečku v E^3 a její průmět na rovinu, tj. v E^2 , pak **není pravdou**, že parametr t odpovídající původnímu bodu v prostoru a parametr λ odpovídajícího bodu na průmětně jsou si rovny.

$$\begin{aligned} X &= X_1 + (X_2 - X_1) t & t \in \langle 0,1 \rangle \\ X'_1 &= X'_1 + (X'_2 - X'_1) \lambda & \lambda \in \langle 0,1 \rangle \end{aligned} \quad (5.1)$$

Pak obecně, kromě koncových bodů, platí:

$$t \neq \lambda \quad (5.2)$$

V případě projekcí se opět budeme snažit převést operaci projekce na maticové násobení, tj. do tvaru:

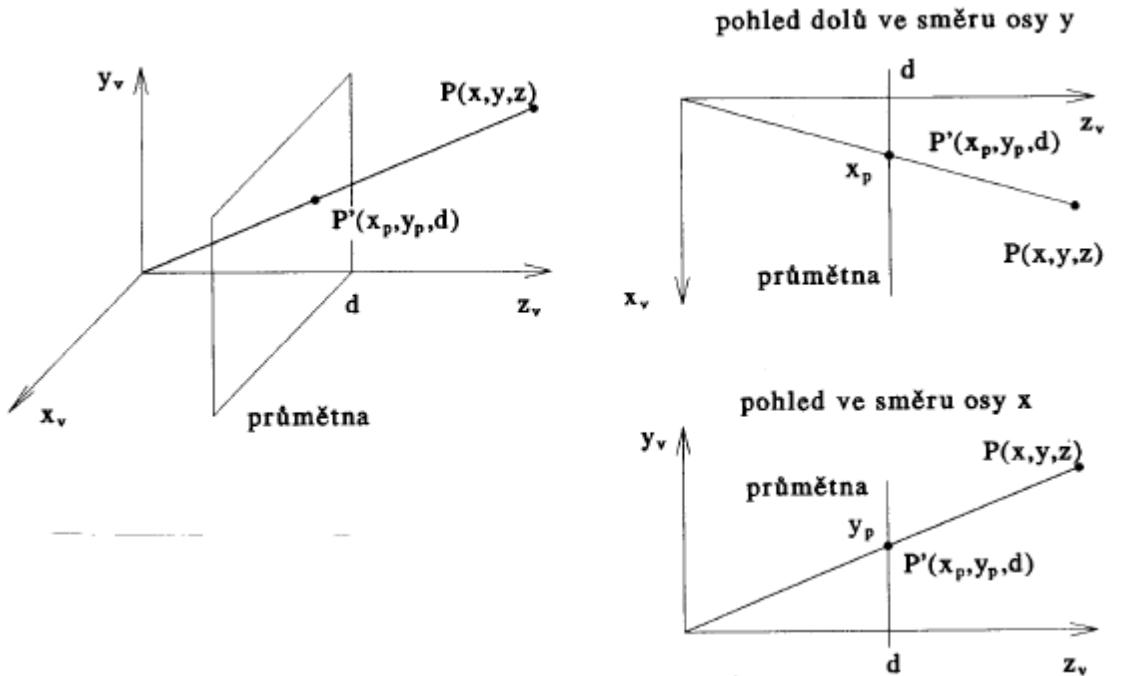
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1' \end{bmatrix} = M_{projekce} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5.3)$$

Je zřejmé, že takto realizovaná projekce však nesplňuje některé jisté vlastnosti, resp. požadavky, které jsou v počítačové grafice nutné, neboť:

- se ztratí informace o vzdálenosti, protože výsledné objekty jsou reprezentovány na průmětně, tj. výsledkem jsou vlastně dvourozměrné souřadnice bodů po projekci
- transformují se i body objektů, které jsou za pozorovatelem. Navíc, pokud geometrický element, např. trojúhelník, má jeden vrchol za pozici pozorovatele a ostatní vrcholy před pozorovatelem, není výsledek operace korektní. Jinými slovy řečeno, je nutné odříznout objekty a jejich části, které jsou „za pozorovatelem“, tj. je nutné použít algoritmus ořezávání (*clipping*).

5.2. Matematický aparát rovinné projekce

- Souřadný systém, ve kterém je objekt definován, je pravotočivý. Takže pokud se pozorovatel dívá směrem k počátku na ose z tohoto souřadného systému, tak se vlastně dívá proti směru této osy. Souřadný systém pozorovatele, resp. kamery, je proto přirozeně *levotočivý*.



Obr.5.4: Geometrická představa projekce

Pro jednoduchost budeme uvažovat průmětnu kolmou na osu z a ve vzdálenosti d . Je zřejmé, že použitím podobnosti trojúhelníků dostaneme rovnice:

$$\frac{x_p}{d} = \frac{x}{z} \quad \frac{y_p}{d} = \frac{y}{z} \quad (5.4)$$

Jednoduchou úpravou pak dostáváme:

$$x_p = \frac{x}{z} d = \frac{x}{z/d} \quad y_p = \frac{y}{z} d = \frac{y}{z/d} \quad (5.5)$$

Dělení hodnotou z souřadnice vlastně způsobí, že pokud jsou objekty stejné velikosti, pak bližší objekt bude větší než objekt vzdálenější. Vzdálenost průmětny od počátku je vlastně jen měřítkovým faktorem.

Perspektivní projekci je možné popsat vztahem:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \mathbf{M}_{per} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5.6)$$

Všimněme si, že hodnota homogenní souřadnice $w = 1$.

Roznásobením dostáváme:

$$[x, y, z : z/d]^T = \left[\frac{x}{z/d}, \frac{y}{z/d}, \frac{z}{z/d} : 1 \right]^T \triangleq \left(\frac{x}{z} d, \frac{y}{z} d, d \right)^T \quad (5.7)$$

kde: \triangleq je operátor projektivní ekvivalence.

1 Zde je nutné poznamenat, že se předpokládá, že buď homogenní souřadnice bodu w je rovna
 2 hodnotě 1, tj. $w = 1$, nebo se musí následně ještě hodnota w' násobit hodnotou w . Nicméně
 3 v grafických akcelerátorech se operace projekce realizuje trochu jiným způsobem, neboť s každým
 4 bodem musí být, např. pro řešení viditelnosti, ještě k dispozici informace o vzdálenosti bodu od
 5 pozorovatele. Vzdálenost bodu od pozorovatele se ve skutečnosti řeší zavedením pseudovzdálenosti,
 6 viz kap.5.4 (Perspektiva a pseudovzdálenost).

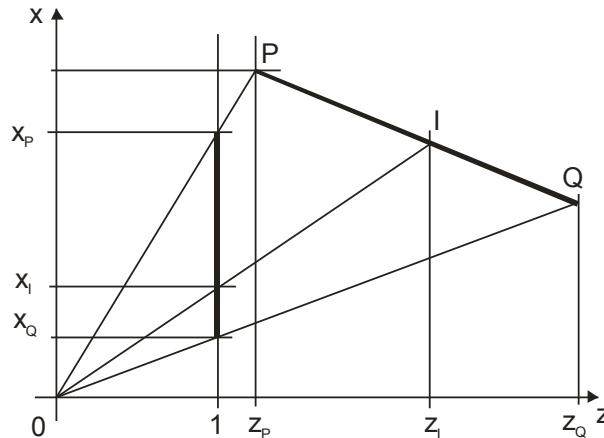
7
 8 Vzniká také otázka, zda lze určit vzdálenost od pozorovatele, tj. souřadnici z , pokud známe
 9 souřadnice původních a promítnutých koncových bodů úsečky z E^3 .

10
 11 **Určení vzdálenosti bodu na úsečce po projekci**
 12 Pro výpočet použijeme tu souřadnici x nebo y , která má *největší diferenci*. Nechť jde o osu x pro
 13 účely odvození vztahu v Eukleidovském prostoru. Pak platí:

$$\begin{aligned}x_I &= (1 - \lambda)x_P + \lambda x_Q \\y_I &= (1 - \lambda)y_P + \lambda y_Q\end{aligned}\quad (5.8)$$

14 kde: λ je parametr na průmětně.

15 Výpočet hodnoty z_I je poměrně jednoduchý.



Obr.5.5: Inverzní výpočet souřadnice z

16 Pokud vyjdeme z obrázku Obr.5.5, kde pro jednoduchost dáme průmětnu do vzdálenosti 1, pak lze
 17 ukázat, že platí:

$$\frac{1}{z_I} = (1 - \lambda) \frac{1}{z_P} + \lambda \frac{1}{z_Q} \quad (5.9)$$

18 neboť:

$$\frac{z_I - z_P}{z_I z_I - x_P z_P} = \frac{z_Q - z_P}{x_Q z_Q - x_P z_P} \quad (5.10)$$

19 Vynásobením jmenovateli a vydělením $z_P z_I z_Q$ dostaváme:

$$\frac{x_Q - x_P}{z_I} = \frac{x_Q - x_I}{z_P} - \frac{x_I - x_P}{z_Q} \quad (5.11)$$

20 Pokud nyní použijeme substituci:

$$x_I - x_P = \lambda(x_Q - x_P) \quad x_Q - x_I = (1 - \lambda)(x_Q - x_P) \quad (5.12)$$

21 a následně vydělíme výslednou rovnici výrazem $x_Q - x_P$, pak dostaneme uvedený vztah.

22 • Ammeraal,L.: Programming Principles in Computer Graphics, pp.146, John Wiley, 1992

23

1 Pro výpočty perspektivní distorze je možné využít ještě jiný postup, kterým určíme závislost:

$$\mu = \lambda(\mu) \quad (5.13)$$

2 Je dána úsečka dvěma body

$$P = P_1 + \mu(P_2 - P_1) \quad (5.14)$$

3 a její projekce

$$P' = P'_1 + \lambda(P'_2 - P'_1) \quad (5.15)$$

4 Pak pro souřadnice x a z platí:

$$\frac{x}{z} = \frac{x'}{1} \quad x = x'z \quad (5.16)$$

5 a lze psát:

$$\begin{aligned} x_1 + \mu(x_2 - x_1) &= x'(z_1 + \mu(z_2 - z_1)) \\ x_1 - x'z_1 &= \mu(x'(z_2 - z_1) - (x_2 - x_1)) \end{aligned} \quad (5.17)$$

6 Pak

$$\begin{aligned} \mu &= \frac{x_1 - x'z_1}{x'(z_2 - z_1) - (x_2 - x_1)} = \frac{z_1z_2(x_1 - x'z_1)}{z_1z_2(x'(z_2 - z_1) - (x_2 - x_1))} \\ &= \frac{z_1(x_1z_2 - x'z_1z_2)}{x'z_1z_2(z_2 - z_1) + x_1z_1z_2 - x_2z_1z_2} \end{aligned} \quad (5.18)$$

7 Ale víme, že

$$x' = x'_1 + \lambda(x'_2 - x'_1) \quad (5.19)$$

8 pak:

$$\begin{aligned} x'z_1z_2 &= x'_1z_1z_2 + \lambda z_1z_2(x'_2 - x'_1) = \left(\frac{x_1}{z_1}\right)z_1z_2 + \lambda z_1z_2\left(\frac{x_2}{z_2} - \frac{x_1}{z_1}\right) \\ &= x_1z_2 + \lambda(x_2z_1 - x_1z_2) \end{aligned} \quad (5.20)$$

9 Dosazením do rovnice pro μ dostáváme:

$$\begin{aligned} \mu &= \frac{z_1(x_1z_2 - x_1z_2 - \lambda(x_2z_1 - x_1z_2))}{(x_1z_2 + \lambda(x_2z_1 - x_1z_2))(z_2 - z_1) + x_1z_1z_2 - x_2z_1z_2} \\ &= \frac{z_1\lambda(x_1z_2 - x_2z_1)}{x_1z_2^2 - x_1z_1z_2 + x_1z_1z_2 - x_2z_1z_2 + \lambda(x_2z_1 - x_1z_2)(z_2 - z_1)} \\ &= \frac{z_1\lambda(x_1z_2 - x_2z_1)}{(x_1z_2 - x_2z_1)(z_2 - \lambda(z_2 - z_1))} \end{aligned} \quad (5.21)$$

10 Pak

$$\mu = \frac{\lambda z_1}{z_2 - \lambda(z_2 - z_1)} \quad (5.22)$$

11 Takže, pokud např. interpolujeme na průmětně pomocí parametru u

$$u = u_1 + \lambda(u_2 - u_1) \quad (5.23)$$

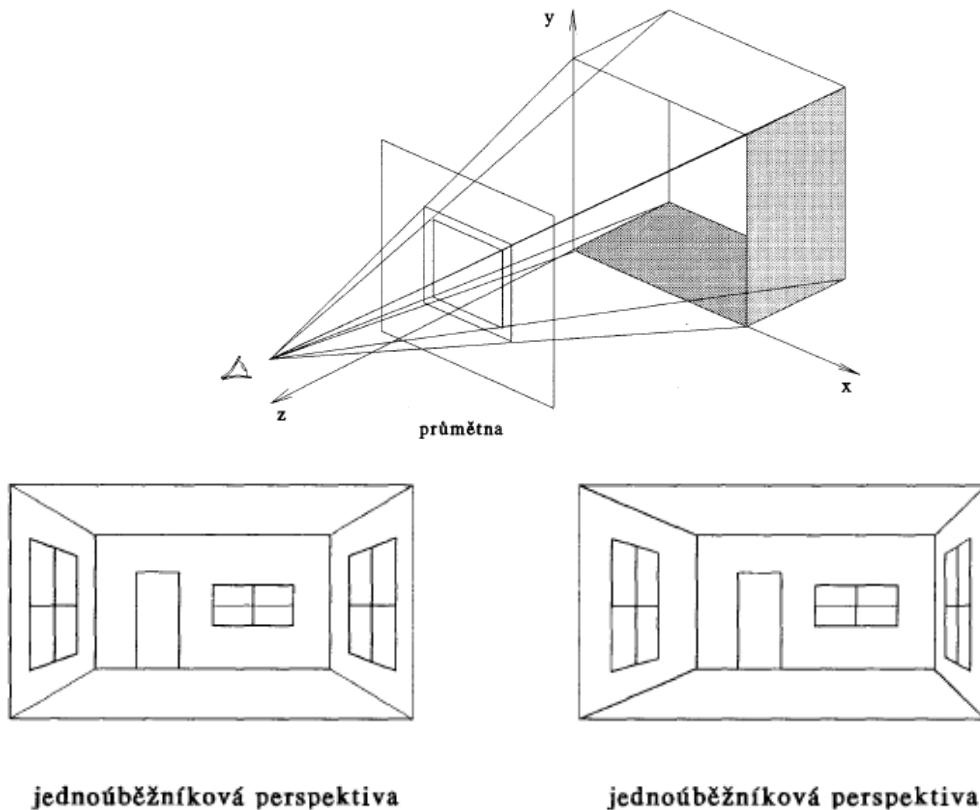
12 pak vlastně musíme realizovat nelineární interpolaci s použitím parametru μ , tj.

$$\begin{aligned} u &= u_1 + \mu(u_2 - u_1) = u_1 + \frac{\lambda z_1}{z_2 - \lambda(z_2 - z_1)}(u_2 - u_1) \\ &= \frac{u_1z_2 - \lambda u_1(z_2 - z_1) + \lambda z_1(u_2 - u_1)}{z_2 - \lambda(z_2 - z_1)} = \frac{u_1z_2 + \lambda(u_2z_1 - u_1z_2)}{z_2 - \lambda(z_2 - z_1)} \\ &= \frac{\frac{u_1}{z_1} + \lambda\left(\frac{u_2}{z_2} - \frac{u_1}{z_1}\right)}{\frac{1}{z_1} + \lambda\left(\frac{1}{z_2} - \frac{1}{z_1}\right)} \end{aligned} \quad (5.24)$$

- 13 • Hitchell,D.A.P.: Fast Algorithms and Hardware for 3D Computer Graphics, PhD Thesis, 1999
<http://www.sed.shef.ac.uk/intranet/phdthesis/Mitchell1992.pdf>

1 Jednoúběžníková perspektiva

2 Při konstrukci jednoúběžníkového perspektivního pohledu na krychli je průmětna rovnoběžná s jednou stěnou promítané krychle.



Obr.5.6: Jednoúběžníková perspektiva

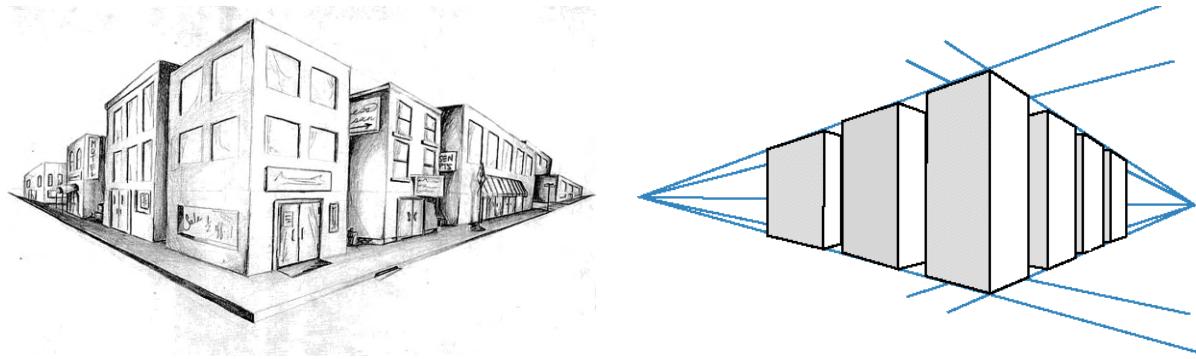
4 Je zřejmé, že pokud budeme pohybovat krychlí nahoru/dolů nebo vpravo/vlevo, úběžník bude mít
5 jinou polohu. Jednoúběžníková perspektiva se používá velmi často, zejména při návrzích bytových
6 interiérů apod.

7 Zkusme se nyní podívat, co se stane, pokud krychli budeme otáčet v rovině zx .

8

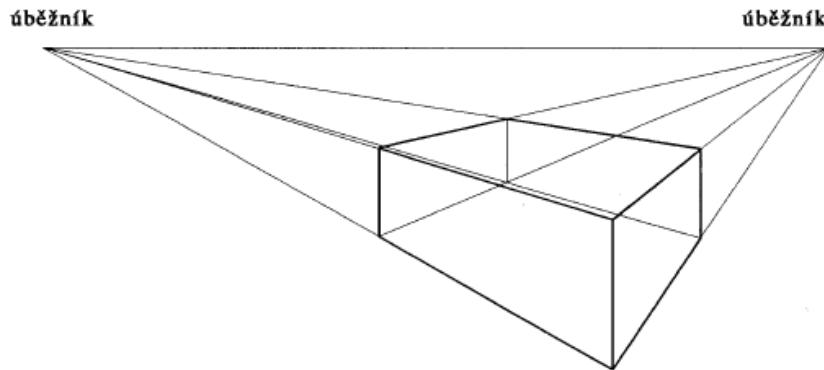
9 Dvouúběžníková perspektiva

10 Případ dvouúběžníkové perspektivy je celkem častý. V praxi takový pohled lze snadno získat tak, že
11 směr pohledu je „diagonálně“ z opačného rohu křížovatky.



Obr.5.7: Dvouúběžníková perspektiva

12 Poznamenejme, že dva úběžníky určují přímku, která je označována jako *horizont*. Jeho pozice závisí
13 na vertikální pozici pozorovatele.



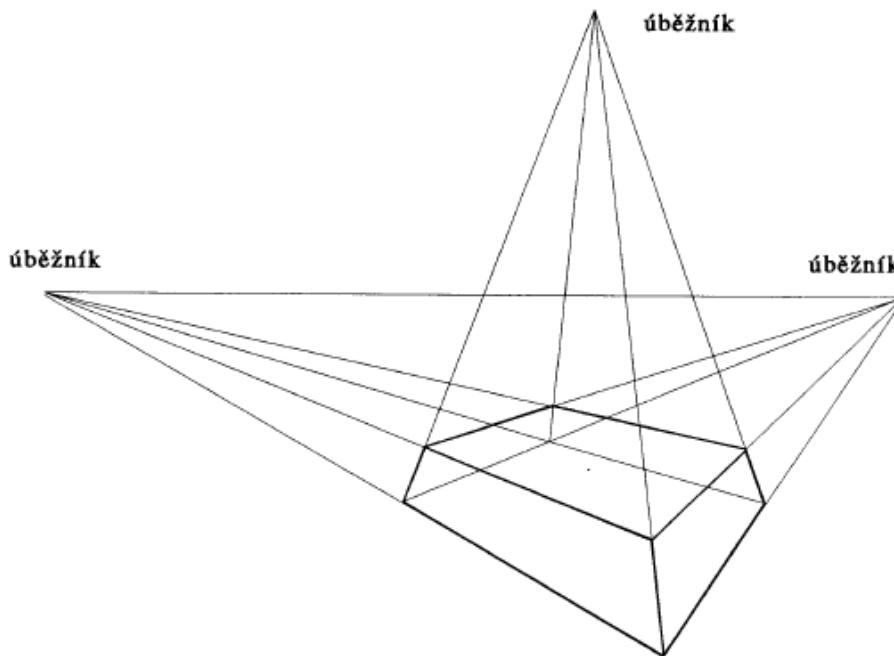
Obr.5.8: Dvouúběžníková perspektiva - vliv horizontu

Obecně lze říci, že dvouúběžníkový perspektivní pohled na krychli vzniká tak, že jedna hrana krychle je *rovnoběžná s jednou osou* (obvykle svislou) souřadného systému.

Pokud budeme fotografovat vysoké objekty, např. kostelní věž, horní část bude na fotografii užší než její šíře u paty věže, což je dáno tím, že se vlastně uplatní tříúběžníková perspektiva, tj. hrana krychle je po *promítnutí rovnoběžná s osou y* na průmětně. Pochopitelně obraz může být rotován, ale to už je operace na průmětně.

Tříúběžníková perpektiva

Tříúběžníková perspektiva u krychle vlastně vznikne tak, že průmětna je vůči krychli v libovolné poloze a ani hrana krychle už není rovnoběžná s plochou průmětny.

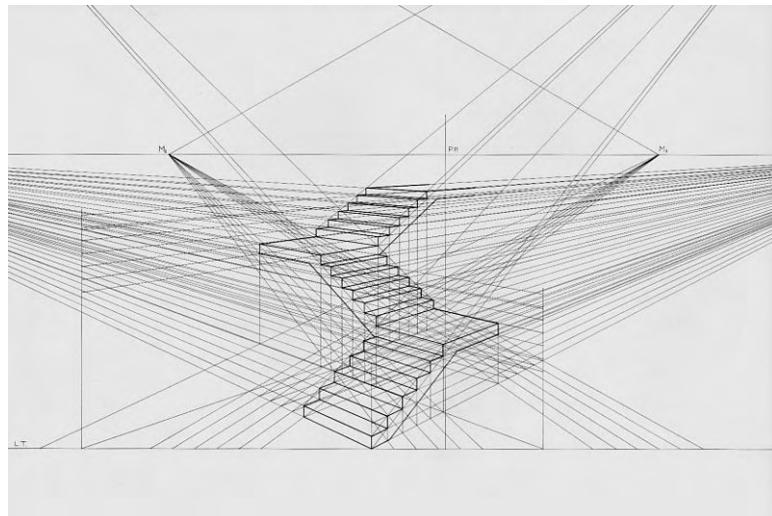


Obr.5.9: Tříúběžníková perspektiva

Tříúběžníková perspektiva se nepoužívá příliš často. U více úběžníkových projekcí jde v principu většinou o dvouúběžníkovou perspektivu s tím, že plochy objektu nejsou pouze na sebe kolmé apod.

1 Perspektiva s více úběžníky

- 2 Pokud si však představíme obecnější těleso než krychli, pak na obrázku po projekci najdeme více úběžníků.
- 3
- 4



Obr.5.10: Situace s více úběžníky

- 5 Další „pseudoúběžníky“ vznikají např. z hran schodiště apod.

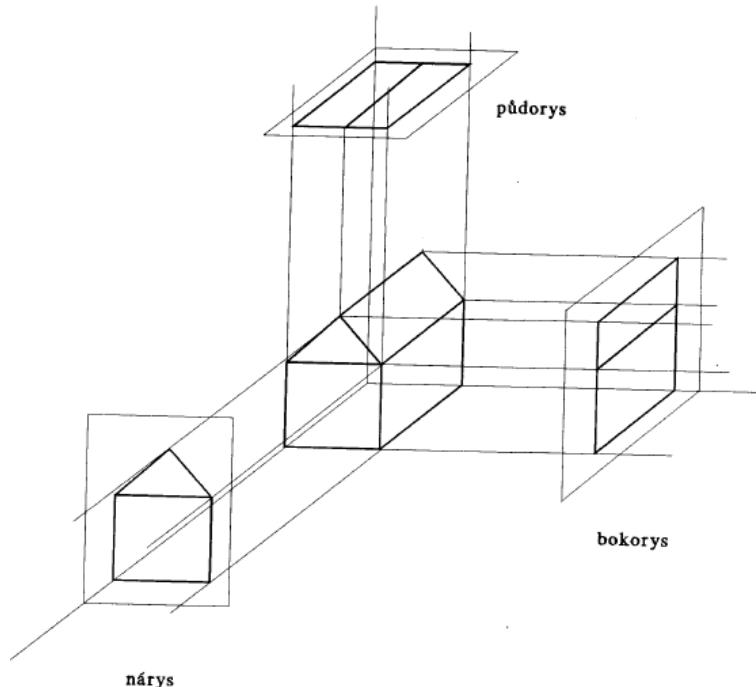
6

5.3. Paralelní projekce

Paralelní projekce se používají především v technické praxi, a to buď projekce pravoúhlé, nebo kosoúhlé. Kosoúhlé projekce jsou oblíbené proto, že umožňují získat 3D představu a je možné částečné odměřování délek.

Pravoúhlá projekce

Nejčastěji se používá pravoúhlá (orthogonal projection) projekce, a to zejména ve strojírenství a stavebnictví apod., neboť umožňuje částečné odměřování, což perspektivní projekce z principu neumožňují.



Obr.5.11: Pravoúhlé projekce

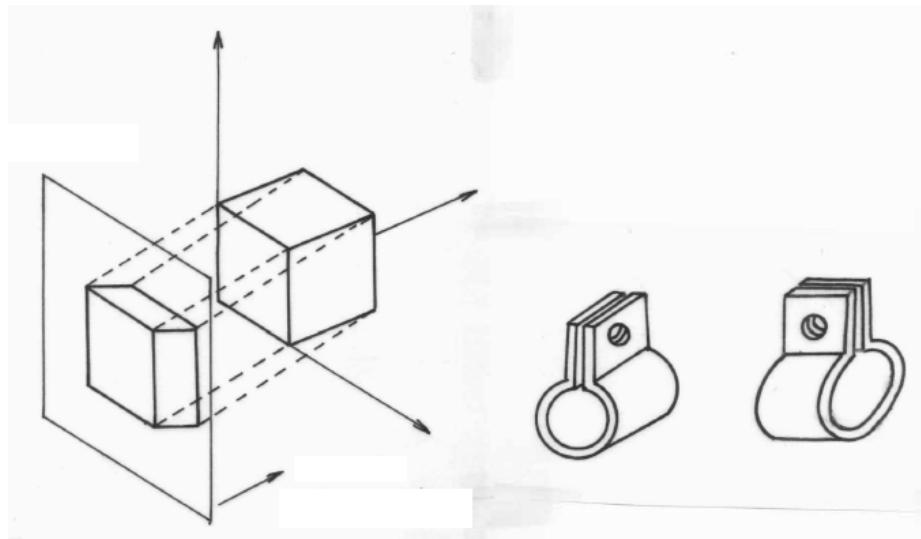
Paralelní projekce jsou vlastně speciálním případem perspektivní projekce, kdy pozorovatel je v „nekonečnu“.

Pravoúhlou projekcí se zabývat speciálně nebudeme, neboť řešení je triviální a vlastní projekce je popsána vztahem:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \mathbf{M}_{par} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5.25)$$

1 Kosoúhlé projekce

2 Kosoúhlé projekce se také často používají v praxi. Jejich výhodou je částečná možnost odměřování,
3 např. z výkresu, hran ploch, které jsou rovnoběžné s průmětnou a dávají 3D vjem.
4



5
6 Obr.5.12: Kosoúhlé projekce
7

- 8 Je nutné zdůraznit, že vlivem zobrazení dochází ke zkreslení tvarů, které nejsou rovnoběžné s rovinou
9 průmětny.
10
11 Až dosud byly rovinné projekce reprezentovány pro případ $w = 1$, což není vždy možné, a navíc byla
12 ztracena vzdálenost bodu od pozorovatele, která je nutná pro řešení viditelnosti.
13

5.4. Perspektiva a pseudovzdálenost

Pro řešení viditelnosti jednotlivých částí objektu se v současné době převážně používá tzv. z-buffer (depth buffer), viz kap.12 (Algoritmy řešení viditelnosti). Ztráta informace o vzdálenosti bodu od pozorovatele, tzv. o hloubce, je nepřijatelná, neboť by pak nebylo možné řešit otázku viditelnosti. Vzdálenost d bodu P od pozorovatele, který je v počátku souřadného systému, je dána vztahem:

$$d = \sqrt{P_x^2 + P_y^2 + P_z^2} \quad (5.26)$$

což je evidentně nelineární vztah. Nicméně pro účely určení viditelnosti vlastně nepotřebujeme vzdálenost, ale potřebujeme informaci, co je dál a co je blíže k rozhodnutí, která část objektu je zakryta, resp. je viditelná pozorovatelem. Takže stačí vlastně funkce, jejíž hodnota monotónně roste se vzdáleností, tzv. pseudovzdálenost. Z výpočetního hlediska je rozumné, aby výpočet pseudovzdálenosti používal stejný jmenovatel, jak pro souřadnici x , tak i pro souřadnici y v perspektivní projekci s nějakým lineárním členem zahrnujícím hodnotu z souřadnice.

Upozornění: Z důvodu konzistentnosti s většinou dostupných publikací a grafických knihoven budeme v následujícím používat pravotočivý souřadný systém pro výklad pseudo-vzdálenosti.

Pro výpočet perspektivní projekce s pseudovzdáleností se používá vztah:

$$(x^*, y^*, z^*) = \left(d \frac{P_x}{-P_z}, d \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right) \quad (5.27)$$

Z uvedené transformace vyplývá, že hodnoty x^* , y^* jsou stejné hodnoty jako v případě jednoduché perspektivní projekce a hodnota z^* určující pseudovzdálenost je ovlivněna dvěma parametry a a b . Tyto parametry by šlo volit prakticky libovolně, ale z praktických důvodů, kdy z^* hodnota je v pohyblivé řádové čárce, je rozumné volit hodnoty parametrů a a b tak, aby $z^* \in (-1, 1)$. V reálné scéně jsou objekty „uzavřené“ mezi dvěma rovinami, a to rovinou přední (Near), která se označuje jako N , a rovinou zadní (Far), která se označuje jako F . Vzhledem k tomu, že vzdálenost průmětny má vliv jen na velikost, ztotožňuje se průmětna s rovinou N a tedy:

$$(x^*, y^*, z^*) = \left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right) \quad (5.28)$$

Z výše uvedených podmínek, tj.:

$$-1 = \frac{aN + b}{-N} \quad 1 = \frac{aF + b}{-F} \quad (5.29)$$

úpravou pak dostáváme:

$$N = aN + b \quad -F = aF + b \quad (5.30)$$

Řešením pak:

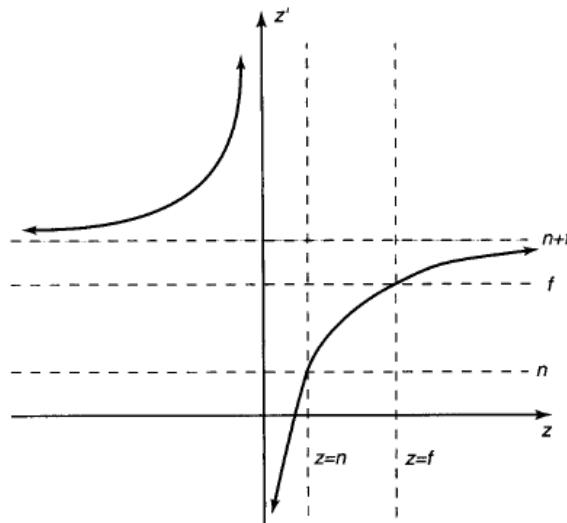
$$a = -\frac{N + F}{F - N} \quad b = -\frac{2NF}{F - N} \quad (5.31)$$

Hodnota z^* je vlastně hodnotou, která v z-bufferu slouží k rozhodnutí, zda daný pixel se má vykreslit, či nikoliv vzhledem k viditelnosti. Z grafu na Obr.5.13 je zřejmé, že pro velké hodnoty z dochází k malým změnám, což může vést ke špatnému určení viditelnosti vlivem konečné přesnosti pro z^* .

Protože $N \ll F$, lze pseudovzdálenost approximovat vztahem:

$$z^* = 1 - \frac{2N}{P_z} \quad (5.32)$$

Výše uvedená formulace je vlastně pro kartézské souřadnice, nikoliv pro projektivní prostor, kde bod je reprezentován svými homogenními souřadnicemi.



1
2 Obr.5.13: Vztah vzdálenosti z a pseudovzdálenosti z^*
3

4 Transformace pro perspektivní projekci pro body danými v homogenních souřadnicích, tj. bod x je
5 určen jako $x = [x, y, z: w]^T$, je určena transformací:

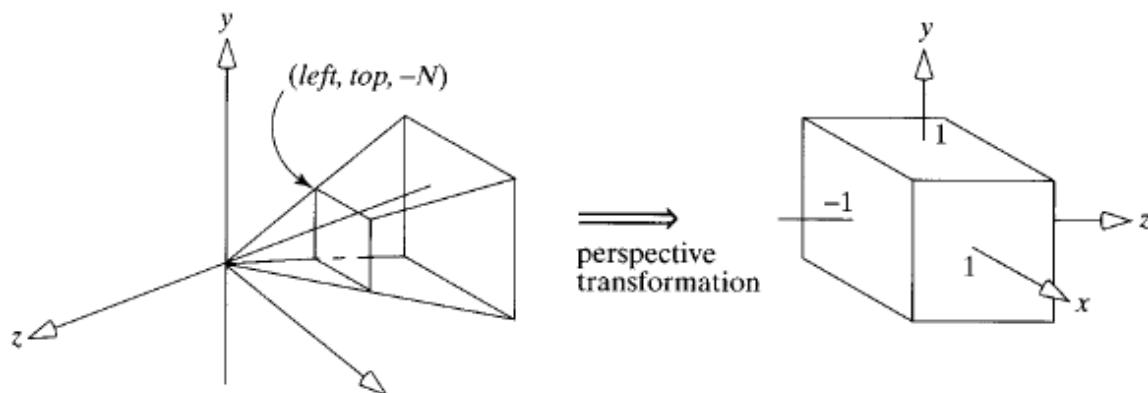
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} Nx \\ Ny \\ az + b \\ z \end{bmatrix} \quad (5.33)$$

6 Pro body s homogenní složkou obecně $w \neq 1$, ale $w \neq 0$ pak můžeme psát (použijeme P_x atd. pro
7 souřadnice Eukleidovské) pro odlišení:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} wP_x \\ wP_y \\ wP_z \\ w \end{bmatrix} = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} wNP_x \\ wNP_y \\ w(aP_z + b) \\ wP_z \end{bmatrix} \quad (5.34)$$

8 Tato transformace však už nemá poslední řádek $[0,0,0:1]^T$. Všechny geometrické transformace,
9 které nemají poslední řádek transformační matice ve tvaru $[0,0,0:1]^T$, už nerealizují afinní
10 transformaci, ale transformaci perspektivní, v našem případě perspektivní projekci. Dá se ukázat, že
11 perspektivní projekce je vlastně zřetězením operace perspektivní transformace a paralelní projekce.

12 Při realizaci vlastního zobrazení se vlastně zobrazují ty objekty, resp. jejich části, které jsou v tzv.
13 pohledové pyramidě. Ta je určena obdélníkem na průmětně a pozici pozorovatele. Je tedy přirozenou
14 otázkou, zda by bylo možné objekty „nějak transformovat“ tak, aby byl nahrazen vliv perspektivní
15 projekce, a vše zpracovávat obdobně jako při paralelní projekci.



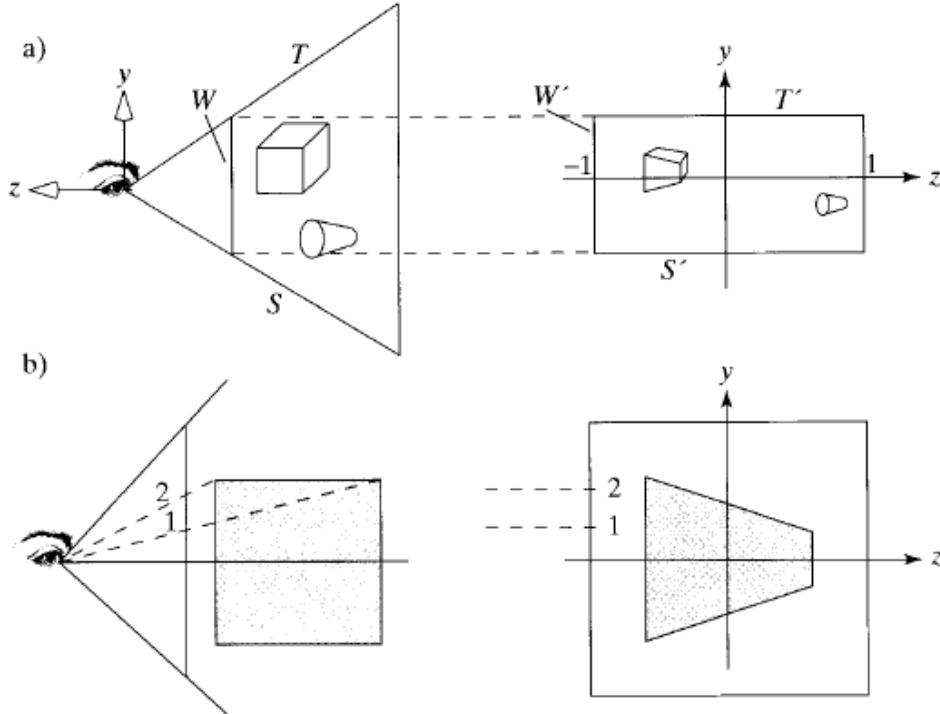
16
17 Obr.5.14: Perspektivní transformace

1 Vzhledem k tomu, že se transformují body, resp. vrcholy lineárních útvarů, jako je úsečka, trojúhelník
 2 apod., lze postupovat ve dvou krocích, a to:

- 3 transformace objektů do tvaru, který po paralelní projekci dá stejný výsledek
- 4 transformace do normalizovaných souřadnic NDC prostoru $(-1,1) \times (-1,1) \times (-1,1)$, který
 5 se nazývá kanonický pohledový objem (canonical view volume).

6

7 Je zřejmé, že místo hodnoty z souřadnice bude výsledkem opět pseudovzdálenost z^* .



8 Obr.5.15: Deformace vlivem perspektivní transformace

9

10 Nepřímým důsledkem této transformace je, že:

- 11
- 12 v mnoha operacích se eliminuje operace násobení; jde vlastně o násobení hodnotou ± 1
 - 13 nevyšší se výpočetní nároky, neboť transformace se „pouze přidává“ do kumulované matice
 - 14 transformace, která je konstantní, pokud se nemění pozice pozorovatele.

15 Zobrazovací okno je určenou obecně souřadnicemi $\langle left, right \rangle \times \langle bottom, top \rangle$ a výsledná
 16 transformace je pak určena maticí:

$$\begin{bmatrix} 2N \\ right - left & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2N}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & -\frac{F + N}{F - N} & -\frac{2FN}{F - N} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (5.35)$$

17

18 Matice projekce (Projection Matrix) zajišťuje perspektivní transformaci, posuv a změnu měřítka
 19 včetně převodu do kanonického pohledového objemu.

20

1 V případě použití knihovny OpenGL je tato matica vytvořena OpenGL funkciemi:

2
3 glFrustum(left, right, bottom, top, N, F)

4 nebo „uživatelsky přívětivou“ funkcí:

5
6 gluPerspective(viewAngle, aspect, N, F)

7 kde:

$$\begin{aligned} top &= N \operatorname{tg}\left(\frac{\pi}{180} \operatorname{viewAngle}/2\right) & right &= top * aspect \\ left &= -right & bottom &= -top \end{aligned}$$

10
11 Je zřejmé, že některé objekty jsou zcela nebo částečně vně pohledové pyramidy, a ty je tedy nutné,
12 resp. jejich části, vhodným způsobem odstranit. Tuto operaci zajistí operace ořezávání, tj. oříznutí
13 vůči kanonickému (pohledovém) objemu, viz kap.6 (Metody ořezávání v E^2 a E^3 a operace s n-
14 úhelníky).

15
16 V současné době se začíná prosazovat trh s 3D zobrazováním, které je v převážné většině realizováno
17 stereoskopickou projekcí. Je tedy vhodné zde uvést alespoň základní informace.

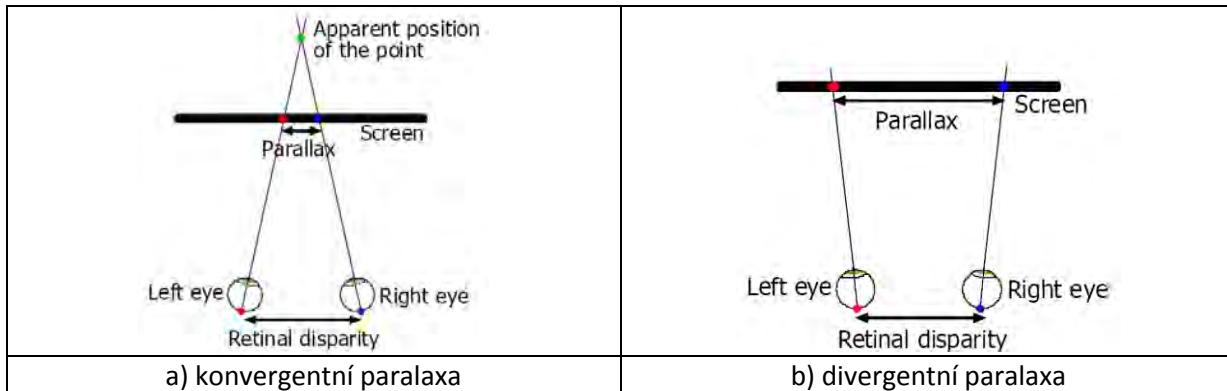
18
19 Zajímavé řešení problému projekce s použitím rotace ve $4D$, viz:

- 20 • Goldman,R.: Modeling perspective projection by rotations in 4-dimensions, Graphical
21 Models, Vol.75, pp.41-45,2013

22
23

1 5.5.Stereoskopická projekce

2 Stereoskopická projekce je v zásadě založena na perspektivní projekci, kdy se generují dva obrazy, a
 3 to jeden pro levé oko a druhý obraz pro pravé oko, přičemž se respektuje oční vzdálenost (*disparita*),
 4 tj. cca 65 [mm]. Tím se vytváří iluze 3D prostoru z hlediska percepce.



5 Obr.5.16: Stereoskopická projekce

6 Je nutné zdůraznit, že zejména:

- 7 • stereoskopický obraz je „korektní“ jen pro takového pozorovatele, který je přesně v pozici,
 8 pro kterou byl obraz generován. To znamená, že např. u 3D televize nebo v 3D kině mají
 9 ostatní pozorovatelé obraz nekorektní
- 10 • stereoskopický obraz musí být generován tak, aby respektoval vlastnosti projekčního zařízení,
 11 tj. skutečnou velikost promítací plochy a fyzické rozlišení. Pokud se změní velikost promítací
 12 plochy, např. 5x v každém směru, tak se disparita také 5x zvětší a oční osy nebudou
 13 konvergovat do společného bodu a budou případně i divergovat.

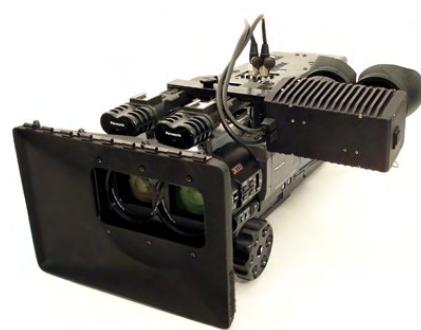
14

15 Při generování stereoskopických obrazů je celá řada dalších faktorů, které je nutné respektovat, aby
 16 výsledný vjem byl akceptovatelný. V mnoha případech výsledný 3D stereo obraz vykazuje „cartoon
 17 effect“, tj. jako kdyby scéna byla složena z 2D řezů.

18



Fotoaparát s 3D nástavcem



3D profesionální kamera

19 Obr.5.17: Stereoskopické kamery

20

21 Rovinné projekce, které byly uvedeny výše, jsou zajisté velmi často používané, avšak také hodně
 22 limitující. Vedle rovinných projekcí se používají i projekce nerovinné, ať už v počítačové grafice,
 23 počítačovém vidění, nebo v kartografii atd. Tyto projekce však už nejsou realizovatelné lineárními
 24 vztahy ani při použití projektivního rozšíření kartézského souřadného systému.

5.6. Nerovinné projekce

Typické nerovinné projekce, mezi které lze počítat válcovou a sférickou projekci, se běžně používají např. v kartografii. Válcová projekce se používá pro zobrazení scény především se širokoúhlým záznamem. U fotografií s malou ohniskovou vzdáleností a širokým úhlem záběru je známo, že objekty na stranách jsou zkresleny. Takže by bylo správné obraz promítat na válcovou plochu. Sférická projekce se s výhodou používá např. ve vizualizaci astronomických dat.

Uvedeme jen některé zajímavé nerovinné projekce, resp. odkazy s podrobnějšími informacemi:

- *Panniniho projekce* (Pannini projection), která je určena především pro perspektivní obrazy s velkým pohledovým úhlem (<http://vedutismo.net/Pannini/>).

Dobrý úvod je viz http://wiki.panotools.org/The_General_Panini_Projection



Gian Paolo Pannini, Interior of St. Peter's, Rome, c. 1754, oil on canvas, 60.75 x 77.5 in. National Gallery of Art, Washington

Obr.5.18: Paniniho projekce

- *projekce Mercatorova* (Mercator projection) používaná v GIS systémech a kartografii http://www.geo.utexas.edu/courses/420k/PDF_files/LABS/GIS/map_project.pdf



Obr.5.19: Mercatorova projekce

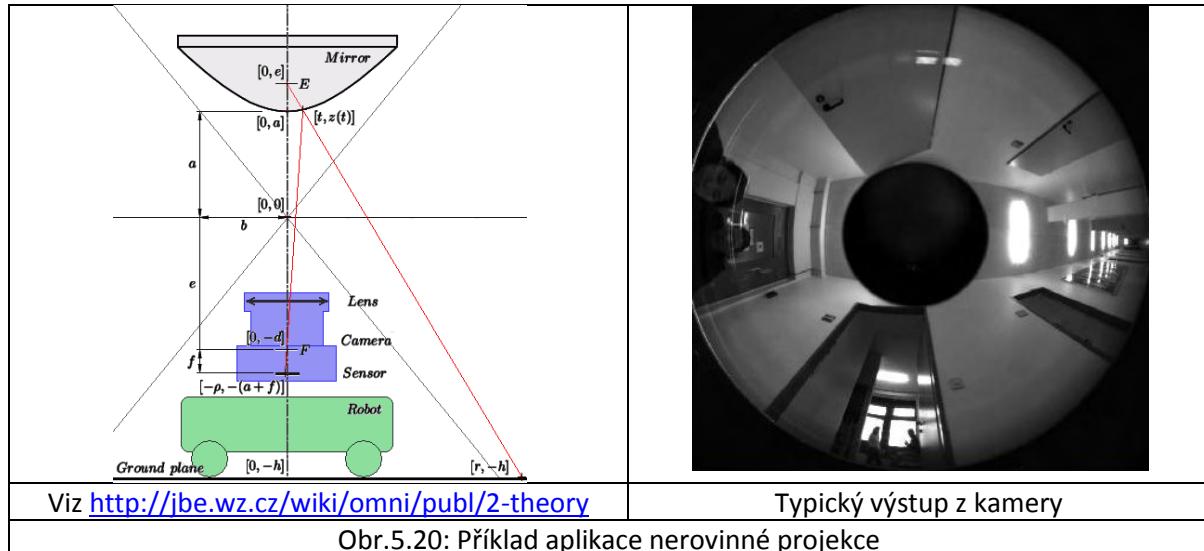
<http://dawnemapy.com.pl/pages/mapy/swiat/xvi-w.php?p=40>

- 1 • Nerovinné projekce a aplikace – Omnidirectional Vision

2 <http://www.cis.upenn.edu/~kostas/omni.html>

3

4 Určitě zajímavou oblastí aplikace nelineárních projekcí je oblast počítačového vidění (computer vision), např. snímání okolí robota s úhlem snímání 360°.



Obr.5.20: Příklad aplikace nerovinné projekce

6

7 Je zřejmé, že:

- z důvodů nelineárních transformací kamera musí mít vysoké rozlišení
- pokud jsou použity vhodně dvě takové kamery, dostáváme možnosti reprezentace prostoru

10 Poznámka

11 Vynikající kniha o projekcích ve spojení s fotografickými technikami je:

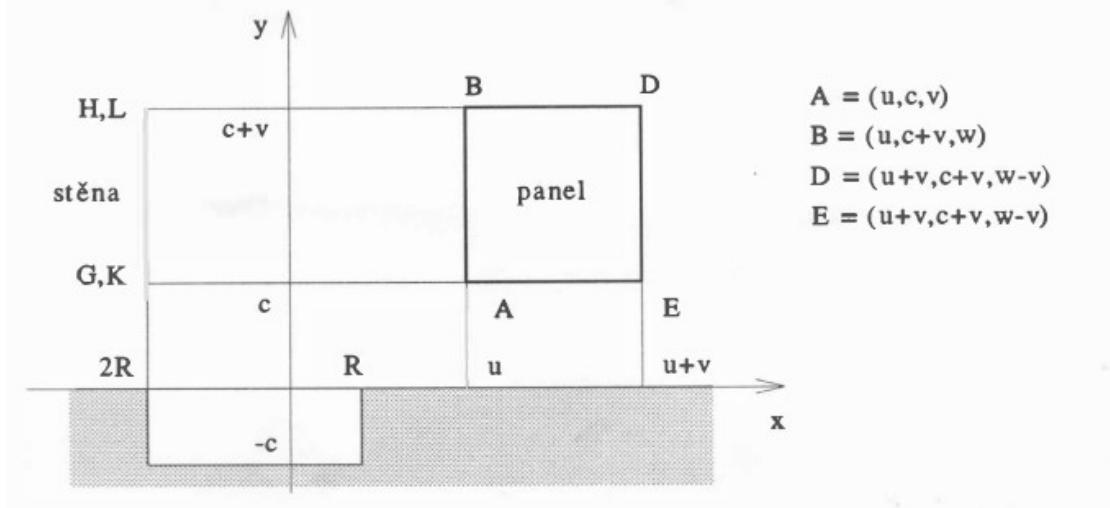
- 12 • Drs,L., Všetečka,J.: Objektivem počítače - geometrie speciálních fotografických technik, SNTL
13 1981

Příklad perspektivní projekce

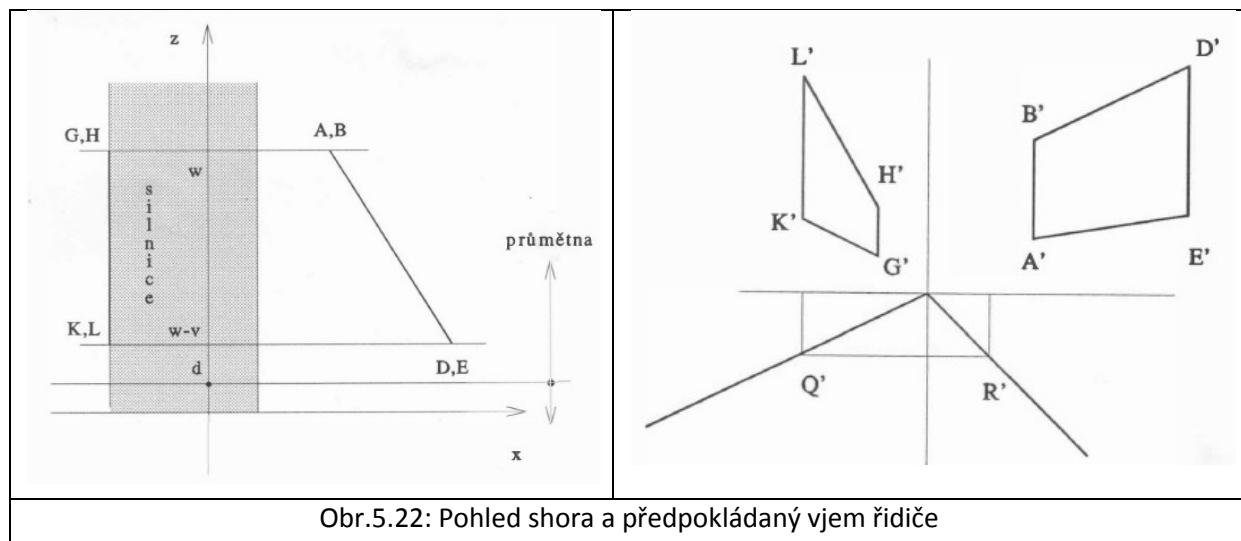
Pro lepší pochopení perspektivních projekcí uvažme následující úlohy.

3 Úloha 1

4 Uvažme následující úlohu, kdy chceme simulovat perspektivní pohled řidiče jedoucího na přímé
5 silnici. Pozice počátku bude vždy v počátku souřadného systému. Jaký bude výsledný pohled řidiče?



Obr.5.21: Situační nákres



Obr.5.22: Pohled shora a předpokládaný vjem řidiče

Úloha 2

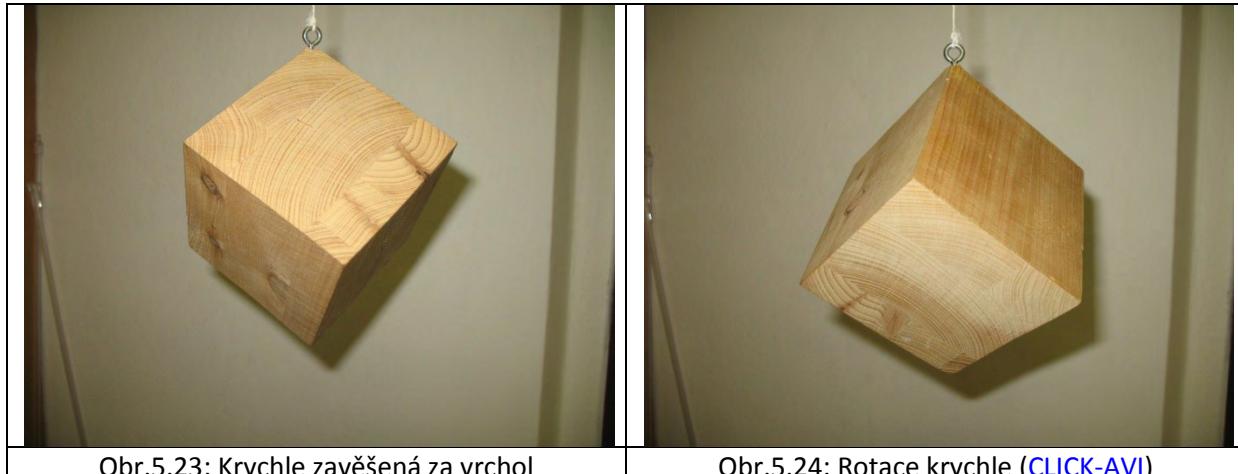
Uvažme krychli $\langle -1,1 \rangle \times \langle -1,1 \rangle$, tj. s těžištěm v počátku a ve standardní pozici. Vrcholy označme A, B, \dots, H . Jaký bude výsledek perspektivní projekce, pokud pozorovatel bude na ose z v pozicích $z = -2, z = -0.5, z = 0.5, z = 2$.

13 Výsledky nakreslete včetně hran a zkontrolujte korektnost. Zkuste odpovědět na otázku:

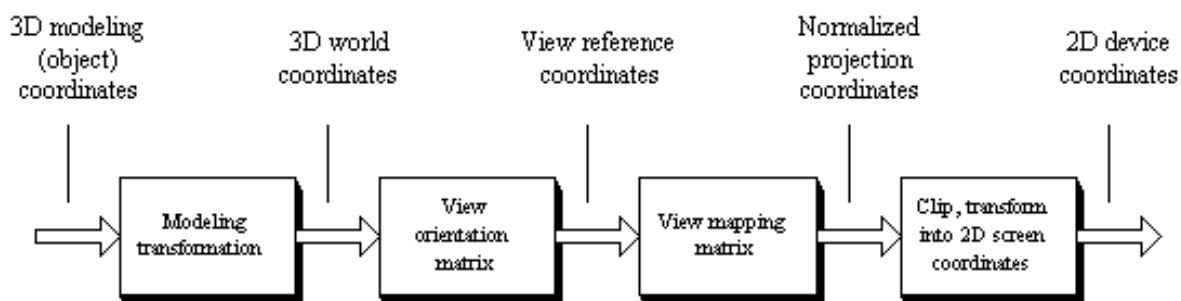
- co je zvláštního a proč?
 - uvidíme v pozici $z = -0,5$ a $z = 0,5$ šestistěn?

1 5.7.Projekce a faktor dynamiky

2 V úvodu pojednávajícím o projekcích byla úloha si představit, jak vypadá krychle zavěšená za vrchol
 3 na niti a její rotační obálka. Na Obr.5.23.a je ukázáno, jak zavěšená krychle za vrchol vypadá.



4
 5 Další úlohou bylo si představit, jak vypadá obálka rotující krychle.
 6
 7 Je zřejmé, že horní a spodní část je tvořena kuželem, zatímco prostřední část je tvořena rotačním
 8 hyperboloidem, který vytváří „po průmětu svislé“ hrany krychle (v prostoru svislé nejsou).
 9
 10 Prostor v „boku“ rotující krychle je skutečně volný a je možné jej využít, např. v případě mechanické
 11 součástky, na něco jiného.
 12
 13
 14 Ukázali jsme si, jak dynamický pohyb tělesa komplikuje naši představivost i pro jednoduchá tělesa
 15 jako je krychle. Takže lze očekávat, že v případě složitějších tvarů bude situace komplikovanější.
 16
 17 Projekcí z E^3 do E^2 je vlastně zakončena část geometrických transformací s objekty. Obr.5.25
 18 znázorňuje posloupnost zpracování geometrických objektů.
 19



20 Obr.5.25: Schéma zpracování geometrických entit

21
 22
 23 Po geometrických transformacích se části objektu, které jsou mimo zobrazovaný prostor, se musí
 24 odstranit. Tento proces se nazývá *ořezávání*, neboli *clipping*.
 25

1 6. Metody ořezávání v E^2 a E^3 a operace s n-úhelníky

2 Metody ořezávání používají algoritmy, které jsou aplikovatelné i v jiných oblastech. Ořezávání je
 3 „úzkým hrdlem“ grafického systému, neboť prakticky všechny elementy musí modulem ořezávání
 4 projít. Je nutné mít na paměti, že počet zpracovávaných elementů je vysoký, běžně $10^6 - 10^{12}$.

5 V následujícím textu budou uvedeny jen základní algoritmy s tím, že mnoho dalších algoritmů lze
 6 nalézt v dostupné literatuře a z digitálních knihoven dostupných ze ZČU, resp. přístupem přes VPN,
 7 např.

- 8 • Skala,V.: Algoritmy ořezávání, ISBN 978-80-86943-22-0, Plzeň, 2011
 9 (<http://herakles.zcu.cz/~skala/EDU-PUB/Habilitace-komplet.pdf>)
- 10 • Skala,V.: Algoritmy počítačové grafiky II, ISBN 978-80-86943-20-6, Plzeň, 2011
 11 (<http://herakles.zcu.cz/~skala/EDU-PUB/APG-2-OCR.pdf>)
- 12 • Bui Duc Huy: Algorithms for Line Clipping and Their Complexity, Ph.D.Thesis (supervizor Skala,V.),
 13 Plzeň, 1999, (http://herakles.zcu.cz/~skala/MSc/Diploma_Data/DIS_1999_Bui_Duc_Huy.pdf)

14 Algoritmy ořezávání jsou především určeny k odstranění těch částí, které jsou mimo kreslenou
 15 plochu, nebo mimo pohledovou pyramidu, resp. k realizaci nějakého výřezu z původního celku.

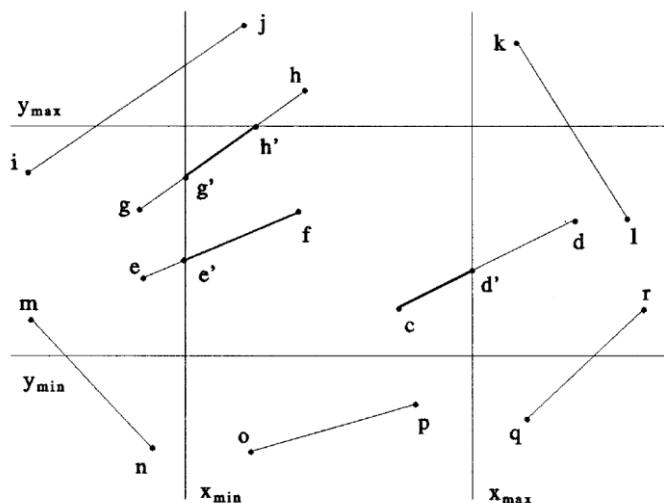
16 6.1. Algoritmy ořezávání v E^2

17 Algoritmy ořezávání v E^2 lze rozdělit podle různých hledisek. Téměř všechny algoritmy pracují
 18 v Eukleidovském souřadném systému, kromě algoritmu S-Clip pro ořezávání přímek a úseček, který
 19 ořezává přímky a úsečky v E^2 zadáné v implicitní formě nebo body v projektivním prostoru P^2 .
 20 Ořezávací okno, tj. obdélník, konvexní nebo nekonvexní n-úhelník, může být zadáno body
 21 v projektivním prostoru. Algoritmus S-Clip nevyžaduje operaci dělení pro převod do Eukleidovského
 22 prostoru.

23 Dále uváděné algoritmy nejsou optimalizovány – při jejich aplikaci je nutné je optimalizovat.

24 6.1.1. Cohen-Sutherland algoritmus

25 Cohen-Sutherlandův algoritmus je asi nejznámějším algoritmem pro ořezávání úsečky obdélníkovým
 26 oknem. Tato úloha je zdánlivě jednoduchá, ale pokud se má realizovat operace efektivně, pak nejde o
 27 zcela triviální algoritmus. Uvažme proto základní případy, které je nutné uvažovat.



28 29 Obr.6.1: Základní situace pro ořezávání úsečky obdélníkem

- 1 Je zřejmé, že počet základních typických situací není triviální a přímý výpočet a testování by byly
 2 výpočetně neúnosné. Cohen a Sutherland navrhli efektivní řešení spočívající především v „chytrém“
 3 kódování koncových bodů úsečky s následným využitím těchto kódů pro efektivní řešení úlohy.

1 0 0 1	1 0 0 0	1 0 1 0
0 0 0 1	0 0 0 0	0 0 1 0
0 1 0 1	0 1 0 0	0 1 1 0

Obr.6.2: Kódování koncových bodů úsečky $c = [top, bottom, right, left]$

Algoritmus je založen na rozdelení plochy na 9 oblastí, viz Obr.6.2, a následujících základních krocích:

- určení, do které oblasti koncové body úsečky patří
- detekce elementárních případů, kdy úsečka je *zcela uvnitř* nebo *zcela vně*
- řešení ostatních případů *iterativně*. Maximální počet kroků je 4 a jsou poměrně výpočetně složité

Určení kódu pro bod:

```

function CODE(x, y);
  byte c;
  { # c = [top, bottom, right, left] #
    if x < xmin then c := [0000] else if x > xmax then c := [0010] else c := [0000];
    if y < ymin then c := c lor [0100] else if y > ymax then c := c lor [1000];
    CODE := c
  }
}

```

Tímto postupem dostaneme kódová slova c_1 a c_2 pro oba koncové body úsečky x_1 a x_2 .

Je zřejmé, že elementárními případy jsou:

- oba koncové body jsou uvnitř okna, tj. $(c_1 \text{ lor } c_2) = 0$, kde **lor** je operace **or** po bitech
 - oba koncové body jsou nad, pod, vlevo nebo vpravo, tj. $(c_1 \text{ land } c_2) \neq 0$,
- kde **land** je operace **and** po bitech

Pokud nejde ani o jeden z těchto triviálních případů, je nutné počítat průsečíky s jednotlivými hranami okna a testovat, zda nový průsečík je vně okna, či nikoliv. Jednotlivé průsečíky se pak učí takto:

$(x_{min}, k(x_{min} - x_1) + y_1)$	Je-li bod vlevo
$(x_{max}, k(x_{max} - x_1) + y_1)$	Je-li bod vpravo
$(q(y_{max} - y_1) + x_1, y_{max})$	Je-li bod nad
$(q(y_{min} - y_1) + x_1, y_{min})$	Je-li bod pod

kde:

$$\begin{aligned}
 k &= \frac{y_2 - y_1}{x_2 - x_1} & q &= \frac{x_2 - x_1}{y_2 - y_1} \\
 &\& x_2 - x_1 \neq 0 & & & & & & (6.1) \\
 && & & & & & & \\
 \end{aligned}$$

```

1 Celý postup lze schematicky popsat takto:
2      $c_1 := \text{CODE}(x_1, y_1); c_2 := \text{CODE}(x_2, y_2);$ 
3     if ( $c_1 \text{ lor } c_2$ ) = 0 then { LINE( $x_1, y_1, x_2, y_2$ ); EXIT } # úsečka je zcela uvnitř #
4     if ( $c_1 \text{ land } c_2$ ) ≠ 0 then EXIT # úsečka je zcela vně #
5     repeat
6         if  $c_1 = 0$  then  $c := c_2$  else  $c := c_1;$ 
7         if ( $c \text{ land } 1$ ) ≠ 0 then # bod je vlevo #
8             {  $y := y_1 + (x_{min} - x_1)(y_2 - y_1)/(x_2 - x_1); x := x_{min}$  }
9             else if ( $c \text{ land } 2$ ) ≠ 0 then # bod je pravo #
10                {  $y := y_1 + (x_{max} - x_1)(y_2 - y_1)/(x_2 - x_1); x := x_{max}$  }
11                else if ( $c \text{ land } 4$ ) ≠ 0 then # bod je pod #
12                    {  $x := x_1 + (y_{min} - y_1)(x_2 - x_1)/(y_2 - y_1); y := y_{min}$  }
13                    else if ( $c \text{ land } 8$ ) ≠ 0 then # bod je nad #
14                        {  $x := x_1 + (y_{max} - y_1)(x_2 - x_1)/(y_2 - y_1); y := y_{max}$  };
15                        if  $c = c_1$  then {  $x_1 := x; y_1 := y; \text{CODE}(x_1, y_1)$  }
16                            else {  $x_2 := x; y_2 := y; \text{CODE}(x_2, y_2)$  };
17                        if ( $c_1 \text{ land } c_2$ ) ≠ 0 then EXIT # poslední část úsečky je zcela vně #
18        until ( $c_1 \text{ lor } c_2$ ) = 0;
19        LINE( $x_1, y_1, x_2, y_2$ );
20        EXIT
21
22
23 Z algoritmu je zřejmé, že algoritmus i přes zdánlivou jednoduchost řešeného problému má poměrně
24 vysokou výpočetní složitost. Je možné jej částečně optimalizovat, např. předpočtením hodnot  $k$  a  $q$ ,
25 neboť odpovídající výrazy se někdy používají  $2x$ .
26
27 Otázka - V jakém případě provede algoritmus celkem 4 cykly??
28
29 Zde je nutné upozornit:
30     • na možné problémy s přesností výpočtu, které mohou vzniknout při výpočtu průsečíku a
31       stanovení odpovídajícího kódového slova
32     • operace porovnání v pohyblivé řádové čárce je nejdelší numerickou operací hned po operaci
33       dělení
34
35 Úloha
36 Porovnejte Cohen-Sutherlandův (CS) algoritmus s následujícími algoritmy, které nahradí poslední část
37 CS algoritmu:
38     • pro netriviální situace se spočtou 4 průsečíky s hrany okna přímo a hodnoty parametru  $t$ 
39       se porovnávají, protože obdélník je konvexní n-úhelník, tak i parametry  $t$  musejí být cyklicky
40       uspořádány
41     • porovnejte CS algoritmus, navržený algoritmus bez použití vláken a paralelizovaný
42       algoritmus. Lze např. podle hodnot  $s_x$  a  $s_y$  rozhodnout, které hrany mohou být protnuty?
43 Nápoředa: Pokud jsou hrany obdélníka indexovány po směru, nebo proti směru hodinových ručiček,
44 hodnoty parametrů  $t$  mají stejnou indexaci a musí mít cyklicky rostoucí nebo klesající hodnotu.
45

```

Algoritmus TTTTTTTTTTTTT

Z algoritmu je zřejmé, že algoritmus i přes zdánlivou jednoduchost řešeného problému má poměrně vysokou výpočetní složitost. Je možné jej částečně optimalizovat, např. předpočtením hodnot k a q , neboť odpovídající výrazy se někdy používají $2x$.

Otázka - V jakém případě provede algoritmus celkem 4 cykly??

Zde je nutné upozornit:

- na možné problémy s přesností výpočtu, které mohou vzniknout při výpočtu průsečíku a stanovení odpovídajícího kódového slova
- operace porovnání v pohyblivé řádové čárce je nejdelší numerickou operací hned po operaci dělení

Úloha

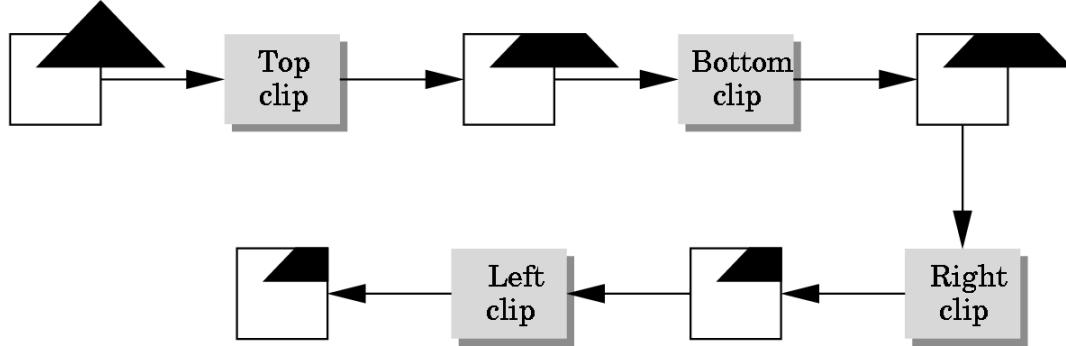
Porovnejte Cohen-Sutherlandův (CS) algoritmus s následujícími algoritmy, které nahradí poslední část CS algoritmu:

- pro netriviální situace se spočtou 4 průsečíky s hrany okna přímo a hodnoty parametru t se porovnávají, protože obdélník je konvexní n-úhelník, tak i parametry t musejí být cyklicky uspořádány
- porovnejte CS algoritmus, navržený algoritmus bez použití vláken a paralelizovaný algoritmus. Lze např. podle hodnot s_x a s_y rozhodnout, které hrany mohou být protnuty?

Nápoředa: Pokud jsou hrany obdélníka indexovány po směru, nebo proti směru hodinových ručiček, hodnoty parametrů t mají stejnou indexaci a musí mít cyklicky rostoucí nebo klesající hodnotu.

1 **6.1.2. Sutherland Hodgman algoritmus**

2 Algoritmus Sutherland-Hodgmanův je určen pro ořezávání trojúhelníka obdélníkovým oknem. Princip
3 algoritmu je jednoduchý a je založen na oříznutí zpracovávaného objektu polorovinou, Obr.6.3.



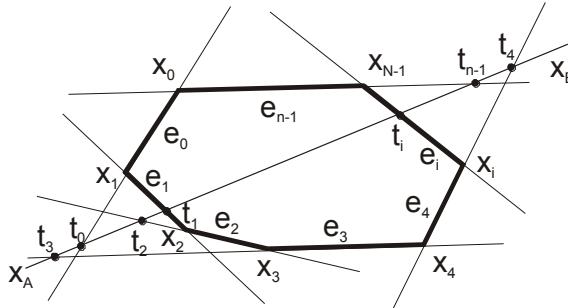
4 Obr.6.3:: Sutherland-Hodgmanův algoritmus v E^2

5 Je zřejmé, že při oříznutí trojúhelníka polorovinou je výsledkem obecně čtyřúhelník, který se rozdělí
6 na dva trojúhelkníky, které se následně zpracují.

7 Zásadní výhodou Sutherland-Hodgmanova algoritmu je jeho jednoduchost, *konstantní rychlosť*
8 zpracování jednoho trojúhelníka jedním blokem (to je důležité z hlediska časování při implementaci
9 v hardwaru) snadná rozšiřitelnost pro ořezávání v E^3 a snadná realizovatelnost v hardwaru.

10 **6.1.3. Cyrus Beck algoritmus**

11 Cyrus-Beck algoritmus (CB) je algoritmus, který umožňuje ořezávání přímky i úsečky konvexním
12 n-úhelníkem a je snadno modifikovatelný i pro případ přímky, resp. úsečky v prostoru. Algoritmus je
13 složitosti $O(n)$, kde n je počet hran n-úhelníka.



14 Obr.6.4: Průsečíky počítané CB algoritmem

15 Přímka je zadána dvěma body X_A a X_B a tedy v parametrické formě:

$$X(t) = X_A + (X_B - X_A)t = X_A + S \quad (6.2)$$

16 Předpokládejme hypotetického pozorovatele, který je v bodě $t = -\infty$ a dívá se ve směru vektoru S .

17 Předpokládejme bez újmy na obecnosti, že pozorovatel je v bodě X_A .

18 Algoritmus je založen na:

- 19 • rozdelení hran n-úhelníka do dvou skupin, a to: na hrany, které pozorovatel „vidí“, a na
20 hrany, které „nevidí“
- 21 • výpočtu průsečíků přímky s jednotlivými hranami a pak:
 - 22 ○ u hran, které vidí, hledá *maximální* hodnotu t_{min} parametru u příslušných průsečíků
 - 23 ○ u hran, které nevidí, hledá *minimální* hodnotu t_{max} parametru u příslušných
24 průsečíků
- 25 • pokud $t_{max} < t_{min}$, tj. interval $\langle t_{min}, t_{max} \rangle = \emptyset$, pak daná přímka konvexní n-úhelník
26 neprotíná

```

1 V případě ořezávání úsečky je nutné určit, zda vypočtený interval má neprázdný průnik s intervalem
2  $\langle 0,1 \rangle$ , tj. s danou úsečkou. Pokud  $\langle 0,1 \rangle \cap \langle t_{min}, t_{max} \rangle = \emptyset$ , pak úsečka nemá s n-úhelníkem průsečík.
3 V opačném případě se body průsečíku dopočtou z parametrické rovnice přímky pro  $t \in \langle 0,1 \rangle \cap$ 
4  $\langle t_{min}, t_{max} \rangle$ .
5
6 procedure CB; # input:  $\mathbf{x}_A, \mathbf{x}_B$  #
7 {  $\mathbf{s} := \mathbf{x}_B - \mathbf{x}_A$ ;
8    $t_{min} := -\infty$ ;  $t_{max} := \infty$ ; #  $t_{min} := 0$ ;  $t_{max} := 1$ ; in case of a line segment #
9   for i:=0 to N-1 do
10  {
11     $\mathbf{q} := \mathbf{n}_i^T \mathbf{s}$ ;
12    if  $q < 0$  then
13      { # the i-th edge can be seen from infinity in the negative direction of  $-\mathbf{s}$  #
14         $t := -(\mathbf{n}_i^T \mathbf{x}_A + c_i)/q$ ;  $t_{min} := \max(t, t_{min})$ ;
15      } else
16        if  $q > 0$  then
17          { #the i-th edge can be seen from #
18            #infinity in the direction of  $\mathbf{s}$  #
19             $t := -(\mathbf{n}_i^T \mathbf{x}_A + c_i)/q$ ;  $t_{max} := \min(t, t_{max})$ ;
20          } else solve a special case  $\mathbf{n}_i^T \mathbf{s} = 0$ 
21      };
22    if  $t_{min} < t_{max}$  then
23      {  $\mathbf{x}_A := \mathbf{x}_A + \mathbf{s} t_{min}$ ;  $\mathbf{x}_B := \mathbf{x}_A + \mathbf{s} t_{max}$ ; output ( $\mathbf{x}_A, \mathbf{x}_B$ ) }
24  } # CB algoritmus #

```

Obr.6.5: Princip Cyrus-Beck algoritmu

6.1.4. Algoritmus se složitostí $O(\lg n)$

Cyrus-beck algoritmus pro ořezávání přímky nebo úsečky konvexním n-úhelníkem využívá konvexitu pouze k tomu, aby vyhodnocoval pouze 2 možné průsečíky. V případě E^2 jsou vrcholy n-úhelníka uspořádané ve směru (clockwise) nebo v protisměru (anticlockwise) hodinových ručiček. Princip duality v E^2 , viz kap.3.3 (Dualita a její aplikace) říká, že bod je duální přímce a naopak, průnik je duální sjednocení a naopak. Takže problém, zda přímka protíná konvexní n-úhelník v E^2 je duální úloze, zda je bod uvnitř konvexního n-úhelníka. Tento problém je složitosti $O(\lg n)$. Vlastní algoritmus ořezávání je založen na půlení intervalu indexů vrcholů daného konvexního n-úhelníka. Vzhledem k větší složitosti algoritmu, vlastní algoritmus zde není uveden. Podrobný popis viz:

- Skala, V.: $O(\lg N)$ Line Clipping Algorithm in E2, Computers & Graphics, Pergamon Press, Vol.18, No.4, 1994

Je nutné zdůraznit, že $O(\lg n)$ algoritmus je pro $n > 3$ rychlejší než Cyrus-Beck algoritmus. Pokud konvexní n-úhelník má např. 1024 vrcholů, pak Cyrus-Beck vyžaduje 1024 kroků, zatímco $O(\lg n)$ algoritmus vyžaduje pouze 10 kroků, což je podstatný rozdíl.

V tomto případě jde o názornou ukázkou, jak aplikace principu duality může vést k návrhu nových efektivních algoritmů. Poznamenejme, že aplikací principu duality se však výpočetní složitost daného problému nemění.

44

1 6.1.5. Algoritmus Smart-Clip (S-Clip)

2 Až dosud jsme se zabývali algoritmy, které byly navrženy pro řešení problému v Eukleidovském
 3 prostoru. Nicméně souřadnice bodů jsou reprezentovány v homogenních souřadnicích a po aplikaci
 4 geometrických transformací nebo jiných výpočtů hodnota homogenní souřadnice je obecně $w \neq 1$.
 5 To znamená, že pro účely ořezávání by bylo nutné provést dělení souřadnicí w , tedy 3 operace dělení
 6 pro $\langle x, y, z \rangle$ na 1 bod. Při počtu zpracovávaných bodů $10^5 - 10^{10}$ by tato konverze vyžadovala
 7 neúměrný výpočetní výkon.

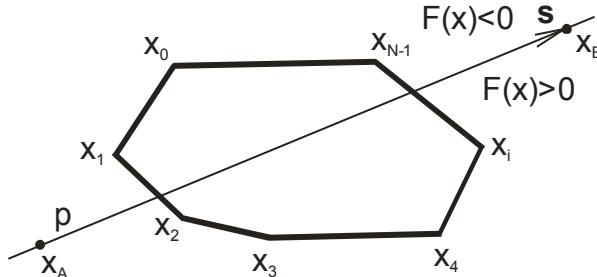
8 Algoritmus S-Clip je algoritmus, který umožňuje ořezávání přímky nebo úsečky obdélníkovým
 9 oknem, konvexním n-úhelníkem se složitostí $O(n)$. Pro nekonvexní n-úhelník je pak složitost
 10 $O(n + m \lg m)$, neboť je nutná operace řazení nad m nalezenými průsečíky.

11
 12 Algoritmus je založen na vyhodnocení pozice vrcholů n-úhelníka vzhledem k přímce dané dvěma
 13 body a testuje se, ve které polorovině vrcholy leží. Na základě této klasifikace se určí indexy hran,
 14 které přímka protíná, a následně se spočtou průsečíky s těmito hranami.

15 Předpokládejme, že je dán konvexní n-úhelník a přímka p rovnici $F(\mathbf{x}) = aX + bY + c = 0$,
 16 resp. $F(\mathbf{x}) = ax + by + cw = \mathbf{a}^T \mathbf{x} = 0$ pro projektivní případ. Přímka p rozdělí prostor na dvě
 17 poloroviny, tj. pro $F(\mathbf{x}) > 0$ a pro $F(\mathbf{x}) \leq 0$. Hodnota funkce $F(\mathbf{x})$ může být vyhodnocena pro každý
 18 vrchol n-úhelníka. Pro i -tý vrchol dostáváme i -tý bit kódového slova \mathbf{c} jako:

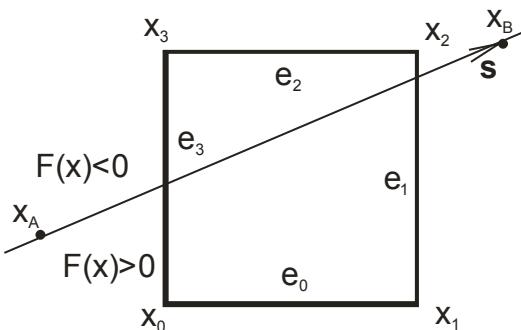
$$c_i = \begin{cases} 1 & \text{pokud } F(\mathbf{x}_i) > 0 \\ 0 & \text{jinak} \end{cases} \quad i = 0, \dots, n-1 \quad (6.3)$$

19 To znamená, že každý vrchol je ohodnocen, zda leží „napravo“ nebo „naľavo“ od přímky p .



Obr.6.6: Ohodnocení vrcholů n-úhelníka vůči přímce p

20
 21 V následujícím se omezíme na obdélníkové okno, bez újmy na principu, či na obecnost algoritmu.



Obr.6.7: Ořezávání obdélníkovým oknem

22
 23 V případě obdélníkového okna je kódové slovo \mathbf{c} určeno:

$$\mathbf{c} = [c_3, c_2, c_1, c_0]^T$$

24 a kód \mathbf{c} nyní určuje jednoznačně pozici přímky vůči vrcholům ořezávacího n-úhelníka. Pro jednotlivé
 25 hodnoty kódových slov \mathbf{c} lze explicitně určit hrany, které budou přímkou p protnuty, viz Tab. 6.1.

- 1 Indexy hran protnutých přímkou p jsou uloženy v TAB1 a TAB2 a lze nahlédnout, že tyto hodnoty jsou
 2 „symetrické“, což je dáno implicitním popisem přímky p , kdy $F(\mathbf{x}) = 0$ a $-F(\mathbf{x}) = 0$ definují stejnou
 3 přímku p . Proto je také nepodstatná orientace vrcholů okna.

C	c_3	c_2	c_1	c_0	TAB1	TAB2	MASK
0	0	0	0	0	None	None	None
1	0	0	0	1	0	3	0100
2	0	0	1	0	0	1	0100
3	0	0	1	1	1	3	0010
4	0	1	0	0	1	2	0010
5	0	1	0	1	N/A	N/A	N/A
6	0	1	1	0	0	2	0100
7	0	1	1	1	2	3	1000
8	1	0	0	0	2	3	1000
9	1	0	0	1	0	2	0100
10	1	0	1	0	N/A	N/A	N/A
11	1	0	1	1	1	2	0010
12	1	1	0	0	1	3	0010
13	1	1	0	1	0	1	0100
14	1	1	1	0	0	3	0100
15	1	1	1	1	None	None	None

Tab. 6.1: Rozhodovací abulka pro S-Clip

kde:

- N/A (non-applicable case) označuje případy, které nemohou principiálně nastat, neboť kombinace $\mathbf{c} = [0,1,0,1]^T$ by značila, že přímka p má 4 průsečíky s konvexním n-úhelníkem, což pro konvexní n-úhelník není možné
- None označuje případ, kdy přímka p neprotíná daný n-úhelník
- význam sloupce MASK bude vysvětlen později – použije se pro rozšíření algoritmu S-Clip pro ořezávání úsečky, tj. algoritmu S-Clip_Segment

Výše uvedenou tabulku lze vygenerovat automaticky pro konvexní n-úhelník, neboť pokud $\mathbf{c}_i \neq \mathbf{c}_{i+1}$, pak hrana $\mathbf{x}_i \mathbf{x}_{i+1}$ bude přímkou p protnuta. Délka tabulky je pak 2^n , kde n je počet vrcholů n-úhelníka.

```

procedure S-CLIP_Line (  $\mathbf{x}_A, \mathbf{x}_B$ );
# input:  $\mathbf{x}_A = [x_A, y_A: w_A]^T$   $\mathbf{x}_B = [x_B, y_B: w_B]^T$   $\mathbf{p} = [a, b: c]^T$   $n = 4$  #
{
#1#  $\mathbf{p} = \mathbf{x}_A \times \mathbf{x}_B$ ; #  $p: ax + by + cw = 0$  #
#2# for  $k := 0$  to  $n - 1$  do # cyklus může být realizován paralelně #
#3# if  $\mathbf{p}^T \mathbf{x}_k \geq 0$  then  $c_k = 1$  else  $c_k = 0$ ; #  $\mathbf{x}_k = [x_k, y_k: w_k]^T$  #
#4# if  $\mathbf{c} = [0, \dots, 0]^T$  or  $\mathbf{c} = [1, \dots, 1]^T$  then EXIT; # numerický test if  $\mathbf{c}=0$  or  $\mathbf{c}=2^n-1$  then EXIT #
#5#  $i := \text{TAB1}[\mathbf{c}]$ ;  $j := \text{TAB2}[\mathbf{c}]$ ;
#6#  $\mathbf{x}_A := \mathbf{p} \times \mathbf{e}_i$ ;  $\mathbf{x}_B := \mathbf{p} \times \mathbf{e}_j$ ; #  $\mathbf{e}_i$  je i-tá hrana n-úhelníka #
#7# DRAW (  $\mathbf{x}_A, \mathbf{x}_B$  )
}
# S-CLIP_Line #

```

Obr.6.8: Alg. Clip-LS - Algoritmus pro konvexní n-úhelník

Úloha

Napište obecný algoritmus generující tabulky TAB1 a TAB2 pro $n \geq 4$.

1 Z výše uvedeného algoritmu je zřejmé, že body definující přímku, resp. úsečku a vrcholy n-úhelníka
 2 jsou v homogenních souřadnicích, přičemž homogenní složky souřadnic bodů mohou být $w \neq 1$. Toto
 3 je důsledkem použití implicitní formulace. Není tedy zapotřebí použít operaci dělení pro převod do
 4 Eukleidovského prostoru.

5 Výše uvedený algoritmus je jednoduchý, elegantní, a pokud je implementován např. na GPU
 6 architektuře, pak vektorový a skalární součin je hardwarovou instrukcí. Otázkou je, zda algoritmus lze
 7 dále optimalizovat, neboť lze využít převodu primitiv z $WC \rightarrow NDC$ a $NDC \rightarrow WC$ souřadných
 8 systémů.

9

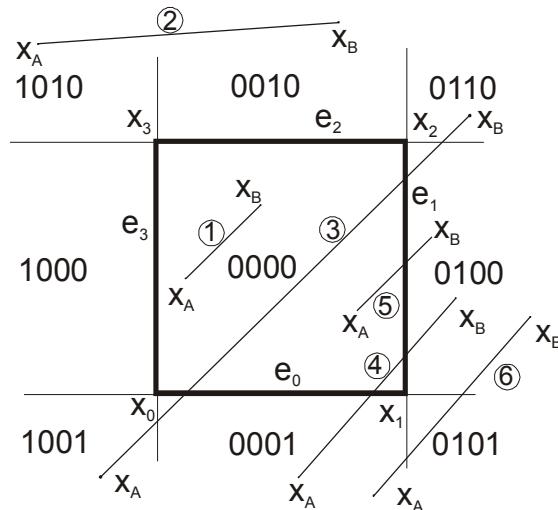
10 Optimalizace algoritmu pro okno $\langle -1, 1 \rangle \times \langle -1, 1 \rangle$

11 U geometrických transformací a projekcí byl zaveden princip NDC prostoru. Analýzou algoritmu výše
 12 uvedeného je zřejmé, že vektorové součiny v přiřazeních $x_A := p \times e_i$ a $x_B := p \times e_j$ mohou být
 13 podstatně zjednodušeny, neboť přímky hran obdélníka mohou být vyjádřeny tak, že jejich koeficienty
 14 nabývají hodnot $\{-1, 0, 1\}$. Není tedy zapotřebí násobení pro realizaci vektorového součinu. Takto lze
 15 ušetřit až 6 násobení na každou přímku.

16

17 Modifikace algoritmu S-Clip pro úsečky

18 Výše uvedený algoritmus S-Clip je určen pro ořezávání přímek. V počítačové grafice se však většinou
 19 zpracovávají „konečné“ elementy, např. úsečky, trojúhelníky apod. Je tedy nutné se podívat, jak
 20 algoritmus S-Clip lze modifikovat pro případ ořezávání úseček. Obdobně jako Cohen-Sutherland
 21 algoritmus použijeme kódování koncových bodů ořezávané úsečky.



22

23 Obr.6.9: Klasifikace koncových bodů ořezávané úsečky

24 Modifikace algoritmu S-Clip pro ořezávání úsečky je založena na jednoduchém principu. Dříve
 25 uvedený algoritmus S-Clip určuje přímo, které dvě hrany jsou protnuty přímkou, na níž daná úsečka
 26 leží. Pokud se eliminují elementární případy, kdy úsečka je zcela uvnitř nebo vně, pak pokud koncový
 27 bod leží uvnitř, je nutné vybrat správnou hranu již dříve z dvou vybraných hran, se kterou se vlastní
 28 průsečík s úsečkou spočte. Např. pokud přímka protíná hrany e_1 a e_3 a c , je kód ohodnocení vrcholu
 29 ořezávacího okna, pak příkazy:

```
30 if cB land MASK[c] ≠ 0
31   then intersection p & e1
32   else intersection p & e3
```

33 určí odpovídající průsečík. Tabulka MASK byla již uvedena dříve, viz Tab. 6.1.

```

1  function CODE ( $x$ );
2  {
3       $c := [0000]$ ;
4      if  $x < x_{\min}$  then  $c := [1000]$ 
5          else if  $x > x_{\max}$  then  $c := [0100]$ ;
6      if  $y < y_{\min}$  then  $c := c \text{ lor } [0001]$ 
7          else if  $y > y_{\max}$  then  $c := c \text{ lor } [0010]$ ;
8      CODE :=  $c$ 
9  } # CODE #;
10
11 procedure S-Clip_Segment ( $x_A, x_B$ ); # input:  $x_A, x_B$  #
12 # C_LS - Clipping Line Segment  $x_A, x_B$  – can be in homogeneous coordinates #
13 # the EXIT statement ends the procedure #
14 # land / lor – bitwise operations and / or #
15 { # end-points classifications#
16      $c_A := \text{CODE} (x_A)$ ;  $c_B := \text{CODE} (x_B)$ ;
17     # detection of trivial cases #
18     # case 1 – line segment inside #
19     if ( $c_A \text{ lor } c_B$ ) = 0 then
20         { output ( $x_A, x_B$ ); EXIT }
21     # case 2 – line segment outside – just EXIT#
22     if ( $c_A \text{ land } c_B$ ) ≠ 0 then EXIT;
23     # all trivial cases solved #
24     # compute coefficients of the line p #
25      $p := x_A \times x_B$ ; #  $ax+by+c=0$ ;  $p = [a,b:c]^T$  #
26     for k := 0 to 3 do #  $x_k=[x_k,y_k:1]^T$  #
27         if  $p^T x_k \geq 0$  then  $c_k := 1$  else  $c_k := 0$ ;
28     #  $c = [c_3, c_2, c_1, c_0]^T$  #
29     if  $c = [0000]^T$  or  $c = [1111]^T$  then EXIT;
30     # the line segment lies outside of the window - it distinguishes also the cases 4 and 6 #
31     # there MUST be one intersection at least #
32     i := TAB1[c]; j := TAB2[c];
33     if  $c_A \neq 0$  and  $c_B \neq 0$ 
34     then
35         # there are two intersections #
36         {  $x_A := p \times e_i$ ;  $x_B := p \times e_j$  }
37         # vector  $e_i$  is pre-defined for the i-th edge #
38     else # there is only one intersection point #
39     if  $c_A = 0$  then #  $x_B$  is outside #
40         { if  $c_B \text{ land } \text{MASK}[c] \neq 0$  then  $x_B := p \times e_i$  else  $x_B := p \times e_j$  }
41     else if  $c_B = 0$  then #  $x_A$  is outside #
42         { if  $c_A \text{ land } \text{MASK}[c] \neq 0$  then  $x_A := p \times e_i$  else  $x_A := p \times e_j$  };
43     output ( $x_A, x_B$ )
44 end # S-Clip_Segment #
45

```

Obr.6.10: Modifikace algoritmu S-Clip pro ořezávání úsečky obdélníkem

1 Z výše uvedeného algoritmu je zřejmé, že S-Clip algoritmus pro ořezávání přímky a úsečky je
 2 jednoduchý, rozšířitelný na konvexní n-úhelník. Tabulky TAB1, TAB2 a MASK se generují podle zadané
 3 hodnoty n , tj. počtu vrcholů daného n-úhelníka.

4

5 **Příklad**

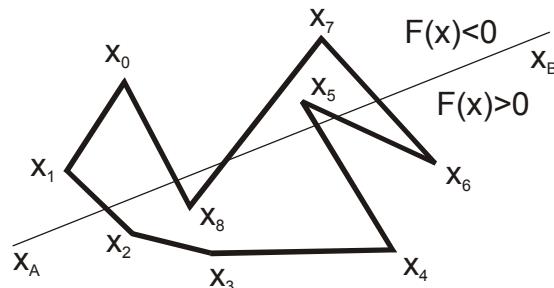
6 Modifikujte algoritmus S-Clip pro případ konvexního n-úhelníka

7

8

9 V případě *nekonvexního* n-úhelníka je nutné přímku, resp. úsečku pro vlastní výpočet reprezentovat
 10 pomocí parametrického vyjádření a hodnoty parametru t jednotlivých průsečíků seřadit.

11



12

13

Obr.6.11: Ořezávání přímky nekonvexním n-úhelníkem

14

15

Výsledná složitost algoritmu je pak $O(N + m \lg m)$, kde m je počet průsečíků, které se musí seřadit.

16

17

Podrobný popis algoritmu S-Clip a jeho modifikací viz:

18

19

- Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, ISSN 0178-2789, Vol.21, No.11, pp.905-914, Springer Verlag, 2005

20

21

- Skala,V.: S-Clip E2: A New Concept of Clipping Algorithms, Poster, SIGGRAPH Asia, ISBN 978-1-4503-1757-3, 2012, Singapore, 2012

22

23

24

Existuje mnoho dalších algoritmů a jejich popis lze nalézt v odborné literatuře, např. viz kap. 21
 (Geometrická algebra)

25

26

27

Čtenář si poslechne záznam odborné přednášky ze záznamu:

28

29

- Lengyel,E.: A Big Mathematical Picture for Computer Graphics, WSCG 2012 (65 min.)
[\(CLICK off-line\)](#)

30

31

6.2. Algoritmy ořezávání v E^3

Algoritmy ořezávání v E^3 jsou určeny k odstranění těch částí základních grafických primitiv, které jsou mimo zobrazovanou oblast, tj. převážně částí úseček a trojúhelníků. V následujícím budou uvedeny jen základní algoritmy.

6.2.1. Cohen-Sutherland algoritmus

Rozšíření Cohen-Sutherland algoritmu do prostoru E^3 je vcelku triviální. Ořezávacím elementem je kvádr, jehož hrany jsou rovnoběžné s osami souřadného systému. Kódové slovo pozice koncových bodů úsečky c má nyní 6 bitů, tj. $c = [near, far, left, right, bottom, top]^T$, kde bit $near = 1$, pokud bod je před kvádrem, kterým ořezáváme. Bit $far = 1$, pokud bod je za kvádrem, kterým ořezáváme. Průsečíky přímky s kvádrem se počítají jako průsečík přímky s plochami kvádru a odpovídajícím způsobem je kód modifikován.

6.2.2. Cyrus-Beck algoritmus

Modifikace Cyrus-Beck algoritmu je také jednoduché. V případě E^2 byl konvexní n-úhelník tvořen vlastně průnikem polorovin, které jednotlivé hrany n-úhelníka definovaly. V případě E^3 je konvexní mnohostěn definován jako průnik poloprostorů, na nichž stěny konvexního n-úhelníka leží. Takže v algoritmu stačí zaměnit pojem hrany za pojem stěna a normála hrany za pojem normála roviny, resp. stěny. Algoritmus je stejný, pouze výpočet parametru musí obsahovat i koeficient roviny pro z složku, takže:

$$t_i = -\frac{\mathbf{n}_i^T \mathbf{x}_A + d_i}{\mathbf{n}_i^T \mathbf{s}} \quad (6.4)$$

kde každá rovina ρ_i je dána rovnicí $a_i x + b_i y + c_i z + d_i = 0$, tj. $\mathbf{n}_i^T \mathbf{x} + d = 0$. Zbytek celého algoritmu je stejný.

Úloha

Formulujte CB algoritmus pro projektivní prostor, tj. když body úsečky jsou dány obecně v homogenních souřadnicích, tj. $\mathbf{x} = [x, y, z: w]^T$, vektor $\mathbf{s} = [s_x, s_y, s_z: s_w]^T$ je také určen v projektivním prostoru, tj. lineární interpolace s nelineární monotonné parametrizací, viz kap.10.1 (Lineární interpolace). Ukažte, že není zapotřebí operace dělení. Diskutujte výhody a nevýhody.

6.2.3. Sutherland-Hodgman algoritmus

Sutherland-Hodgmanův (S-H) algoritmus pro E^3 je vlastně jednoduchým rozšířením S-H algoritmu pro E^2 , kdy místo průsečíku dané úsečky, resp. hrany trojúhelníka, s hranou obdélníka se počítá průsečík úsečky se stěnou kvádru, obvykle jednotkové krychle, kdy výpočet je velmi jednoduchý. Toto se využívá v grafických akcelerátorech, kdy ořezávání úseček jednou rovinou krychle realizuje hardwarový modul, a tyto moduly jsou řazeny za sebou. Takže celý blok realizující ořezávání se skládá z 6 modulů, které jsou v sérii za sebou. Toto řešení má své výhody, a to:

- jednoduché HW řešení a
- stejnou dobu zpracování v každém modulu pro zpracovávané grafické elementy

```
1 Algoritmus S-H lze popsat sekvencí
2 (http://www.sunshine2k.de/coding/java/SutherlandHodgman/SutherlandHodgman.html)
3
4 for each clipping edge do
5     for (i = 0; i < Polygon.length -1; i++)
6         Pi = Polygon.vertex[i];
7         Pi+1 = Polygon.vertex[i+1];
8         if (Pi is inside clipping region)
9             if (Pi+1 is inside clipping region)
10                clippedPolygon.add(Pi+1)
11            else
12                clippedPolygon.add(intersectionPoint(Pi, Pi+1, currentEdge))
13            else
14                if (Pi+1 is inside clipping region)
15                    clippedPolygon.add(intersectionPoint(Pi, Pi+1, currentEdge))
16                    clippedPolygon.add(Pi+1)
17        end for
18        Polygon = clippedPolygon // keep on working with the new polygon
19    end for
20
21
```

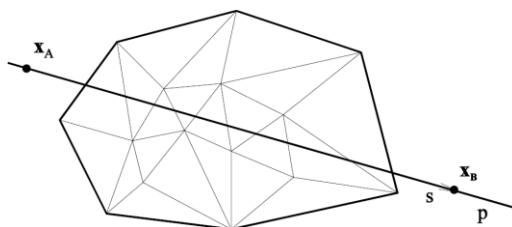
Obr.6.12: Sutherland Hodmanův algoritmus

Zajisté vznikne otázka, zda existuje algoritmus pro ořezávání přímky, resp. úsečky v E^3 s menší složitostí než $O(N)$. Je nutné si uvědomit, že v případě E^2 jsme u algoritmu $O(\lg N)$ využívali nejen vlastnosti konvexity, ale především uspořádanosti indexů. Je to podobné jako při hledání kořene rovnice $F(x) = 0$ metodou půlení intervalu, kdy využíváme uspořádanosti hodnot na ose x . Bohužel v případě E^3 podobné uspořádání již není definováno. Nicméně existuje algoritmus se složitostí nižší než $O(N)$, ale jde však o složitost očekávanou.

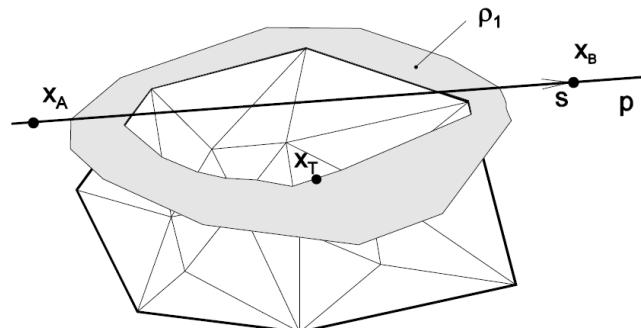
31

6.2.4. Algoritmus se složitostí $O_{expected}(\sqrt{N})$

1 Algoritmus s očekávanou složitostí $O_{expected}(\sqrt{N})$ je založen na celkem jednoduchém principu.
 2 Představme si případ, kdy přímka p je dána body x_A a x_B a konvexní mnohostěn je dán
 3 trojúhelníkovou sítí, tj. nikoliv jako množina trojúhelníků. Je zřejmé, že vlastnosti trojúhelníkové sítě,
 4 tj. znalost sousedních trojúhelníků, bychom měli využít ke snížení výpočetní složitosti.



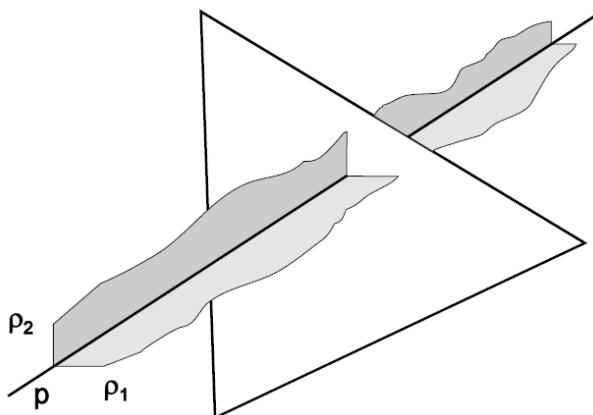
Obr.6.13: Přímka protínající konvexní mnohostěn



Obr.6.14: Přímka s bodem na povrchu určuje rovinu

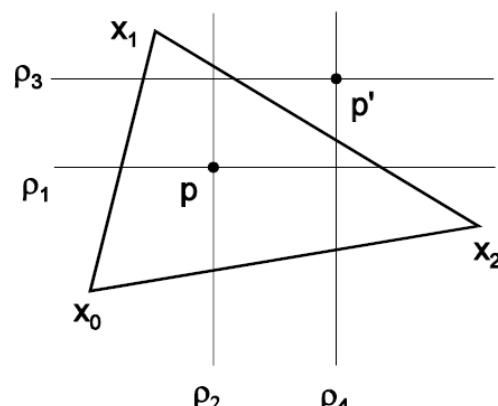
6 Pokud vybereme libovolný bod x_T , který neleží na přímce p , pak body x_A , x_B , x_T definují rovinu ρ ,
 7 která zajisté protíná i ty trojúhelníky, které protíná přímka p , viz Obr.6.14. To znamená, že pokud za
 8 bod x_T vybereme těžiště téměř libovolného trojúhelníka, můžeme se pohybovat „vlevo“ nebo
 9 „vpravo“ po sousedních trojúhelnících, které jsou takovou rovinou protnuty a označme tuto rovinu
 10 ρ_1 . Je zřejmé, že počet protnutých trojúhelníků bude zhruba \sqrt{N} , kde N je počet trojúhelníků daného
 11 mnohostěnu. Vzhledem k tomu, že test průsečíku přímky s trojúhelníkem je relativně složitý, je
 12 vhodné daný postup ještě opakovat pro rovinu ρ_2 , která je kolmá k rovině ρ_1 , viz Obr.6.15. Získáváme
 13 tak podle indexů neuspořádanou množinu Q_1 , resp. Q_2 indexů trojúhelníků protnutých rovinou ρ_1 ,
 14 resp. ρ_2 . Indexy trojúhelníků, které mohou být protnuty přímkou p , jsou pak dány průnikem daných
 15 množin, tj. $Q_1 \cap Q_2$. Je nutné upozornit, že může nastat případ „falešné“ detekce průsečíku viz
 16 Obr.6.16, což je nutné vyloučit následným podrobným testem. Je zřejmé, že algoritmus má
 17 očekávanou složitost $O_{expected}(\sqrt{N})$. Tedy pokud mnohostěn má 10^4 trojúhelníků a je definován
 18 trojúhelníkovou sítí, pak lze očekávat potřebu cca 10^2 testů na existenci průsečíku trojúhelníka
 19 s rovinou a několik, obvykle 2-8 podrobných testů na průsečík přímky s trojúhelníkem.
 20

21



Obr.6.15: Přímka jako průsečnice dvou rovin

22 Vlastní algoritmus je pak popsán dále.



Obr.6.16: Možné situace

```

1  procedure CLIP_3D_MOD (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  );
2  begin
3       $t_{min} := 0.0$ ;    $t_{max} := 1.0$ ;    $i := 1$ ;    $s := \mathbf{x}_B - \mathbf{x}_A$ ;
4      { for the line clipping  $t_{min} := -\infty$ ;  $t_{max} := \infty$ ; }
5      {  $\rho_1: F_1(\mathbf{x}) = A_1 x + C_1 z + D_1 = 0$             $\rho_2: F_2(\mathbf{x}) = B_2 y + C_2 z + D_2 = 0$  }
6      for  $k := 1$  to  $N_v$  do {  $N_v$  number of vertices }
7           $Q_k := \text{sign}(F_1(\mathbf{x}_k))$ ; {  $Q_k$  is a vector of int or char types }
8      for  $i := 1$  to  $N$  do
9          begin
10             {  $\mathbf{x}_k$  means a  $k$ -th vertex of the  $i$ -th triangle }
11             { INDEX( $i,k$ ) gives the index of  $k$ -th vertex of the  $i$ -th triangle, i.e.  $\mathbf{x}_k = \mathbf{x}_{\text{INDEX}(i,k)}$  }
12             if  $Q_{\text{INDEX}(i,0)} = Q_{\text{INDEX}(i,1)}$  then
13                 if  $Q_{\text{INDEX}(i,0)} = Q_{\text{INDEX}(i,2)}$  then goto 1;
14                     { do nothing  $\rho_1$  does not intersect the  $i$ -th triangle }
15                     if  $\text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,0)})) = \text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,1)}))$  then
16                         if  $\text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,0)})) = \text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,1)}))$  then goto 1;
17                         { both planes  $\rho_1, \rho_2$  intersect the  $i$ -th triangle }
18                         { detailed test finds a value  $t_{min}$  and  $t_{max}$  using a single step of the CB algorithm }
19                         { ----- }
20                         {  $\mathbf{n}_i$  must point out of the given convex polyhedron }
21                          $\xi := s^T \mathbf{n}_i$ ;            $s_i := \mathbf{x}_i - \mathbf{x}_A$ ;
22                         if  $\xi <> 0.0$  then
23                             begin  $t := s_i^T \mathbf{n}_i / \xi$ ;
24                             if  $\xi > 0.0$  then  $t_{max} := \min(t, t_{max})$ 
25                             else  $t_{min} := \max(t, t_{min})$ ;
26                         end
27                         else Special case solution; { line is parallel to a facet }
28                         { ----- }
29     1:     $i := i + 1$ ;
30     end;
31     { if  $t_{min} > t_{max}$  then no intersection point exists }
32     if  $t_{min} \leq t_{max}$  then SHOW_LINE ( $\mathbf{x}(t_{min})$ ,  $\mathbf{x}(t_{max})$ );
33 end { of CLIP_3D_MOD };
34
35
36
37 Vzhledem k tomu, že algoritmus je založen na implicitním popisu  $F_i(\mathbf{x}) = 0$  je možné jej snadno
38 modifikovat i pro projektivní reprezentaci.
39 Podrobné informace viz:
40     • Skala,V.: A Fast Algorithm for Line Clipping by Convex Polyhedron in E3, Computers &
41         Graphics, Vol.21, No.2, pp.209-214, Elsevier (Pergamon), 1997
42

```

1 6.3. Operace s n-úhelníky v E^2

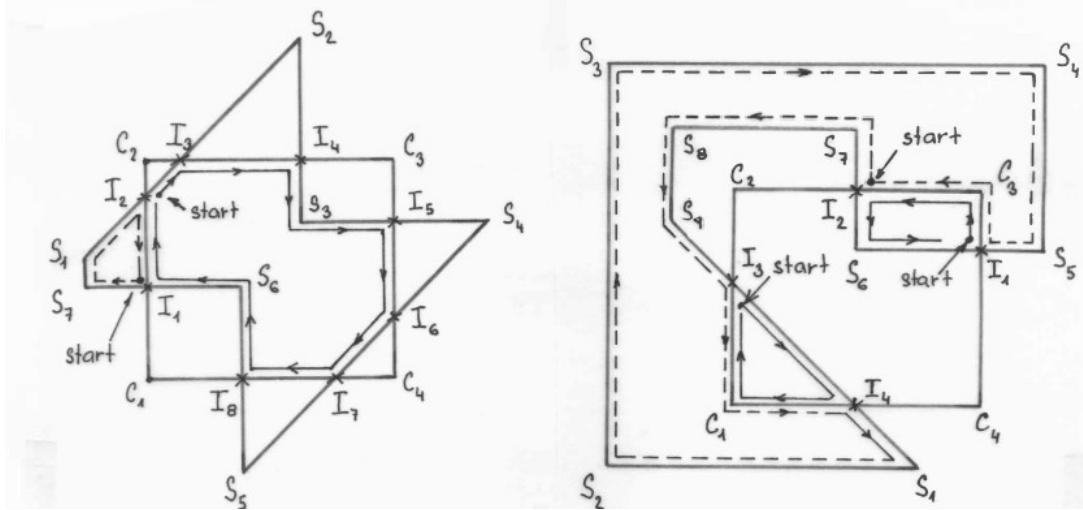
2 Mezi algoritmy ořezávání patří algoritmy pro operaci s n-úhelníky, zejména pak operace množinové
 3 jako je průnik, sjednocení, rozdíl atd. Je zřejmé, že algoritmy se budou zásadně odlišovat podle typů
 4 n-úhelníků, se kterými jsou operace prováděny. Hlavním faktorem pak je, zda n-úhelníky jsou
 5 konvexní nebo nekonvexní. Při sjednocení dvou disjunktních konvexních n-úhelníků zajisté můžeme
 6 obdržet 2 samostatné konvexní n-úhelníky. Avšak i v případě sjednocení dvou konvexních n-úhelníků
 7 můžeme obdržet n-úhelník nekonvexní. V případě nekonvexních n-úhelníků pak obě operace, tj.
 8 sjednocení i průnik, obecně vedou k množině nekonvexních n-úhelníků na výstupu dané operace.
 9 Operace s nekonvexními n-úhelníky jsou složitosti $O(m n)$, kde m , resp. n je počet vrcholů prvního,
 10 resp. druhého n-úhelníka. Pokud jeden n-úhelník je konvexní, lze očekávat složitost $O(m \lg n)$.

11 Weiler-Athertonův algoritmus je jedním ze základních algoritmů pro ořezávání nekonvexního
 12 n-úhelníka nekonvexním n-úhelníkem a vlastně realizuje operaci průniku.

13

14 6.3.1. Weiler-Athertonův Algoritmus

15 Weiler-Athertonův (W-A)algoritmus používá *orientovaný seznam* vrcholů k reprezentaci n-úhelníka.
 16 Algoritmus je založen na práci se seznamy vrcholů reprezentující nekonvexní n-úhelníky.

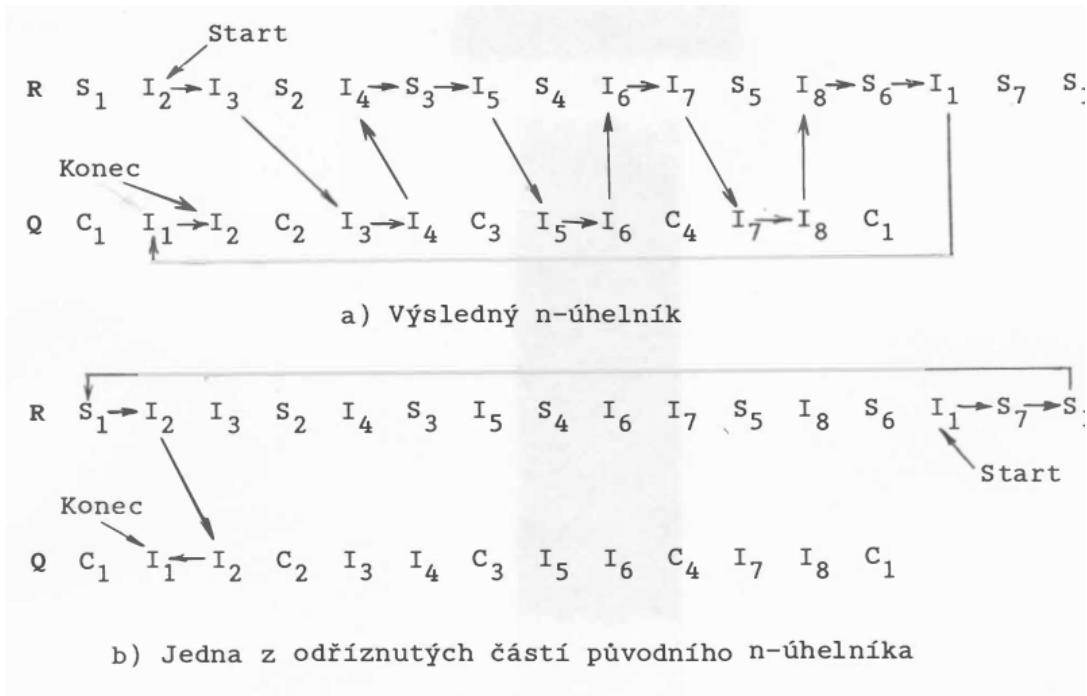


17 Obr.6.17: Reprezentace n-úhelníků a postup při generování výsledného n-úhelníka

18

19 Vzájemná orientace vrcholů v seznamu pak určuje, zda W-A algoritmus bude realizovat průnik, tj.
 20 oříznutí, nebo sjednocení apod.

22



Obr.6.18: Postup procházení seznamem vrcholů a seznam reprezentující výsledný n-úhelník

Podrobnější popis W-A algoritmu lze nalézt v odborné literatuře nebo v:

- Skala,V.: Algoritmy počítačové grafiky II, skripta, Plzeň 2011
(volně dostupné v elektronické formě)

6.3.2. Množinové operace

Množinové operace v rovině nejsou příliš časté, ale jejich použití v prostoru je poměrně časté.
Uveďme alespoň základní typy operací:

Průnik	Sjednocení	Rozdíl	Doplněk
$C := A \cap B$	$C := A \cup B$	$C := A - B$	$C := -A$

kde A, B, C jsou obecně nekonvexní n-úhelníky, resp. množiny nekonvexních n-úhelníků.

Při složitějších operacích, které jsou vlastně definovány stromovou strukturou, je vhodné tuto strukturu transformovat tak, aby operace průniku a rozdílu byly pokud možno co nejníže, operace sjednocení pak naopak co nejvíše. Jde o proces známý pod názvem „optimalizace dotazu uživatele“ z databázových systémů. Množinové operace s geometrickými objekty se běžně používají v geometrickém modelování, viz kap.15.4 (CSG stromy).

Je zřejmé, že pro efektivní algoritmy jsou nutné i vhodné datové struktury. Určitě není nejlepším způsobem reprezentovat např. konvexní n-úhelník, resp. konvexní mnohostěn pomocí průniku polorovin, resp. poloprostorů apod.

Volba datové struktury zásadně ovlivňuje efektivitu algoritmu.

1 7. Datové struktury

2 Datové struktury a jejich použití je klíčovou součástí všech aplikací. Datové struktury musí být pokud
3 možno voleny tak, aby:

- 4 • byly používány v celé aplikaci, aby zbytečně nedocházelo k jejich převodu do jiných datových
5 struktur apod.
- 6 • mohly maximálně využívat *cache* paměti při předpokládaném zpracování. Je vhodné pokud
7 možno eliminovat indexování pomocí dvou nebo tří indexů
- 8 • podporovaly konzistence geometrických modelů, tj. musí umožňovat snadnou kontrolu
9 konzistence dat a geometrických aspektů
- 10 • nevyžadovaly časté prohazování stránek virtuální paměti apod.

11 Je nutné si uvědomit, že při současném rozsahu zpracovávaných dat už není příliš kritická rychlosť
12 elementárních operací, ale kritická je rychlosť přenosu dat z paměti, případně z velkokapacitní paměti
13 do cache paměti daného systému.

14

15 7.1. Základní typy popisu dat

16 Popis geometrických entit můžeme rozdělit v zásadě na:

$$\text{objekt zadaný} \left\{ \begin{array}{l} \text{implicitně} \\ \text{parametricky} \\ \text{explicitně} \\ \text{datovou strukturou} \end{array} \right.$$

17 Jednotlivé typy popisu jsou pro jednoduchost ukázány na rovnicích přímky a roviny.

18

19 Implicitní zadání

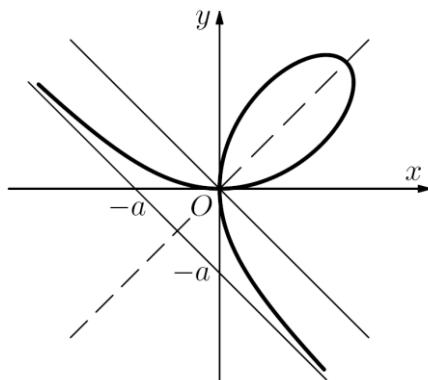
20 Implicitní forma je určena implicitní rovnicí ve tvaru $F(\mathbf{x}) = 0$. Objekt je tedy určen v případě E^2
21 křivkou nebo v případě E^3 povrchem, splňujícím danou rovnici. Je nutné zdůraznit, že obecně není
22 dána orientace a vynásobení funkce libovolnou konstantou $q \neq 0$ určuje stejnou křivku, resp. stejný
23 povrch.

Přímka	Rovina	Kružnice
$Ax + By + C = 0$	$Ax + By + Cz + D = 0$	$x^2 + y^2 - R^2 = 0$

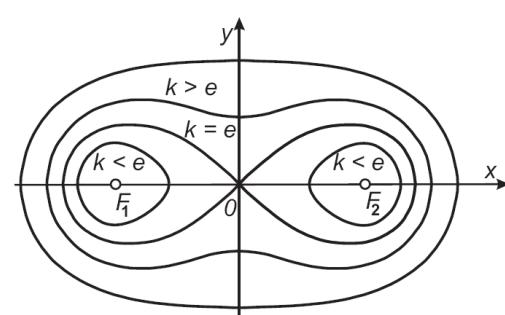
24 Implicitní popis je velmi silný, avšak v mnoha případech vede k řešení nelineární algebraické rovnice
25 nebo jejich soustav.

26

27 Příklad složitějších funkcí



Obr.7.1: Descartův list $x^3 + y^3 - 3axy = 0$



Obr.7.2: Cassiniho křivky

1 **Parametrické zadání**

2 Parametrická forma je často používána, neboť poskytuje orientaci v rovině parametrů. Typickou
3 ukázkou je např. lineární interpolace, nicméně parametricky lze vyjádřit mnohé křivky nebo plochy.
4

Přímka	Rovina	Kružnice
$\mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)t$	$\mathbf{x}(u, v) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)u + (\mathbf{x}_3 - \mathbf{x}_1)v$	$\mathbf{x} = R \begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix}$ nebo $\mathbf{x} = R \begin{bmatrix} 2t \\ \frac{1-t^2}{1+t^2} \end{bmatrix}^T$

5
6 Parametrická forma je poměrně často používána pro geometrické modelování křivek a ploch. Při
7 vykreslování se parametrické křivky nahrazují lomenou čárou, parametricky zadané plochy pak
8 množinou trojúhelníků nebo trojúhelníkovou sítí.
9 Podrobnější informace, viz kap.11 (Parametrické křivky a plochy), kde jsou popsány kubické křivky a
10 bikubické parametrické plochy.

11
12 Parametrickým popisem se dají popisovat i objekty, které mají jen jednostrannou plochu. Jako příklad
13 uvedeme Kleinovu lahev, která je definována takto:

14
15 Pro $0 \leq v \leq 2\pi, r > 0$ a $0 \leq u < \pi$

$$x = 6 \cos u (1 + \sin u) + 4r \left(1 - \frac{\cos u}{2}\right) \cos u \cos v$$

$$y = 16 \sin u + 4r \left(1 - \frac{\cos u}{2}\right) \sin u \cos v$$

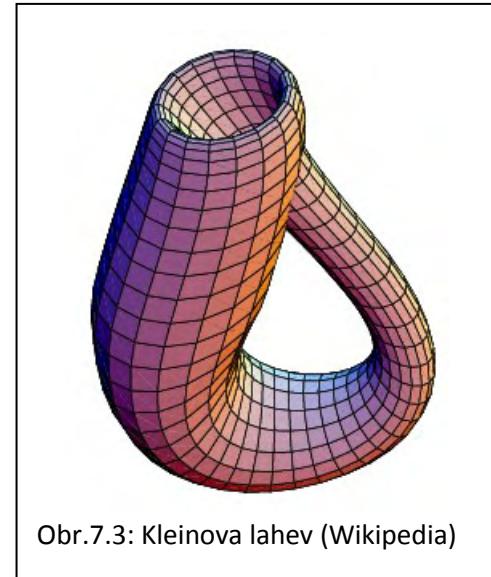
$$z = 4r \left(1 - \frac{\cos u}{2}\right) \sin v$$

16 a pro $0 \leq v \leq 2\pi, r > 0$ a $\pi \leq u < 2\pi$

$$x = 6 \cos u (1 + \sin u) - 4r \left(1 - \frac{\cos u}{2}\right) \cos u \cos v$$

$$y = 16 \sin u$$

$$z = 4r \left(1 - \frac{\cos u}{2}\right) \sin v$$



Obr.7.3: Kleinova lahev (Wikipedia)

17

18

19 **Explicitní zadání**

20 Explicitní forma je asi v praxi nejčastějším případem, neboť pro zadané hodnoty nezávisle
21 proměnné x umožnuje výpočet funkční hodnoty v tomto bodě, tj. $y = f(x)$.

22 Uvážíme-li rovnici přímky v implicitní formě, pak její explicitní ekvivalenty jsou:

$Ax + By + C = 0$	$y = kx + q$ $k \neq \infty$	$x = my + p$ $m \neq \infty$
-------------------	---------------------------------	---------------------------------

23 Je tedy zřejmé, že pro přímku se sklonem do $\pm 45^\circ$ je vhodnější pro numerické výpočty rovnice
24 $y = kx + q$, v opačném případě pak je vhodné použít rovnice $x = my + p$. Např. v případě kružnice,
25 tj. $x^2 + y^2 - R^2 = 0$, dostáváme $y_{1,2} = \pm \sqrt{R^2 - x^2}$ nejen dvě hodnoty, ale pro $x \rightarrow R$ bude
26 hodnota y zřejmě vypočetna s velkou numerickou chybou.

7.2.Základní geometrická primitiva

V současných grafických systémech jsou k dispozici základní geometrické elementy, pro které je optimalizován grafický hardware.

Základní primitiva

- body – zadávají se obecně v E^3 , resp. v P^3 v homogenních souřadnicích, tj. $\mathbf{x} = [x, y, z: w]^T$
- úsečka (line segment) – je dána koncovými body
- trojúhelník (triangle) – je určen 3 body. Orientace vrcholů trojúhelníka v E^3 obecně není možná bez referenčního bodu, např. pozice pozorovatele, nebo nějakou úmluvou, např. že normála trojúhelníka směřuje z objektu ven atd.

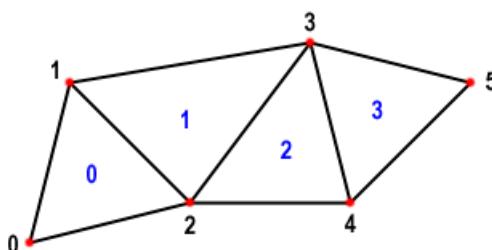
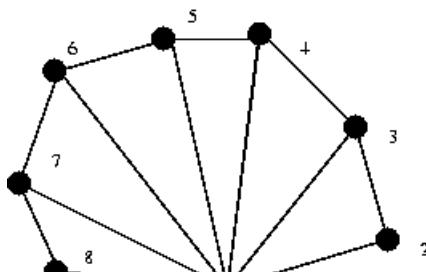


Fig. 1: Triangle strip

Obr.7.4: Pás trojúhelníků (strip)



Obr.7.5: Vějíř trojúhelníků (fan)

Složená primitiva, která se používají, pro urychlení vykreslování:

- pás trojúhelníků (triangle strip) - pro vykreslení se zadává jen posloupnost bodů $P_0, P_1, P_2, \dots, P_5$. V případě vykreslování jako množiny trojúhelníků bychom body P_1, P_2, \dots, P_4 transformovali vícekrát a byly by zpracovávány pro každý trojúhelník samostatně.
- vějíř trojúhelníků (triangle fan) – nepoužívá se příliš často, ale opět šetří čas zpracování. Pro vykreslení se zadává jen posloupnost bodů P_1, P_2, \dots, P_8 , místo zadávání 6 samostatných trojúhelníků.

V některých systémech se ještě používají čtyřúhelníky, ale zde je nutné zdůraznit, že i když jsou rovinné z hlediska čistě matematického, nejsou rovinné z důvodů numerické přesnosti.

Obecně pak vykreslování grafických primitiv je dáno sekvencí:

- nastav atributy – nastavuje se např. barva apod.
- vykresli dané primitivum, resp. primitiva

V mnoha postupech, např. při stínování, je nutné určit normálu trojúhelníka. Je zřejmé, že normálu lze určit např. jako $\mathbf{n} = (P_1 - P_0) \times (P_2 - P_0)$. Je opět nutné zdůraznit, že jde vlastně o bivektor, tj. vektor orientované plochy, takže pro geometrické transformace je nutné použít postup uvedený v kap.4.8 (Transformace přímek a rovin).

Až dosud bylo víceméně předpokládáno, že se pracuje s povrchy objektů, které jsou reprezentovány množinou trojúhelníků, které jsou dále zpracovávány do vizuální podoby. Jde tedy o zpracování plošných elementů, které jsou v obecné poloze v prostoru E^3 .

7.3. Reprezentace třírozměrných objektů a scén

V případě plošných elementů bylo možné reprezentovat n-úhelníky buď *uspořádaným seznamem* po sobě jdoucích vrcholů, neboť v E^2 je relevantní pojem orientace, nebo seznamem hran s odkazem do tabulky se souřadnicemi příslušných vrcholů. V případě E^3 je však nutné reprezentovat uzavřený povrch, resp. objem geometrického objektu.

Reprezentace geometrických objektů lze rozdělit na:

- *reprezentace hraniční B-rep* (Boundary representation)
 - *hranové* reprezentace slouží převážně k reprezentaci „drátěných“ modelů. Není však možné řešit viditelnost ploch, neboť tato informace není v datové struktuře k dispozici
 - *plošné* reprezentace, např. okřídlená hrana, half-edge atd., reprezentují povrch tělesa tak, že obsahují informaci o plochách, vrcholech a hranách daného objektu. V případě parametrických ploch, datová struktura obsahuje informace o vrcholech a případně hraničních křivkách atd.
- *reprezentace objemové*
 - *diskrétní* – objekt je reprezentován v diskrétní 2D nebo 3D mřížce a uzel mřížky je asociovaný s hodnotou skalární nebo vektorovou. Typickou ukázkou skalárních dat jsou např. CT a MRI data
 - *CSG stromy* (Constructive Solid Geometry) apod., je objekt reprezentován např. pomocí „orientované“ implicitní funkce $F(x)$, kdy se např. předpokládá, že $F(x) < 0$ pro všechny body uvnitř tělesa, $F(x) > 0$ pro body vně tělesa a $F(x) = 0$ reprezentuje povrch objektu. Složitější objekty se pak vytvářejí pomocí množinových operací, např. sjednocení, průnik atd., ze základních objektů

Vedle uvedených datových struktur jsou používány speciální datové struktury, zejména pro účely urychlování, pro reprezentaci rozmístění objektů v rovině nebo prostoru.

Obecně lze datové struktury pro reprezentaci objektů a jejich pozic v prostoru rozdělit takto:

- *hierarchické* datové struktury sloužící především k účelům „zjemňování“, resp. zmenšování prostoru, který je nutno při výpočtu uvažovat:
 - *statické* – sloužící především k uložení hierarchie, resp. vazeb mezi objekty, jejich hloubka je předem dáná
 - *adaptivní* – hloubka struktury se může měnit podle složitosti scény v té které části prostoru
- *ne-hierarchické* datové struktury – sem např. patří množina trojúhelníků, n-úhelníků, mnohostěnů atd. bez struktur popisující jejich vzájemný vztah

V následujícím budou uvedeny jen základní datové struktury. Pokročilé datové struktury a jejich aplikace lze nalézt např. v:

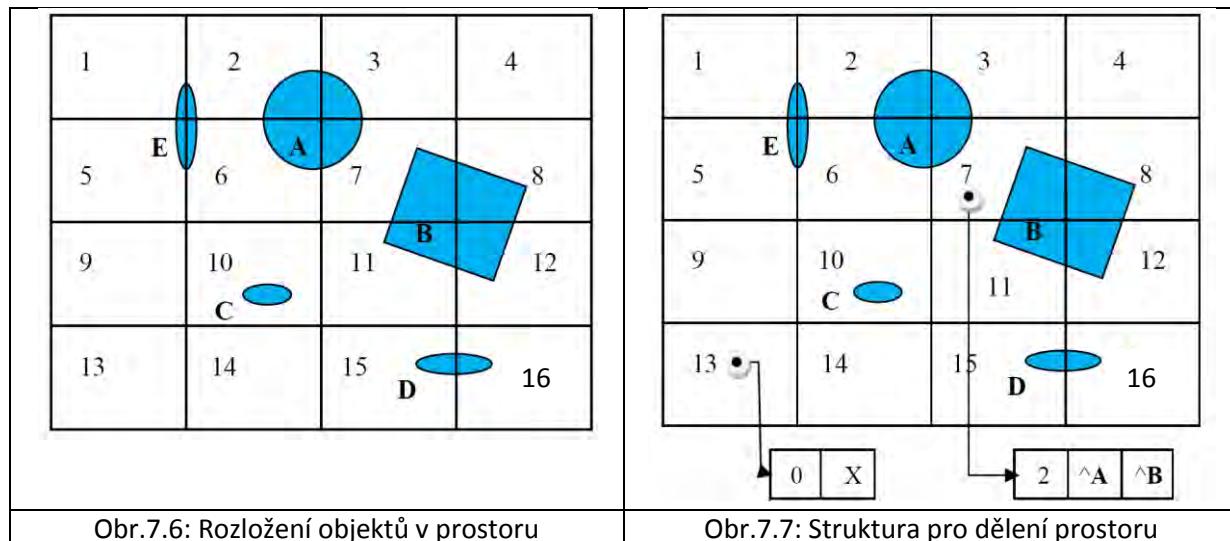
- Langetepe,E., Zachmann,G.: Geometric Data Structures for Computer Graphics, A.K.Peters, 2006

7.4. Dělení prostoru a Binární masky

V mnoha aplikacích je nutné zjistit, zda v nějaké části prostoru je nějaký objekt včetně jeho identifikace. Jde tedy o problém lokalizace objektu v určité části prostoru. Jednou z možností efektivního řešení této úlohy je dělení prostoru na menší prostory a pro ně pak uchovávat identifikaci objektů, které s daným prostorem incidují, tj. mají neprázdný průnik. Tento přístup v zásadě využívají techniky typu „rozděl a panuj“ (Divide and Conquer).

7.4.1. Dělení prostoru

Standardní dělení prostoru (Space Subdivision) je poměrně oblíbená jednoduchá technika, kdy daný prostor rozdělíme, většinou rovnoměrně, na menší prostory. Pro každý prostor pak uchováváme, např. v seznamu, identifikátory objektů, které s daným prostorem sdílí nějakou část.



Paměťová náročnost techniky dělení prostoru může být odhadnuta na:

$$O(p q M^d) \quad (7.1)$$

kde: M je počet dělení na jedné ose, d je dimenze prostoru, p je počet objektů, $q = q(M, p, s)$ je pravděpodobnost, že „průměrný“ objekt zasáhne sub-prostor. Jde tedy o funkci závisející na jemnosti dělení a velikosti „průměrného“ objektu, podrobně viz

- Skala,V.: Memory Saving Technique for Space Subdivision Technique, Machine Graphics and Vision, Vol.2, No.3, pp.237-250, ISSN 1230-0535, 1993 ([CLICK off-line](#))

Paměťové nároky pro standardní dělení prostoru (Space Subdivision) pak jsou:

$$Mss = M^d (1 + pq + 1) * 4 = M^d (pq + 2) * 4 \quad [B] \quad (7.2)$$

pokud se alokují se 4 B na ukazatel (pointer).

Je tedy zřejmé, že paměťové nároky prudce vzrůstají s jemností dělení prostoru, tj. hodnotou M a dimenzí prostoru d .

7.4.2. Rezidenční masky

Jiný přístup jsou rezidenční masky (Residency Masks), viz Cychosz, které používají bitové masky. Pro každý objekt je k dispozici vektor bitů, který říká, zda daný objekt se v dané oblasti nachází, tj. počet bitů je dán celkovým dělením prostoru. Matice Q ukazuje stav pro situaci na Obr.7.6.

$$\mathbf{Q} = \begin{bmatrix} 0000 & 0000 & 0110 & 0110 \\ 0000 & 1100 & 1100 & 0000 \\ 0000 & 0010 & 0000 & 0000 \\ 1100 & 0000 & 0000 & 0000 \\ 0000 & 0000 & 0011 & 0011 \\ 15 & bits & \dots & 0 \end{bmatrix} = \begin{bmatrix} \text{Object A} \\ \text{Object B} \\ \text{Object C} \\ \text{Object D} \\ \text{Object E} \end{bmatrix} \quad (7.3)$$

1 Lze ukázat, že paměťové nároky jsou určeny:

$$M_{RM} = pM^d/8 [B] \quad (7.4)$$

2 kde: p je počet objektů v prostoru.

3 Metoda rezidenčních masek umožňuje rychlou detekci kolize objektů pomocí bitové operace **land**,
4 takže při zkoumání možné detekce objektů **C** a **D** řešíme jednoduchou podmínu:

$$\mathbf{Q}[3,*] \text{ land } \mathbf{Q}[4,*] \neq [0, \dots, 0]$$

5

6 7.4.3. Binární masky

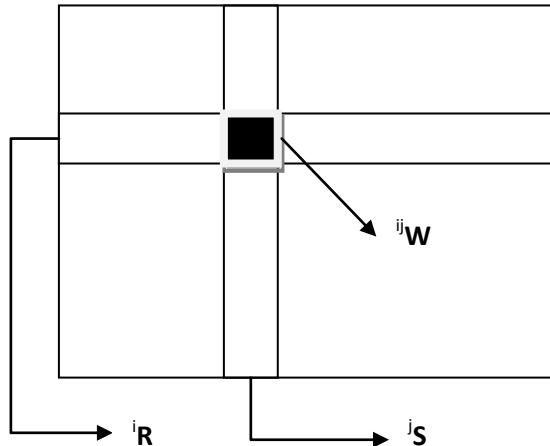
7 Binární masky (Binary Masks) jsou technikou, založenou na diametrálně jiném přístupu. Metoda
8 rezidenčních masek velmi dobře zodpoví otázku:

9 **Q1:** Nalezní oblasti, které incidují s daným objektem.

10 Nicméně v převážné většině případů potřebujeme odpověď trochu jinou otázku, a to:

11 **Q2:** Nalezní všechny objekty, které incidují s danou oblastí.

12 Pokud dotazy analyzujeme, zjistíme, že jsou to otázky komplementární, viz situace:



13

Obr.7.8: Princip binárních masek

14 Uvažme množinu iR , resp. jS objektů, které incidují s řádkovým „řezem“ i , resp. se sloupcovým
15 „řezem“ j . Pak množina ijW , která je definována jako:

$$ijW = iR \cap jS \quad (7.5)$$

16 určuje objekty, které **mohou** incidovat s oblastí na pozici ij . Obdobně pro E^3 dostáváme:

$$ijkW = iR \cap jS \cap kT \quad (7.6)$$

17 kde kT je množina objektů, které incidují s „hloubkovým řezem“ k .

18 Podrobnější analýzou snadno zjistíme, že se podařilo převést paměťovou složitost z $O(M^d)$ na
19 $O(dM)$. V případě E^3 pro $M = 256$ paměťové nároky snížíme v poměru $256^3 \rightarrow 3 \cdot 256$!

1 Z hlediska implementačního je vhodné množiny iR , jS , kT implementovat jako bitový vektor o
 2 délce p bitů, kde p je počet objektů v prostoru. Bit $i_{r_k} = 1$, pokud k -tý objekt inciduje s i -tou
 3 řádkou, jinak $i_{r_k} = 0$, analogicky i pro ostatní řady.

4 Paměťová náročnost metody bitových masek M_{BM} je:

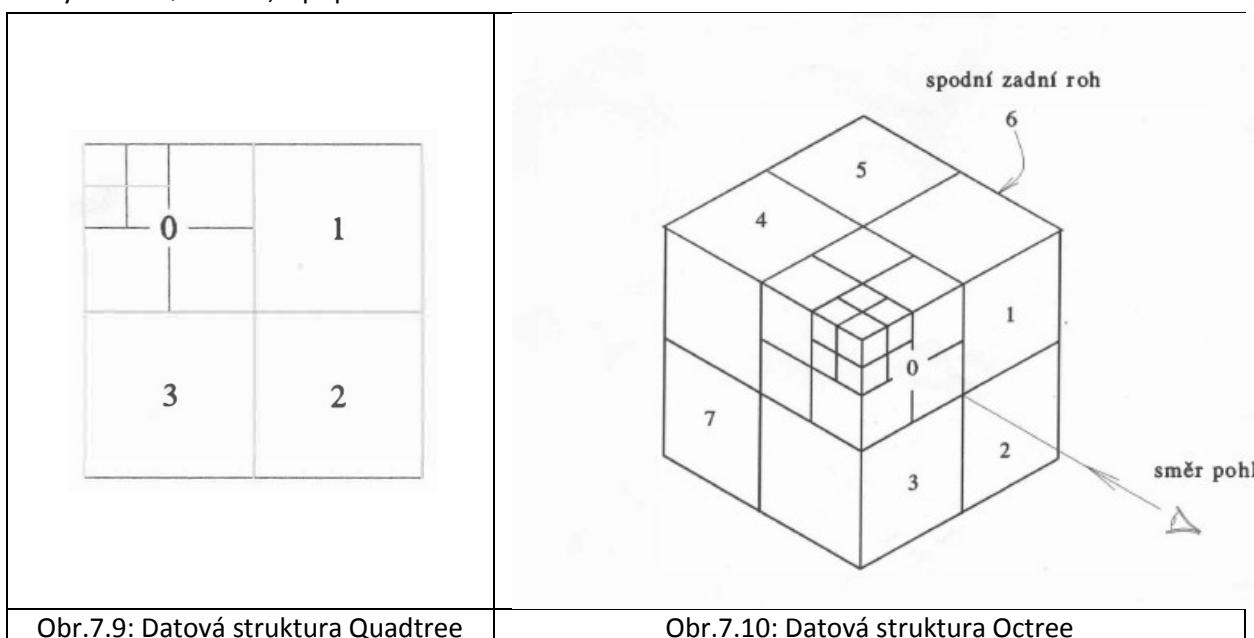
$$M_{BM} = \frac{dpM}{8} \quad [\text{Byte}] \quad (7.7)$$

5 Podrobněji viz:

- 6 • Cychosz,J.M., Use of Residency Mask and Object Space Partitioning to Eliminate Ray Object
 7 Intersection Calculation, Graphics Gems III (Ed.Kirk,D.), pp.284-287, 1992.
- 8 • Skala,V.: An Efficient Space Partitioning Method Using Binary Maps, 11th Conference SIP'2012
 9 conference , pp.121-124, ISBN: 978-1-61804-081-7, St.Malo, WSEAS, France, 2012
 10 ([CLICK off-line](#))
- 11 • Skala,V.: Memory Saving Technique for Space Subdivision Technique, Machine Graphics and
 12 Vision, Vol.2, No.3, pp.237-250, ISSN 1230-0535, 1993
 13 ([CLICK off-line](#))

15 7.4.4. Quadtree a Octree

16 Quadtree a Octree jsou techniky hierarchického dělení prostoru. V zásadě jde o stejný princip a
 17 oblast daného prostoru obsahující objekty se postupně dělí na menší a menší sub-oblasti, pokud
 18 s danou oblastí incideuje větší než specifikovaný počet objektů. V případě E^2 dostáváme techniku
 19 nazývanou *Quadtree*, v případě E^3 dostáváme *Octree*.



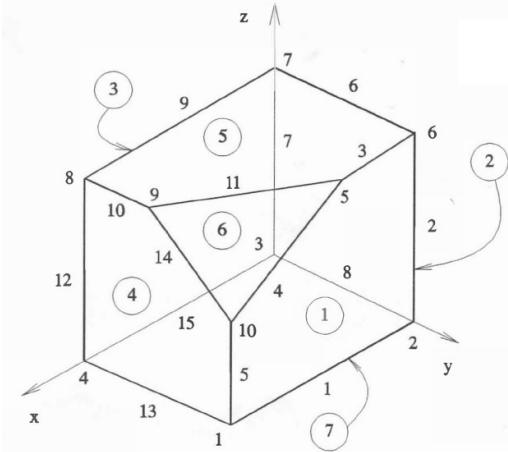
20 Hlavní nevýhodou datových struktur Quadtree a Octree je to, že nejsou invariantní např. vůči rotaci.

21 Takže velmi často dochází k nutnosti jejich opětovného vytvoření, což je výpočetně velmi náročné.

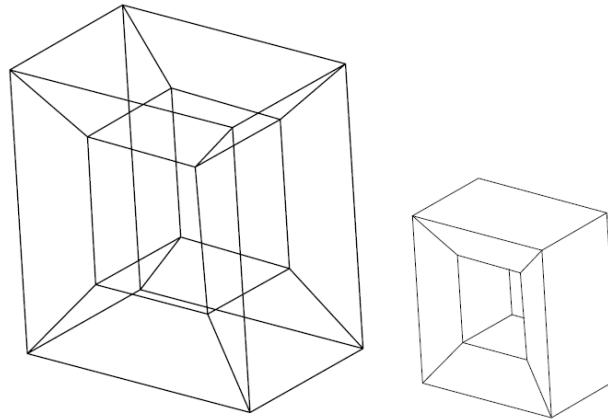
22 V mnoha aplikacích, zejména ve strojírenský orientovaných CAD systémech, se vyžaduje hraniční
 23 reprezentace k reprezentaci povrchu a manipulaci s ním. Vedle toho je však v mnoha aplikacích
 24 potřeba počítat s mechanickými veličinami jako je objem, váha, moment setrvačnosti atd. K výpočtu
 25 těchto veličin však povrchová reprezentace není vhodná.

7.5. Hraniční reprezentace - dodělat Half-Edge

Hraniční reprezentace jsou charakterizovány popisem hranic daného objektu a jsou označovány termínem *B-rep*, zkratkou od anglického *Boundary representation*. Pro jejich vysvětlení uvažme geometrický objekt (seseknutý kvádr) na Obr.7.11.



Obr.7.11: Objekt reprezentovaný hraniční reprezentací



Obr.7.12: Drátěný model a jedna z možných reprezentací

Nejjednodušší reprezentací je reprezentace hranová.

8. Hranová reprezentace

Hranová reprezentace je nejjednodušší datovou strukturou, neboť je dána pouze tabulkou vrcholů V s jejich souřadnicemi a tabulkou hran H , která pro každou hranu určuje její koncové body.

Tabulka vrcholů V je dána:

vrchol	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	
v =	[4	0	0	4	1	0	0	4	4	4
		3	3	0	0	3	3	0	0	2	3
		0	0	0	0	4	4	4	4	2	

Tabulka hran H je dána:

hrana	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	
H =	[1	2	6	5	10	6	7	3	7	8	9	8	4	9	3
		2	6	5	10	1	7	3	2	8	9	5	4	1	10	4

Hranová reprezentace není vhodná, pokud budeme chtít určovat viditelnost hran, neboť schází informace o plochách. Proto se také někdy označuje termínem *drátěný model*, který je nejednoznačný z hlediska interpretace ploch. na Obr.7.12 je uvedena jedna ze tří možných interpretací.

19

1 Plošná reprezentace

K řešení viditelnosti, tj. eliminaci neviditelných ploch a jejich částí, které jsou zakryty jinými plochami daného či jiného objektu, je zapotřebí informace o plochách. Je tedy nutné hranový model rozšířit o hraniční plochy, které jsou definovány tabulkou ploch P .

číslo plochy	počet hran	číslo hrany ohraničující plochu
1	5	1 , 2 , 3 , 4 , 5
2	4	2 , 6 , 7 , 8
3	4	7 , 9 , 12 , 15
4	5	12 , 13 , 5 , 14 , 10
5	5	6 , 9 , 10 , 11 , 3
6	3	4 , 11 , 14
7	4	1 , 8 , 15 , 13

Tabulka ploch P .

7 Tabulky V , H a P se někdy ještě doplňují o informaci, které dva n-úhelníky sdílejí danou hranu, viz
8 tabulka H' .

hrana	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
$H' =$	1	2	6	5	10	6	7	3	7	8	9	8	4	9	3
	2	6	5	10	1	7	3	2	8	9	5	4	1	10	4
	1	1	1	1	1	2	2	2	3	4	5	3	4	4	3
	7	2	5	6	4	5	3	7	5	5	6	4	7	6	7

TAB Modifikovaná tabulka hran H

První dva řádky tabulky obsahují indexy vrcholů hran a druhé dva řádky obsahují indexy ploch, které danou hranu sdílí. Pokud lze dodržet uspořádání vrcholů jednotlivých ploch konzistentně, pak je možné tabulku hran H vynechat a modifikovaná tabulka P' má pak tvar:

číslo plochy	počet vrcholů	čísla hran plochy
1	5	1 , 2 , 3 , 4 , 5
2	4	2 , 8 , 7 , 6
3	4	15 , 12 , 9 , 7
4	5	13 , 5 , 14 , 10 , 12
5	5	11 , 3 , 6 , 9 , 10
6	3	4 , 11 , 14
7	4	1 , 13 , 15 , 8

TAB Modifikovaná tabuľka ploch P'

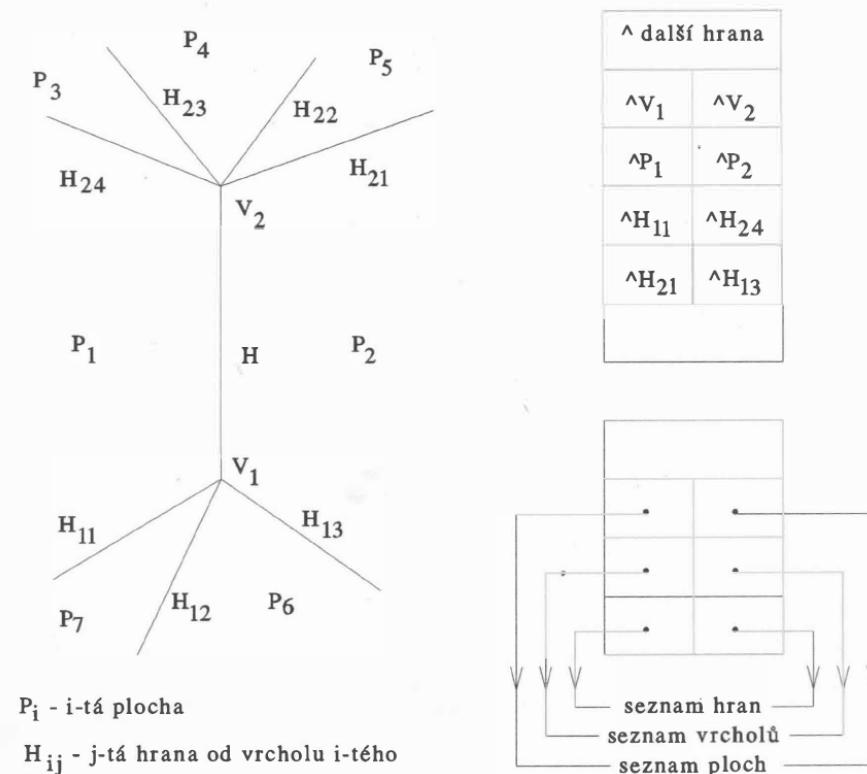
16 Pro vnitřní reprezentaci je však vhodné z tabulky P' odvodit tabulky P a H s tím, že u odkazu na
17 hranu v tabulce P je uložena i informace o orientaci hrany.

18

19 Pro rozsáhlé aplikace byla vyvinuta speciální datová struktura známá pod označením okřídlená hrana,
20 která je založena na seznamech.

1 **Okřídlená hrana**

2 Okřídlená hrana (Winged Edge) je datová struktura vyvinuta původně pro reprezentaci povrchu
 3 objektu, který je tvořen rovinnými n-úhelníky, a pro efektivní zpracování objektů definovaných
 4 rovinnými plochami.



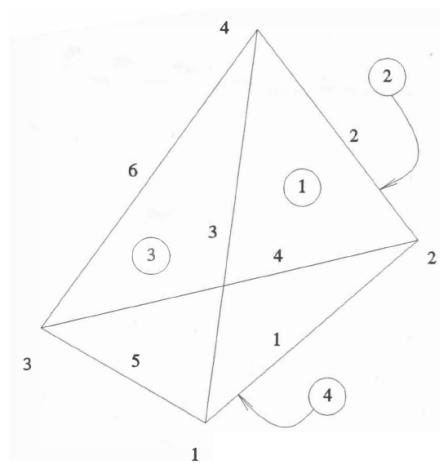
Obr.7.13: Okřídlená hrana

5

6

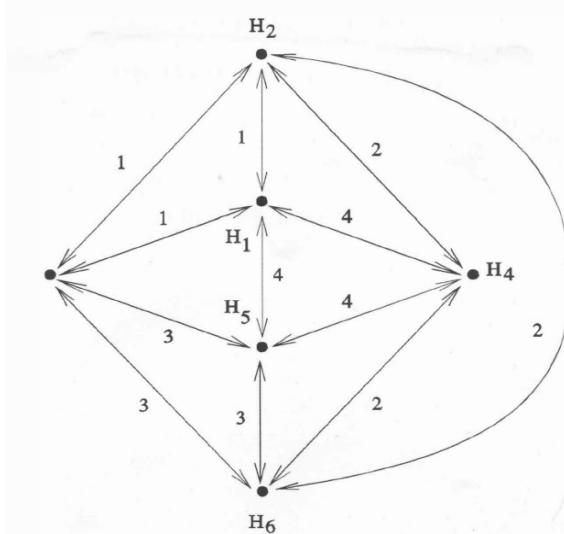
7

8 Datová struktura je založena na seznamové architektuře. Uveďme pro ilustraci, jak by datová
 9 struktura vypadala pro čtyřstěn.



Obr.7.14: Čtyřstěn a jeho plochy, hrany a vrcholy

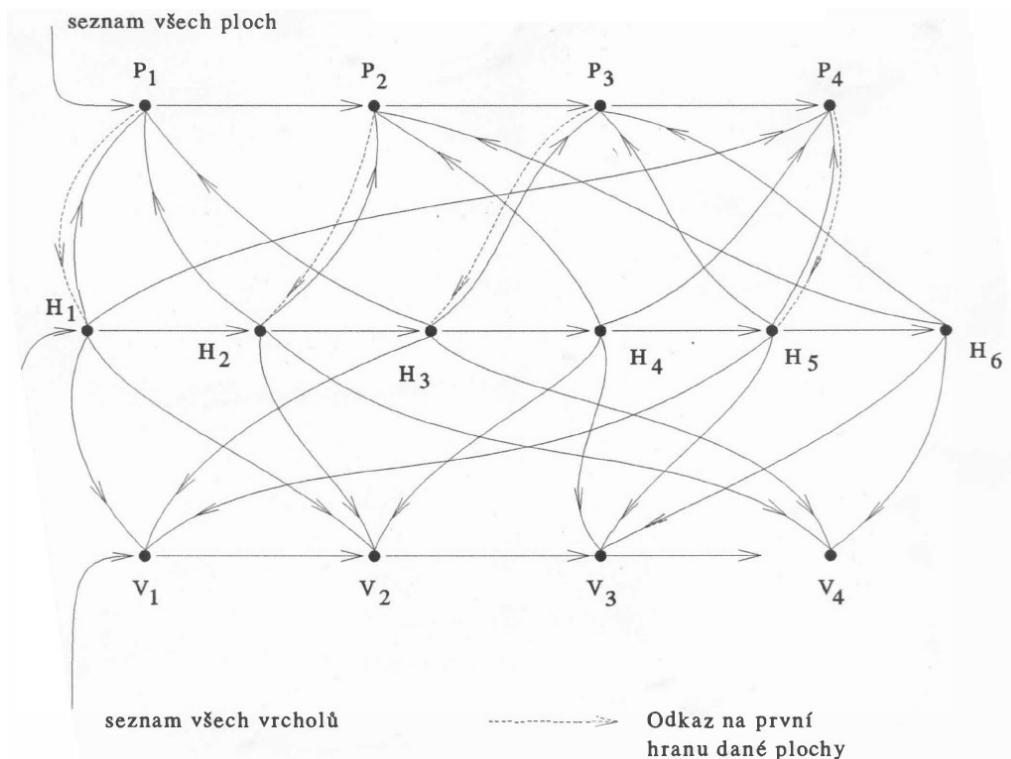
10



Obr.7.15: Grafové pojednání struktury tělesa

hrana H_1 je společná plochám 1 a 4
 a inciduje s hranami H_2, H_3, H_4, H_5

1 Datová struktura okřídlená hrana pak pro daný čtyřstěn má tvar:



2
3 Obr.7.16: Datová struktura reprezentující daný objekt

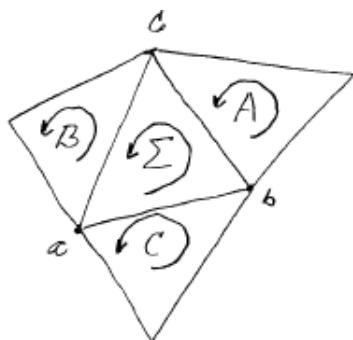
4 Vzhledem k omezené přesnosti v reprezentaci souřadnic vrcholů je rozumné n-úhelníky
5 reprezentovat pomocí trojúhelníkové sítě. Pak se okřídlená hrana podstatně zjednoduší a je
6 reprezentovatelná jednoduchými datovými strukturami.

7 Modifikace pro trojúhelníkové sítě

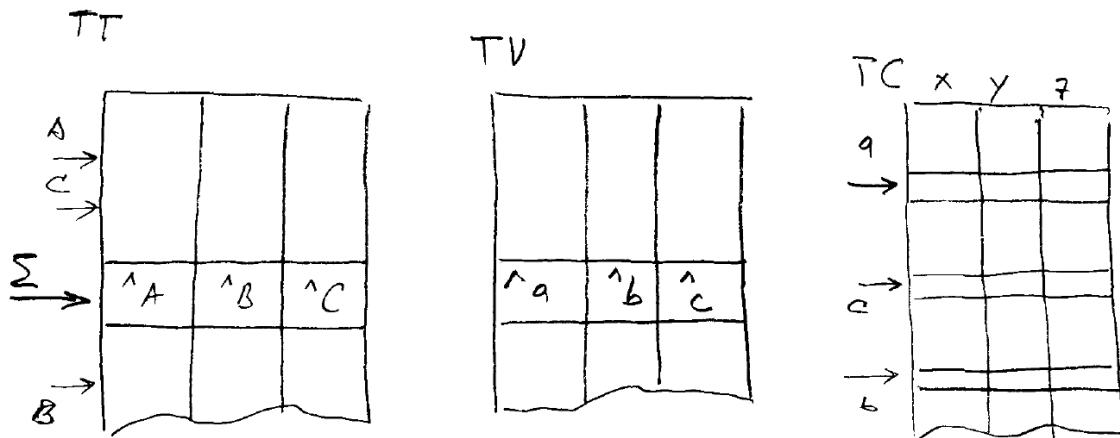
9 Datovou strukturu „okřídlená hrana“ je možné modifikovat pro trojúhelníkové sítě poměrně
10 jednoduše. Uvažme část trojúhelníkové sítě včetně jejich orientace, Obr.7.17. Pravidla indexace jsou
11 jednoduchá, a to:

- 12 • proti vrcholu s indexem a je trojúhelník s indexem A
- 13 • pořadí v tabulce TT a TV si odpovídají
- 14 • orientace všech trojúhelníků je konzistentní pro celý povrch, např. tak, že normálový vektor
15 směruje z objektu

16 Pak modifikovaná datová struktura je:



17
18 Obr.7.17: Trojúhelník a jeho sousedi s pevnou orientací



Obr.7.18: Tabulky pro reprezentaci trojúhelníkové sítě

kde **TT** je tabulka trojúhelníků, **TV** je tabulka vrcholů, **TC** je tabulka souřadnic.

V případě potřeby lze datovou strukturu doplnit o tabulku **TN**, kde jsou uloženy normálové vektory ve vrcholech, resp. normálový vektor ploch

Poznamenejme, že se používají indexy do tabulky, nikoliv ukazatelé („pointry“), a tabulky by měly být implementovány jako jednorozměrné pole struktury, např. **struct(x,y,z: real)**, aby nebylo nutné používat dvouozměrné indexování.

Pokud se v datové struktuře provádějí změny typu rušení (delete) a vkládání (insert) je vhodné implementovat „memory management“ pro vlastní přidělování volných pozic v tabulkách.

Modifikace pro tetrahedronové sítě

Modifikace výše uvedené datové struktury pro tetrahedronové sítě je velmi jednoduchá, neboť v zásadě stačí přidat jeden sloupec k tabulkám **TT** a **TV**.

18

1 **Datová struktura „Half-edge“**

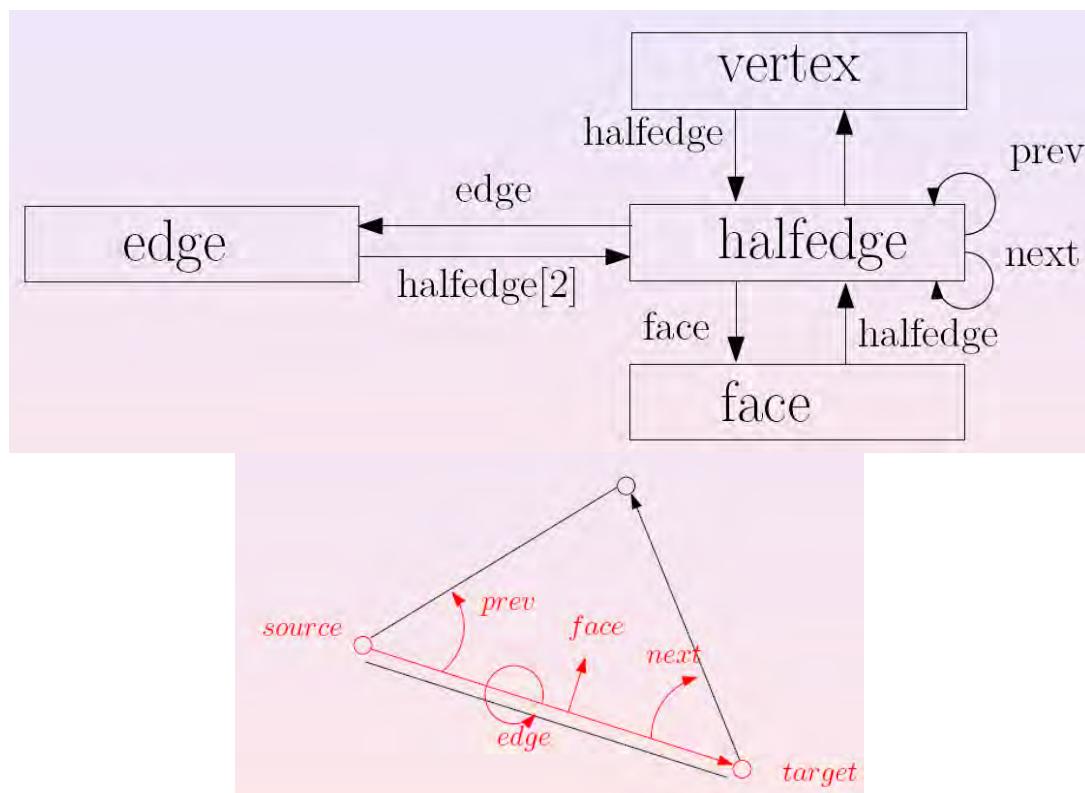
2 Dalším datovým typem je datová struktura „half-edge“. Její přesnou definici lze nalézt např. v
 3 http://www.cgal.org/Manual/latest/doc_html/cgal_manual/HalfedgeDS/Chapter_main.html
 4 Podstatnou výhodou je především snadnost realizace Eulerových operátorů a spotřeba paměti je
 5 minimalizována.

6

7 Datová struktura má následující části:

- 8 • vrcholy
- 9 • half-edge, tj. orientovaná hrana
- 10 • hrana, tj. neorientovaná
- 11 • plocha orientovaná

12



Obr.7.19: „Half-Edge struktura“

13

14

15

16

17

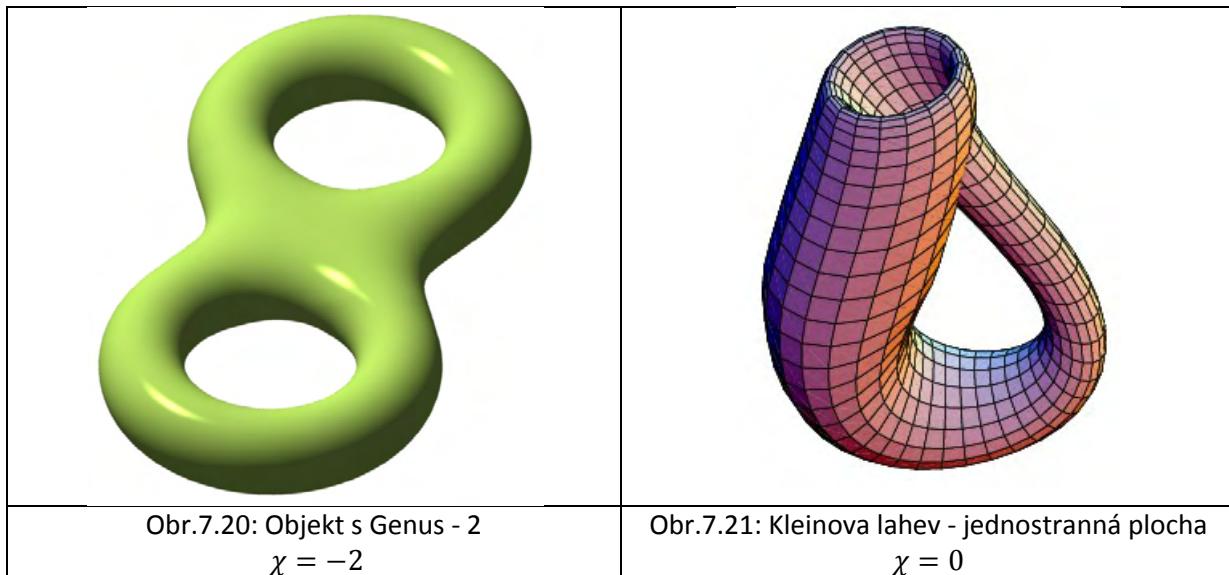
1 7.6. Eulerovy operátory a manifoldy

2 Dříve popsané objekty, které jsou popsány, jako mnohostěny, splňují jednu vlastnost, kterou jsme
 3 implicitně předpokládali, že hrana je sdílena sudým počtem ploch. Taková vlastnost se označuje
 4 pojmem *manifold*. Ve většině případů jde o *2-manifold*, tj. hrany sdílí právě 2 plochy. Pro mnohostěny
 5 bez děr pak platí Eulerův vztah, někdy též Eulerova rovnice:

$$F + V - E = \chi - 2$$

6 kde F je počet ploch (Face), V je počet vrcholů (Vertex), E je počet hran (Edge). Je nutné upozornit,
 7 že jde o implikaci, ne o ekvivalenci, tj. pokud platí uvedený vztah, pak o objektu nemůžeme tvrdit, že
 8 je to mnohostěn bez děr.

9 Eulerova charakteristika χ může nabývat hodnoty $\chi \neq 2$ pro jiná tělesa, např. pro:



10 Pokud objekt má díry, pak platí tzv. zobecněná Euler-Poincaré formule:

$$F + V - E = H + 2(S - G)$$

11 kde H je počet děr (Hole), S je počet spojených komponent (Shell) a G je genus objektu (Genus).
 12 Vedle Eulerovy formule jsou nad manifoldy dále definovány Eulerovy operátory, které definují, jak se
 13 změní Eulerovy vztahy při manipulaci, např. odebrání vrcholu apod., a slouží ke kontrole
 14 konzistentnosti geometrického modelu. Jdou zavedeny dva typy, a to:

Name	Description	ΔV	ΔE	ΔF	ΔH	ΔS	ΔG
MBFLV	Make Body-Face-Loop-Vertex	1	0	1	0	1	0
MEV	Make Edge-Vertex	0	1	1	0	0	0
MEFL	Make Edge-Face-Loop	1	1	0	0	0	0
MEKL	Make Edge, Kill Loop	0	1	0	-1	0	0
KFLEV	Kill Faces-Loops-Edges-Vertices-Body	-2	-n	-n	0	-1	0
KFLEVGM	Kill Faces-Loops-Edges-Vertices, Make Genus	-2	-n	-n	0	0	1

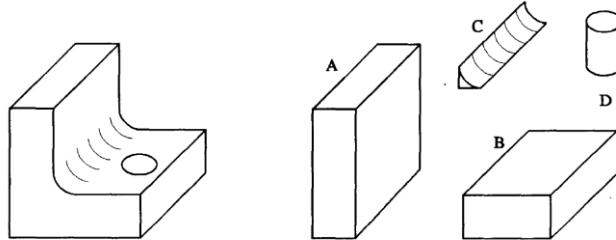
15 První písmeno operace M/K označuje akci Make/Kill, tj. vytvoř/zruš.

16 Podrobné info:

- Havemann,S.: Generative Mesh Modeling, PhD Thesis, 2005, ([CLICK off-Line](#))
http://www.generative-modeling.org/GenerativeModeling/Documents/Dissertation-Havemann_print.pdf

7.7.CSG stromy

2 Reprezentace objektů CSG (Constructive Solid Geometry) je poměrně jednoduchá a elegantní.
3 Předpokládejme, že jsou dány základní objekty jako kvádr, koule, válec, jehlan, kužel atd. a z těchto
4 objektů chceme realizovat složitější objekty pomocí množinových operací, tj. průniku sjednocení atd.



Obr.7.22: Tvořené těleso a použitá základní primitiva

8 Výsledné těleso T je tedy vlastně definováno:

$$T = (A \cup B \cup C) - D$$

9 nebo jako:

$$T = (B - D) \cup A \cup C$$

10 Je tedy zřejmé, že těleso:

- je vlastně reprezentováno stromem, který se nazývá CSG strom, kde listy stromu jsou základní objekty a uzly stromu reprezentují např. množinové operace
 - může mít několik různých reprezentací, tj. CSG stromů, které ve výsledku definují stejné těleso

15 Předpokládejme, že povrch tělesa je reprezentován rovnicí:

$$F(x) = 0$$

16 s tím, že $F(x) < 0$ pro všechny body uvnitř tělesa, $F(x) > 0$ pro body vně tělesa. Tedy funkce $F(x)$
17 definuje *orientovanou plochu*. Proto se také někdy tento postup nazývá funkcionální nebo
18 modelování implicitními funkcemi apod.

20 Předpokládejme, že těleso A je reprezentováno funkcí $F_1(x)$ a těleso B je reprezentováno funkcí
 21 $F_2(x)$. Pak základní operace nad tělesy lze definovat takto:

Název	Matematický symbol	Operátor C++	Matematický popis
Sjednocení	$A \cup B$	<code> </code>	$F(\mathbf{x}) \stackrel{\text{def}}{=} \min\{F_1(\mathbf{x}), F_2(\mathbf{x})\}$
Průnik	$A \cap B$	<code>&</code>	$F(\mathbf{x}) \stackrel{\text{def}}{=} \max\{F_1(\mathbf{x}), F_2(\mathbf{x})\}$
Rozdíl	$A - B$	<code>\</code>	$F(\mathbf{x}) \stackrel{\text{def}}{=} F_1(\mathbf{x}) - F_2(\mathbf{x})$
Doplněk	- A (unární)	-	$F(\mathbf{x}) \stackrel{\text{def}}{=} -F_1(\mathbf{x})$

Tab.XXX: Tabulka množinových operací

24 Je nutné si uvědomit, že tento způsob konstrukce objektů je poměrně velmi mocný. Uvažme objekt
25 vytvořený sjednocením dvou koulí. Pak je zřejmé, že:

- 26 • koule je reprezentována středem x_s a poloměrem R , tj. 4 hodnotami v pohyblivé řádové
 27 čárce. Pokud bychom reprezentovali povrch koule trojúhelníkovou sítí (pro jednoduchost
 28 předpokládejme generaci trojúhelníků pomocí sférických souřadnic), pak budeme potřebovat

cca 20 000 trojúhelníků (dělení v prostoru φ a ϑ na 100×100 bodů a dělení vzniklého čtyřúhelníka na 2 trojúhelníky). Tedy kompresní poměr je $20\,000:4 = 5\,000:1$.

- při realizaci operace sjednocení a i dalších operací vzniká na průsečíku ploch křivka - průsečnice. V případě trojúhelníkové sítě toto vede k následné „explozi“ malých trojúhelníků, neboť je nutné reprezentovat přesně povrch objektu na společné hranici daných objektů.
- v případě funkcionálního popisu a CSG stromů je popis jednoduchý a kompaktní
- funkční popis umožňuje realizaci i prostorově „neomezených“ objektů, jako poloprostory, „nekonečný“ válec, rotační paraboloid atd.

Je nutné upozornit, že gradient výsledné funkce průniku, resp. sjednocení, není spojitý. Tento problém je fundamentální, neboť ani povrch není hladký. Proto se používají trochu složitější funkce, které „vyhlazují“ povrch v místě protínání, a tím pak i gradient funkce je spojitý. Tabulka základních operací, Tab. 7.1, ukazuje jedno možné řešení, kde koeficient α určuje „míru“ vyhlazení.

Na funkcionálním modelování s CSG stromy je založena celá řada systémů, např. systém Hyper-Fun, viz http://cis.k.hosei.ac.jp/~F-rep/HF_descr.html

Název	Operátor	Matematický popis
Sjednocení	$A \cup B$	$F(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{1+\alpha} \left(F_1(\mathbf{x}) + F_2(\mathbf{x}) + \sqrt{ F_1^2(\mathbf{x}) + F_2^2(\mathbf{x}) - 2\alpha F_1(\mathbf{x}) F_2(\mathbf{x}) } \right)$
Průnik	$A \cap B$	$F(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{1+\alpha} \left(F_1(\mathbf{x}) + F_2(\mathbf{x}) - \sqrt{ F_1^2(\mathbf{x}) + F_2^2(\mathbf{x}) - 2\alpha F_1(\mathbf{x}) F_2(\mathbf{x}) } \right)$
Rozdíl	$A - B$	Realizuje se jako $A \cap (-B)$
Doplňek	- A (unární)	$F(\mathbf{x}) \stackrel{\text{def}}{=} -F_1(\mathbf{x})$

Tab. 7.1: Tabulka základních operací

Zásadní výpočetní problém s funkcionální reprezentací je pak extrakce (výpočet) a zobrazení výsledného povrchu. Ve většině případů:

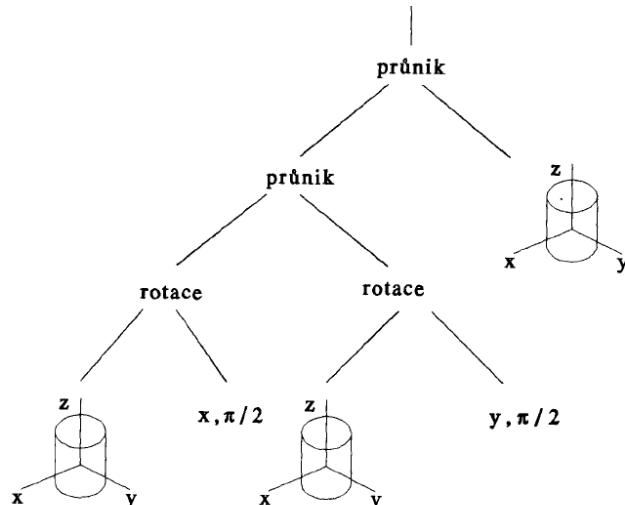
- prostor objektu se diskretizuje (obvykle uživatel musí specifikovat rozsah zobrazované oblasti) a použije se některá metoda extrakce iso-povrchu pro objemová data (metody pro zpracování CT, MRI dat), nebo
- povrch se zobrazí metodou sledování paprsku (Ray Tracing), viz kap.15.1 (Metoda sledování paprsku) apod.

Pomocí výše uvedeného postupu můžeme konstruovat i velmi složité geometrické objekty, pokud umožníme do stromové struktury vkládat také geometrické operace, např. posuv, rotace atd.

Příklad

Úkolem je vytvoření objektu, který je určen jako průnik 3 válců o stejném průměru (dostatečně dlouhé nebo „nekonečné“), jejichž osy jsou navzájem na sebe kolmé. V případě použití trojúhelníkové sítě je zásadní problém volby velikosti trojúhelníků, které nahradí jednotlivé válce tak, aby průsečnice byla „dostatečně“ hladká. Při řešení je pak nutné očekávat „explozi“ generování malých trojúhelníků v místě průsečíku jednotlivých ploch.

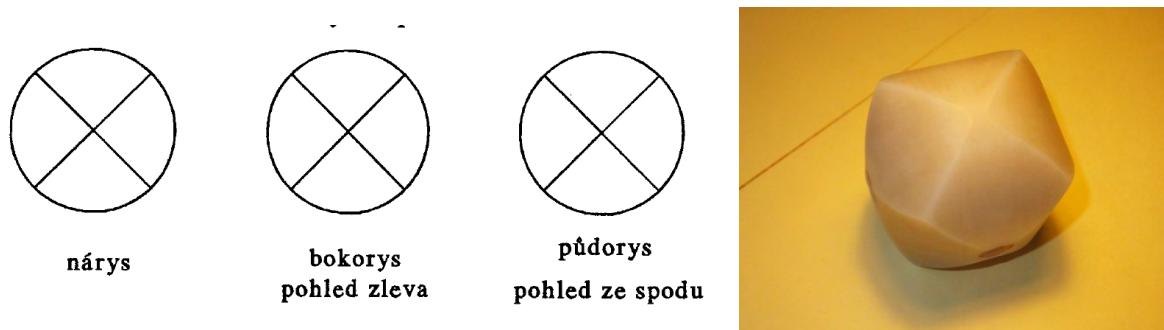
- 1 Předpokládejme, že geometrický objekt je popsán následujícím CSG stromem.



2
3

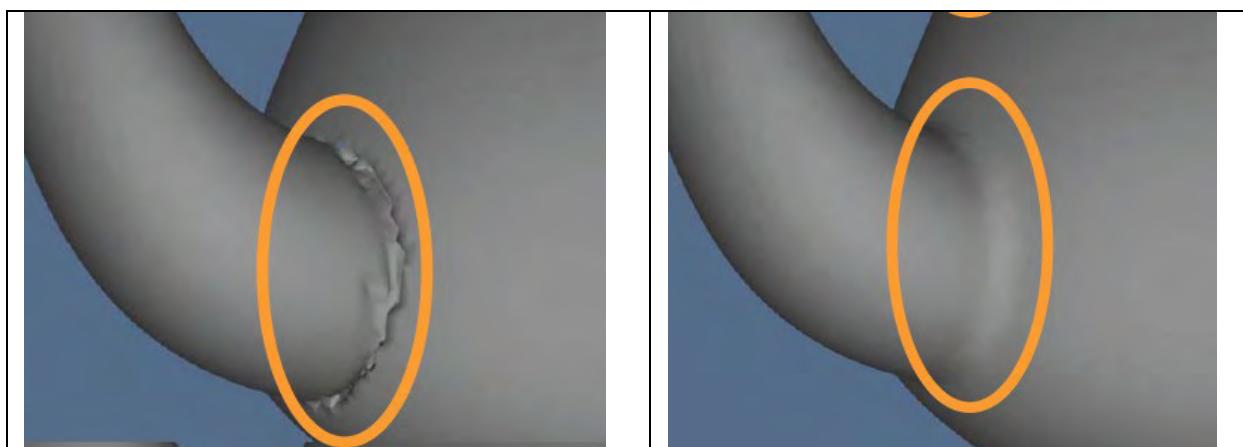
Obr.7.23: Ukázka CSG stromu

- 4 Pokud výsledné těleso zobrazíme v paralelní projekci, pak dostáváme jednotlivé pohledy. Z pohledu
5 je zřejmé, že výsledné těleso by se mělo „koulet“, nicméně má jakési hrany. Nejde tedy o kouli, jak
6 bývá prvotní odhad, ale o těleso, které se může „válet“ ve třech na sebe kolmých směrech.



Obr.7.24: Průnik 3 ortogonálních válců

7



Obr.7.25: „Nehladkost“ při množinových operacích
<http://www.cad.zju.edu.cn/home/zhx/GM/012/00-ism.pdf>

8
9
10
11
12

- Nezbytnou součástí každého geometrického elementu u CSG stromů je informace o minimálním intervalu, resp. minimálním ohraňujícím objemu, který daný objekt zahrnuje. Je nutné zdůraznit, že např. pro rotační paraboloid v základní poloze je interval v jednom směru $(0, \infty)$, interval však může obecně být $(-\infty, +\infty)$.

1 Optimalizace CSG stromů

2 Při realizaci složitých objektů je nutné mít na paměti, že dva různé CSG stromy mohou reprezentovat
3 stejný objekt. To znamená, že různí uživatelé mohou vytvořit pro stejné zadání různé CSG stromy.
4 Efektivita při výpočtech a zobrazování je však zásadně ovlivněna pořadím jednotlivých operací. Navíc
5 při zobrazování povrchu objektu je vhodné se omezit na prostor, ve kterém se objekt nalézá, resp.
6 který se bude zobrazovat.

7 Z výše uvedeného vyplývá, že CSG strom je vhodné transformovat tak, aby např. operace průniku
8 byly co nejbliže k listům CSG stromu a naopak operace sjednocení co nejvíše, tj. ke kořeni stromu.
9

10 Množinové operace se aplikují také na ohraničující objekty a může dojít k situacím:

- 11 • při *operaci průniku*:
 - 12 ○ průnik ohraničujících objektů je prázdný, pak i odpovídající podstrom může být ze
13 struktury vypuštěn a jde pravděpodobně o chybový stav
 - 14 ○ obalové objekty podřízených podstromů mají neprázdný průnik a mohou se
15 rozpadnout na množinu geometricky menších obalových objektů
- 16 • při *operaci sjednocení*:
 - 17 ○ obalové objekty mají prázdný průnik – výsledek operace sjednocení „zdědí“
18 původní obalové objekty
 - 19 ○ obalové objekty mají neprázdný průnik – je vhodné je redukovat, resp. nahradit
20 tak, aby výsledná množina obalových objektů byla prostorově co nejmenší
- 21 • a mnoho dalších

22 Je zřejmé, že optimalizací CSG stromů lze docílit podstatné úspory ve výpočetních náročích a vyplatí
23 se tedy optimalizaci CSG stromu realizovat.
24

25 V mnoha případech, které jsou řešeny pomocí průsečíku přímky s objektem, se přímka vyjadřuje
26 v parametrické formě a následně se pak pracuje s intervaly parametrů, se kterými se dělají
27 množinové operace, např. průnik, sjednocení atd. Tohoto postupu se zejména využívá při aplikaci
28 metody sledování paprsku.
29

30 Reference

- 31 • Langtepe,E., Zachmann,G.: Geometric Data Structures for Computer Graphics, A.K.Peters,
32 2006

33 Pro přenos geometrických dat se používá mnoho formátů, mnohé z nich byly standardizovány.
34 Jedním ze základních a jednoduchých formátů je formát STL.
35

7.8.STL formát

STL formát (STereoLithography) je formát používaný v CAD/CAM systémech (CAD - Computer Aided Design, CAM - Computer Aided Manufacturing) a v systémech, které se označují termínem „rapid prototyping“ pro účely 3D tisku. Základní formát STL zachycuje pouze geometrii 3D objektů bez ostatních atributů jako je barva, textura apod. STL formát je definován jak ve formě textových dat, tj. *ASCII STL*, tak i pro binární data, tj. *binary STL*. Povrch objektu je definován množinou samostatných trojúhelníků, tedy nikoliv trojúhelníkovou sítí, posloupností dat pro každý trojúhelník, a to:

normálový vektor – 3 hodnoty
vrchol₁ – 3 hodnoty souřadnicové, tj. $[x, y, z]^T$
vrchol₂ – 3 hodnoty souřadnicové
vrchol₃ – 3 hodnoty souřadnicové
:
normálový vektor – 3 hodnoty
vrchol₁ – 3 hodnoty souřadnicové
vrchol₂ – 3 hodnoty souřadnicové
vrchol₃ – 3 hodnoty souřadnicové

Orientace normálového vektoru by vždy měla být taková, že normála plochy směruje ven z objektu.

Některé nové modifikace STL formátu umožňují zahrnutí i atributů jako je barva, průsvitnost atd., což je důležité zejména pro 3D tisk (3D print nebo Rapid Prototyping), kde se barva již běžně používá.

Základní nevýhody STL formátu (podle původní definice standardu) jsou:

- všechny souřadnice vrcholů musí být nezáporné
- jde o množinu samostatných trojúhelníků, takže není žádná informace o sousedních trojúhelnících ani o trojúhelnících sdílejících daný vrchol
- souřadnice vrcholů se vyskytují mnohonásobně, neboť se souřadnice vrcholů uvádějí pro každý trojúhelník zvlášť
- obtížná kontrola, zda objekt je konzistentní a zda plocha je uzavřená, tj. že tvoří objem bez „průrvy“
- náchylnost k nekonzistenci vlivem nepřesných dat v souřadnicích vrcholů
- neobsahuje žádné informace o měřítku, jednotkách ([mm] x [inch]) atd.

Přes všechny nevýhody je STL formát používaný pro svoji jednoduchost. Tento formát je postupně nahrazován formátem PLY. Formát PLY byl vyvinut na Stanford University pro zpracování dat velkého rozsahu z 3D skenování sochy Davida a následnou rekonstrukci povrchu. Tento formát je znám též pod názvy „Polygon File Format“ nebo „Stanford Triangle Format“, viz [http://en.wikipedia.org/wiki/PLY_\(file_format\)](http://en.wikipedia.org/wiki/PLY_(file_format)).

7.9. Algoritmy výpočtu průsečíků a ohraničující tělesa

V geometrických algoritmech se velmi často používá výpočet průsečíků geometrických entit, např. přímka s rovinou, přímka s koulí, trojúhelník s trojúhelníkem atd. Pro urychlení výpočtu se používají různé techniky k urychlení detekce, zda průsečík existuje. K tomuto slouží jednoduchá obalová tělesa, jako je kvádr nebo koule. V případě složitých objektů, resp. dlouhých objektů, se používá více ohraničujících těles, přičemž mohou mít obdobnou strukturu jako CSG stromy. V převážné většině je výpočet průsečíků založen na parametrickém tvaru přímky nebo na implicitní rovnici definující danou entitu.

V následujícím budou ukázány základní techniky pro detekci průsečíku s přímkou.

Průsečík přímky s rovinou

Přímka je dána v parametrickém tvaru:

$$\mathbf{X}(t) = \mathbf{X}_A + \mathbf{S} t \quad (7.8)$$

a rovnice roviny ρ :

$$aX + bY + cZ + d = 0 \quad (7.9)$$

tj.

$$\mathbf{a}^T \mathbf{X} + d = 0 \quad (7.10)$$

Dosazením rovnice přímky do rovnice roviny dostaváme:

$$\mathbf{a}^T (\mathbf{X}_A + \mathbf{S} t) + d = 0 \quad (7.11)$$

pak průsečík přímky s rovinou je určen hodnotou parametru t :

$$t = -\frac{\mathbf{a}^T \mathbf{X}_A + d}{\mathbf{a}^T \mathbf{S}} \quad (7.12)$$

Pokud přímka je kolmá na normálu roviny, pak $t = \pm\infty$. To znamená, že je zde numerický problém a otázka stability výpočtu. Zkusme analyzovat, jak by řešení vypadalo v případě reprezentace v projektivním prostoru. Přímka je nyní dána rovnicí:

$$\mathbf{x}(t) = \mathbf{x}_A + \mathbf{s} \tau \quad (7.13)$$

kde $\mathbf{x}_A = [x_A, y_A, z_A; w_A]^T$ a $\mathbf{s} = [s_x, s_y, s_z; s_w]^T$.

Rovnice roviny ρ v projektivním prostoru je dána (původní rovnici roviny vynásobíme $w \neq 0$):

$$ax + by + cz + dw = 0 \quad (7.14)$$

tj. :

$$\mathbf{a}^T \mathbf{x} = 0 \quad (7.15)$$

Vidíme tedy, že jde o více kompaktní formu i z hlediska datové reprezentace. Pak průsečík je určen:

$$\mathbf{a}^T (\mathbf{x}_A + \mathbf{s} \tau) = 0 \quad (7.16)$$

a tedy:

$$\tau = -\frac{\mathbf{a}^T \mathbf{x}_A}{\mathbf{a}^T \mathbf{s}} \quad (7.17)$$

Hodnotu parametru pro průsečík můžeme reprezentovat přímo v homogenních souřadnicích a pak:

$$\boldsymbol{\tau} \triangleq [-\mathbf{a}^T \mathbf{x}_A: \mathbf{a}^T \mathbf{s}]^T \quad (7.18)$$

kde \triangleq je projektivní ekvivalence.

Je zřejmé, že v případě projektivní reprezentace není problém s operací dělení ani se stabilitou výpočtu, neboť operace dělení je „odložena“ včetně rozhodnutí o singularitě.

V následujícím textu budou použity pouze Eukleidovské souřadnice. Modifikace jednotlivých algoritmů pro projektivní prostor jsou jednoduché.

1 **Průsečík přímky s kvádrem** (AABB - Axis Aligned Bounding Box)

2 Algoritmus průsečíku přímky s kvádrem, jehož hrany jsou rovnoběžné s osami souřadného systému,
 3 je velmi jednoduchý, neboť jde vlastně o Cyrus-Beck algoritmus v extrémní podobě, kdy rovnice
 4 jednotlivých rovin jsou ve tvaru:

$$x = \pm a \quad \text{nebo} \quad y = \pm b \quad \text{nebo} \quad z = \pm c \quad (7.19)$$

5 a přímka je dána v parametrickém tvaru:

$$\mathbf{x}(t) = \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A)t = \mathbf{x}_A + \mathbf{s}t \quad (7.19)$$

6 V případě algoritmu sledování paprsku, viz kap.15.1 (Metoda sledování paprsku), pak hledáme jen
 7 hodnotu t_{min} , tj. bodu nejbližší k počátku paprsku, tj. polopřímky. AABB box má, přes svoji extrémní
 8 výpočetní jednoduchost, zásadní nevýhodu - není rotačně invariantní a není vhodný pro dlouhé štíhlé
 9 objekty v obecné poloze. Proto se někdy přímka transformuje do polohy, kdy je objekt v základní
 10 poloze, resp. v poloze, která umožnuje, aby AABB byl co nejmenší.

11

12 **Ohraničující koule** (Bounding sphere)

13 Koule jako ohraničující těleso má zásadní výhodu v tom, že je rotačně invariantní. Předpokládejme, že
 14 máme přímku p v Eukleidovském prostoru E^3 danou rovnicí

$$\mathbf{x}(t) = \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A)t = \mathbf{x}_A + \mathbf{s}t \quad (7.20)$$

15 a povrch koule je dán vztahem:

$$(\mathbf{x} - \mathbf{x}_s)^T(\mathbf{x} - \mathbf{x}_s) - r^2 = 0 \quad (7.21)$$

16 Dosazením výrazu pro přímku do rovnice pro kouli pak dostáváme:

$$(\mathbf{x}_A + \mathbf{s}t - \mathbf{x}_s)^T(\mathbf{x}_A + \mathbf{s}t - \mathbf{x}_s) - r^2 = 0 \quad (7.22)$$

17 a tedy:

$$(\mathbf{s}t + \xi)^T(\mathbf{s}t + \xi) - r^2 = 0 \quad \xi = \mathbf{x}_A - \mathbf{x}_s \quad (7.23)$$

18 tedy kvadratickou rovnici a její řešení:

$$\mathbf{s}^T \mathbf{s} t^2 + 2\mathbf{s}^T \xi t + \xi^T \xi - r^2 = 0 \quad (7.24)$$

$$at^2 + bt + c = 0 \quad t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (7.25)$$

19 Je zřejmé, že pokud:

$$b^2 - 4ac < 0 \quad (7.26)$$

20 pak průsečík a ani dotyk přímky s koulí neexistuje. Pokud se počítá větší počet průsečíků se stejnou
 21 přímou, je výpočetně výhodné normalizovat vektor \mathbf{s} , tj. $\|\mathbf{s}\| = 1$, čímž se výpočet opět zjednoduší.

22

23 **Úlohy**

24 Porovnejte výpočetní náročnost standardního algoritmu s postupem, kdy se scéna transformuje tak,
 25 že koule má střed v počátku, tj. $\mathbf{x}_s = \mathbf{0}$ a vektor \mathbf{s} je normalizován, tj. $\|\mathbf{s}\| = 1$. Optimalizujte detekci
 26 průsečíku přímky s koulí.

27

28 Zkuste odpovědět na otázku, zda, např. pro metodu sledování paprsku, není výhodnější všechna
 29 tělesa reprezentovat v základní poloze a počítat průsečíky objektů, resp. obalových těles, s přímou,
 30 která je pro dané těleso odpovídajícím způsobem transformována.

31

32 Dosud byly popisovány základní datové struktury pro reprezentaci objektů. Avšak v počítačové
 33 grafice se používají k zobrazení ještě další entity, atributy, např. barva, průhlednost atd. Navíc scéna
 34 se obvykle skládá z více objektů, světelných zdrojů atd. To znamená, že potřebujeme reprezentovat
 35 také zobrazovanou scénu, viz kap.9 (Reprezentace scény).

1 8. Polygonální síťe – dodělat

2

3

4

5 Dopsat !!!

6

7

8 8.1. Progressive mesh refinement - dopsat

9

10

11

12 Dopsat !!!

13 8.2. Metody redukce trojúhelníkových sítí - dopsat

14

15

16 Dopsat !!!

17

**18 8.3. Diskrétní charakteristiky polygonálních povrchů (křivosti apod.) -
19 dopsat**

20

21

22

20 Dopsat !!!

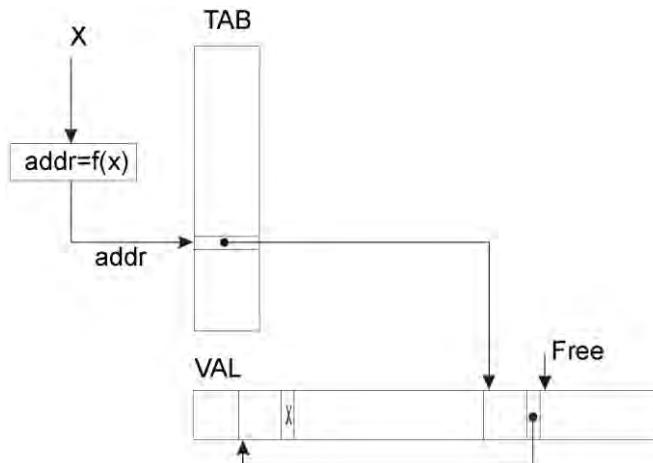
8.4. Hashing a eliminace duplicit

V počítačové grafice se jako paměťové struktury používají vícerozměrná pole, stromy, fronty a zásobníky, seznamy atd. Jednou z opomíjených datových struktur je hashing, která se velmi často používá při zpracování textu. Hash datová struktura má výhodu, že v ideálním případě, tzv. „perfect hashing“, je uložení hodnoty a její nalezení složitost $O(1)$. Zkusme se podívat, jaké jsou rozdíly mezi textovými data a daty geometrickými.

		Dimenzionalita	
		Malá	„Nekonečná“
Interval hodnot	Malý	Obrazová data Dimenze 2, 3 Interval hodnot $\langle 0,255 \rangle$ apod.	Textová data Dimenze, tj. délka řetězce dlouhá Interval hodnot, např. $\langle 0,255 \rangle$ pro ASCII
	„Nekonečný“	Geometrická data Dimenze 2, 3 Interval hodnot $(-\infty, +\infty)$	Harmonická analýza apod.

Tab. xx: Rozdílnost textových a geometrických dat

- textová data – rozsah hodnot je dán použitou alfabetou, např. 256 pro ASCII, zatímco dimenzionalita je „nekonečná“, neboť řetězce mohou být velmi dlouhé, např.:
 - protein titin je popsán 189,819 znaky
 - železniční stanice *Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch* ve Walesu atd.
- a počet slov, tj. zpracovávaných elementů je např. pro český slovník cca 2-3 mil. slov
- geometrická data – obvykle mají jen 2, resp. 3 souřadnice $\langle x, y \rangle$, resp. $\langle x, y, z \rangle$, ale „nekonečný“ interval hodnot $(-\infty, +\infty)$, přičemž běžně zpracováváme $10^6 - 10^{15}$ hodnot



Obr. 8.1: Hash datová struktura

Princip hashování je poměrně jednoduchý. Z dané hodnoty (klíče v textovém pojetí) se pomocí hashovací funkce určí adresa, kam se hodnota uloží. Je tedy zřejmé, že klíčovou otázkou je návrh hashovací funkce, která by neměla pro dvě různé hodnoty dávat stejnou adresu do hashovací tabulky. V případě geometrických dat však máme souřadnice x, y, z a interval hodnot je „nekonečný“. Standardní hashovací funkce obvykle využívají operaci **mod**, která je vlastně kompozicí násobení, dělení a převodu na celé číslo:

$$a \bmod p = a - p * \left\lfloor \frac{a}{p} \right\rfloor \quad (8.1)$$

1 Lze ukázat, že dobrou hashovací funkcí pro geometrická data je funkce:

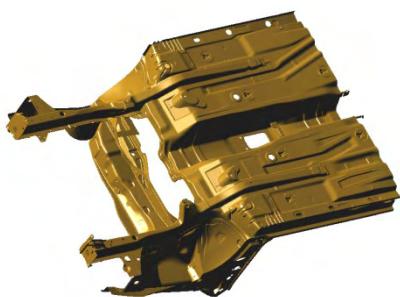
$$f(x, y, z) = h \text{ and } (2^q - 1) \quad h = C(\alpha x + \beta y + \gamma z) \text{ and } (2^k - 1) \quad (8.2)$$

2 kde: α, β, γ jsou malé iracionální hodnoty, např. $\frac{1}{3}, \pi, e, \sqrt{2}, \sqrt{3}$, C je měřítková konstanta závislá na
3 předpokládaném rozsahu hodnot, např. v zpracovávaném datovém setu, k je konstanta hardwarově
4 závislá (32/64 bitová reprezentace pro integer a počet bitů mantisy), q určuje délku hashovací
5 tabulky, která je délky 2^q .

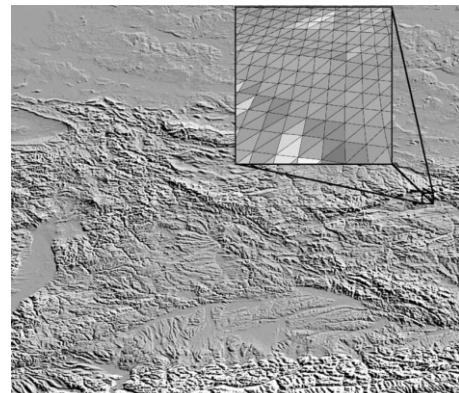
6 Algoritmus je založen na tom, že z hodnot $\langle x, y, z \rangle$ pomocí hash funkce se určí adresa do „virtualní“
7 hash tabulky $h \in \langle 0, 2^k - 1 \rangle$. Tento interval se pak mapuje na interval adres fyzické hashovací
8 tabulky, tj. na interval $\langle 0, 2^q - 1 \rangle$. Podrobně viz:

- Glassner, A.: Building vertex normals from an unstructured, polygon list. In: Graphic Gems IV. Academic Press, New York, pp. 60–73, 1994
- Hradecký, J., Skala, V.: Hash Function and Triangular Mesh Reconstruction, Vol.29, No.6., pp.741-751, Computers&Geosciences, Pergamon Press, ISSN 0098-3004, 2003

13 Jako příklad uvedeme vrcholy trojúhelníků, kdy z STL formátu bylo nutné eliminovat duplicitu a
14 následně vytvořit trojúhelníkovou síť pro podvozek automobilu.

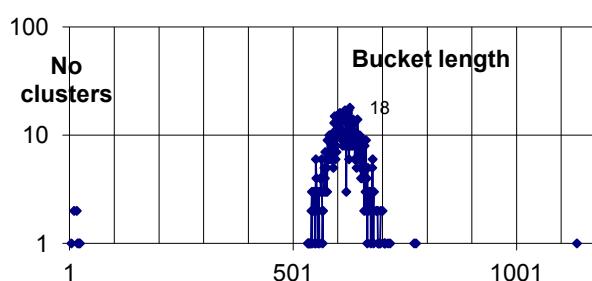


Obr.8.2: Datový set podvozku automobilu

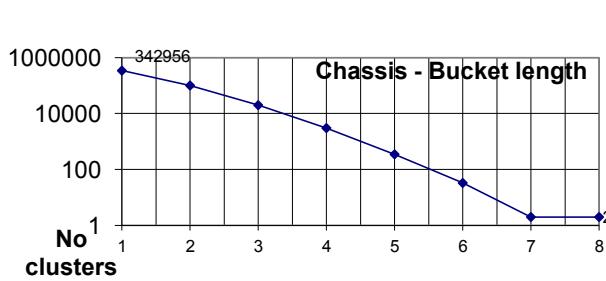


Obr.8.3: reprezentace povrchu Země

15
16 Pro efektivní činnost hashování je nutné analyzovat délku vznikajících seznamů



Obr.8.4: „Standardní“ hashing funkce



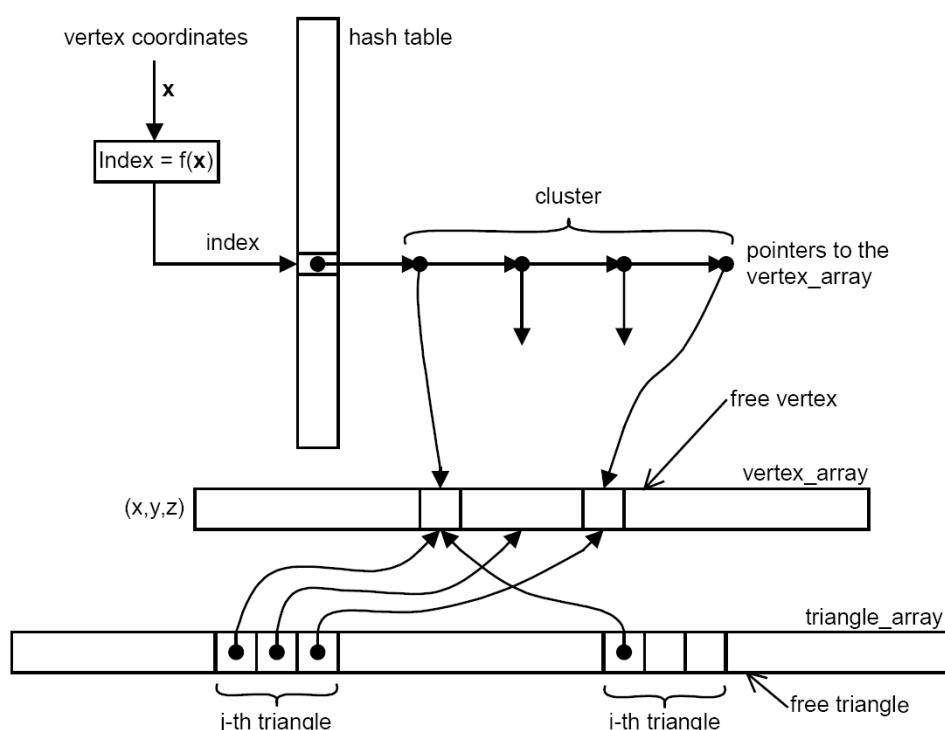
Obr.8.5: Vhodně navržená hash funkce

17
18 Z histogramu délky zřetězených hodnot je zřejmé, že v případě „standardní“ hash funkce
19 musíme provést cca 600 dotazů, zda je bod uložen v datové struktuře, zatímco s popsanou hash
20 funkcí je dotazů podstatně méně, jen cca 2-4, což znamená podstatně urychlení.
21 Hashování lze velmi dobře použít k odstranění duplicit v datovém setu. Při běžném přístupu
22 algoritmus zřejmě bude $O(n^2)$, nebo při použití řazení $O(n \lg n)$. Při použití hashování pak můžeme
23 dosáhnout $O_{expected}(n)$.

8.5. Rekonstrukce trojúhelníkové sítě z množiny trojúhelníků

V praxi je velmi častý případ, kdy z množiny samostatných trojúhelníků potřebujeme vytvořit trojúhelníkovou síť, např. s použitím okřídlené hrany, viz kap.7.5 (Hraniční reprezentace). Toto je už poněkud komplikovanější úloha, neboť musíme uvážit to, že počet zpracovávaných trojúhelníků bude běžně $n \in \langle 10^5, 10^{12} \rangle$ a počet vrcholů $m \cong 3n$. Vytvořením trojúhelníkové sítě, kdy je každý vrchol uložen pouze jednou, podstatně zredukujeme paměťové nároky.

Princip algoritmu je poměrně jednoduchý. Pro každý trojúhelník se uloží do tabulky souřadnice každého vrcholu trojúhelníka, pokud tam ještě nebyl. Nyní máme identifikaci, kde jsou všechny tři souřadnice trojúhelníka uloženy. Do seznamu trojúhelníků se uloží informace o vrcholech a k vrcholům se přidá informace na právě zpracovávaný trojúhelník. Po zpracování všech trojúhelníků se dopočtou všechny ostatní požadované vazby, např. sousednost trojúhelníků.



Obr.8.6: Datová struktura pro rekonstrukci trojúhelníkové sítě

Úlohu rekonstrukce trojúhelníkové sítě z množiny samostatných trojúhelníků už není možno realizovat jednoduchou modifikací algoritmu pro eliminaci duplicit se složitostí $O(n \lg n)$. Takže máme nyní pouze možnost algoritmu brutální síly se složitostí $O(n^2)$, nebo použití hashování se složitostí $O_{expected}(n)$. Je zřejmé, že algoritmus se složitostí $O(n^2)$ je absolutně nepoužitelný vzhledem ke zpracovávanému počtu trojúhelníků.

- Podrobně viz:
- Glassner, A.: Building vertex normals from an unstructured polygon list, Graphic Gems IV. Academic Press, New York, pp. 60–73, 1994
 - Hradek,J., Skala,V.: Hash Function and Triangular Mesh Reconstruction, Vol.29, No.6., pp.741-751, Computers&Geosciences, Pergamon Press, ISSN 0098-3004, 2003

1 9. Reprezentace scény

2 Zobrazovaná scéna v reálné situaci obsahuje nejen popisem geometrických objektů, ale jsou též
3 potřebné další informace, např.:

- 4 • pozice kamery, resp. pozorovatele, její směrovou orientaci, vlastnosti kamery, např.
5 ohnisková vzdálenost atd.
- 6 • světelné poměry na scéně, např. pozice a charakteristika světelných zdrojů, případná fixace
7 světelného zdroje na objekt, nebo skupinu objektů, kdy se při geometrické transformaci mění
8 s objektem i pozice zdroje světla
- 9 • hierarchická struktura geometrie scény a atributy jednotlivých geometrických objektů, např.
10 textury, průhlednost

11 Graf scény umožnuje efektivně budovat komplikované scény, jednoduchou editaci jejich částí a
12 selektivní manipulaci s vybranými objekty ve scéně.

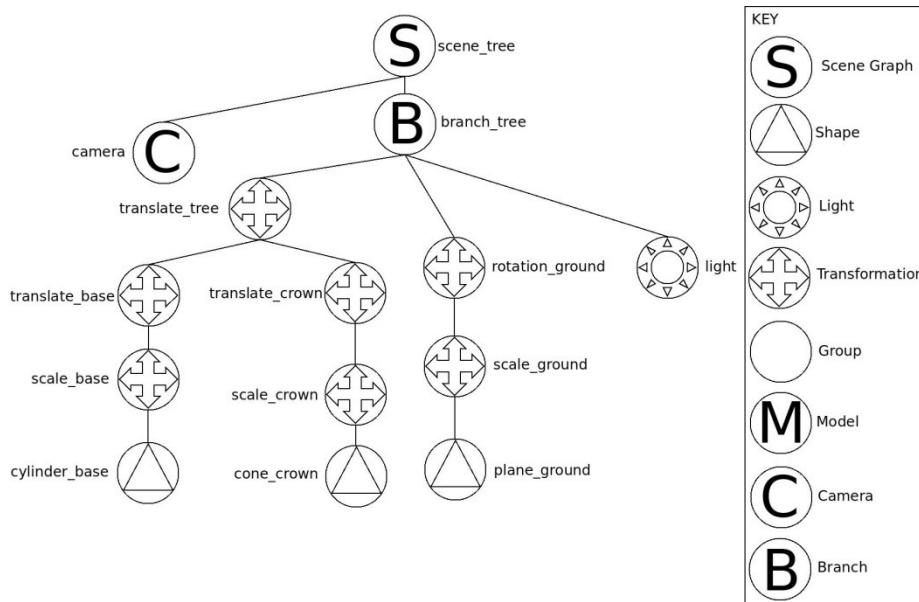
13 9.1. Graf scény (Scene Graph)

14 Vlastnosti scény a objektů ve scéně jsou zaznamenány ve struktuře, která se nazývá graf scény (*scene
15 graph*) a která je primárně tvořena stromovou strukturou, podobně jako u CSG stromu, avšak
16 geometrické objekty mohou být definovány i jiným než funkcionálním popisem. Základní vlastností
17 grafu scény je pak to, že jak transformace, tak i atributy atd. se „dědí“ na podřízené uzly stromu,
18 pokud není stanoveno jinak.

19

20 Pro ilustraci uvažme jednoduchý graf scény (viz <http://www.disgruntledrats.com/?p=266>):

21



22 Obr.9.1: Ukázka grafu scény

23

24

9.2. Reprezentace grafu scény

Graf scény pak může být zapsán ve formě XML, např.:

```

1<?xml version="1.0"?>
2<scenegraph name="scene_tree">
3  <branch name="branch_tree">
4    <translation name="translate_tree" x="200.0" y="0.0" z="-100.0">
5      <translation name="translate_base" x="0.0" y="5.0" z="0.0">
6        <scale name="scale_base" x="5.0" y="5.0" z="5.0">
7          <shape name="cylinder_base" type="cylinder" color="brown" />
8        </scale>
9      </translation>
10     <translation name="translate_crown" x="0.0" y="15.0" z="0.0">
11       <scale name="scale_crown" x="15.0" y="15.0" z="15.0">
12         <shape name="cone_crown" type="cone" color="green" />
13       </scale>
14     </translation>
15   </translation>
16   <rotation name="rotation_ground" ray_x="1" ray_y="0" ray_z="0"
17     angle="90">
18     <scale name="scale_ground" x="1000.0" y="1000.0" z="1000.0">
19       <shape name="plane_ground" type="plane" color="gray" />
20     </scale>
21   </rotation>
22   <directional_light_source name="light" x="20.0"
23     y="300.0" z="20.0" />
24 </branch>
25 <camera name="camera" cop_x="0" cop_y="0" cop_z="0" vrp_x="20"
26   vrp_y="150" vrp_z="20" vuv_x="0" vuv_y="1" vuv_z="0" />
27 </scenegraph>
28
29
30
31
32
33
34 Zásadní výhody grafu scény a jeho použití:
35
36   • je především snadná realizace úprav scény, tedy možnost editace, i pro velmi rozsáhlé scény
37   • možnost „klonování“ skupin objektů, tedy vlastně podstromů
38   • jednoznačný popis scény a možnost optimalizace grafu scény podobně jako u CSG stromů
39
40 Až dosud jsme se zabývali víceméně geometrickými vlastnostmi geometrických objektů s relevantním
41 matematickým popisem. Při zobrazování scén je nutné brát v úvahu také světelné poměry ve scéně.
```

10. Interpolace a approximace uspořádaných a neuspořádaných dat

Interpolace a approximace dat je opět jednou z velmi používaných operací v počítačové grafice. Obecně se pojmem interpolace označují ty metody, kdy výsledný interpolant zadanými body prochází. Např. u Bézierovy křivky interpolovaná křivka prochází body x_0, x_{n-1} , zatímco body x_1, \dots, x_{n-2} jsou řídící body, které určují chování křivky. Pojem approximace se označují ty metody, kdy výsledný interpolant zadanými body nemusí procházet a většinou ani neprochází, ale dané hodnoty approximuje podle zadaného kritéria, např. metodou nejmenších čtverců. Někdy se obě skupiny označují jako interpolace, avšak chápané v širším slova smyslu. V následujícím budou probrány základní metody interpolací a approximací s ohledem na možnosti aplikace v počítačové grafice a vizualizaci dat.

Asi ve všech publikacích pojednávajících o interpolacích se uvádí Lagrangeova interpolace. Předpokládejme, že máme $n + 1$ navzájem různých bodů x_0, \dots, x_n a hodnoty dané v těchto bodech, tj. $f(x_0), \dots, f(x_n)$. Pak Lagrangeův interpolační polynom je určen:

$$L_n(x) = f(x_0)l_0(x) + \dots + f(x_n)l_n(x) \quad (10.1)$$

kde:

$$l_i(x_j) = \begin{cases} 1 & \text{pokud } i = j \\ 0 & \text{pokud } i \neq j \end{cases}$$

Tyto podmínky např. splňuje polynom:

$$l_i(x) = \prod_{\substack{0 \leq k \leq n \\ k \neq i}} \frac{x - x_k}{x_i - x_k} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n} \quad (10.2)$$

Vzhledem k výpočetní náročnosti se využívá rovnosti, a to:

$$L_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = \sum_{i=0}^n a_i x^i = 1 \quad (10.3)$$

ke konstrukci matice $\Lambda_{n+1,n+1}$, reprezentující Lagrangeův polynom. Pak lze koeficienty a_i vypočítat řešením soustavy lineárních rovnic $\Lambda\alpha = \beta$, kde:

$$\Lambda = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_i & x_i^2 & \cdots & x_i^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \quad \alpha = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} \quad \beta = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix} \quad (10.4)$$

a kde matice Λ je známá Vandermondova matice (Alexandre Théophile Vandermonde).

Nicméně je nutné poznamenat, že:

- pro větší počet bodů může nastat problém s numerickým řešením soustavy rovnic
- základní vlastností Lagrangeovy interpolace je její charakteristické „rozkmitávání“ se s rostoucím počtem bodů

Osobní poznámka – přímý dopad do běžného života:

Čím více se zavede zákonů, které lidé musejí respektovat,
tím větší je rozkmitávání chování lidí, tj. chaos.

10.1. Lineární interpolace

Lineární interpolace je zřejmě nejčastější interpolací v aplikacích počítačové grafiky. V zásadě jednotlivá interpolační schémata vycházejí buď z explicitní nebo implicitní rovnice přímky, roviny apod., nebo z parametrického tvaru. V následujícím textu se budeme zabývat parametrickou lineární interpolací.

Lineární interpolace lze rozdělit:

$$\text{Lineární interpolace} = \begin{cases} \text{lineární interpolace souřadnic} \\ \text{lineární interpolace hodnot} \end{cases} \quad (10.5)$$

Lineární interpolace hodnot je interpolací, kdy např. interpolujeme skalární hodnotu např. nad trojúhelníkovou nebo čtyřúhelníkovou základnou, tj. neinterpolujeme povrch. Tento typ se také někdy nazývá $2\frac{1}{2}D$ interpolací, podrobněji viz např. kap. 10.3 (RBF interpolace).

Parametrická lineární interpolace

Předpokládejme, že jsou dány dva body v E^n , a to \mathbf{X}_1 a \mathbf{X}_2 . Pak lineární interpolace, tj. „na přímce“, je dána vztahem:

$$\mathbf{X}(t) = \mathbf{X}_1 + (\mathbf{X}_2 - \mathbf{X}_1) t \quad (10.6)$$

kde: $t \in \langle 0,1 \rangle$ je parametr. Výše uvedený vztah se používá pro parametrické vyjádření přímky, resp. polopřímky nebo paprsku (ray), kdy parametr $t \in (-\infty, +\infty)$, resp. $t \in \langle 0, \infty \rangle$. Parametrického vyjádření přímky se s výhodou používá pro výpočet průsečíku přímky s rovinou, koulí apod.

V případě lineární interpolace „na rovinné ploše“ v prostoru E^n (v námi uvažovaných aplikacích je to téměř výlučně prostor E^2 nebo E^3) je interpolace určena vztahem:

$$\mathbf{X}(u, v) = \mathbf{X}_1 + (\mathbf{X}_2 - \mathbf{X}_1) u + (\mathbf{X}_3 - \mathbf{X}_1) v \quad (10.7)$$

kde: u, v jsou parametry. Výše uvedený vztah se používá pro parametrické vyjádření trojúhelníka, kde pro parametry platí: $u, v \in \langle 0, 1 \rangle$ & $0 \leq u + v \leq 1$.

V případě lineární interpolace „v objemu“ v prostoru E^n (v námi uvažovaných aplikacích je to téměř výlučně prostor E^3) je interpolace určena vztahem:

$$\mathbf{X}(u, v, w) = \mathbf{X}_1 + (\mathbf{X}_2 - \mathbf{X}_1) u + (\mathbf{X}_3 - \mathbf{X}_1) v + (\mathbf{X}_4 - \mathbf{X}_1) w \quad (10.8)$$

kde: u, v, w jsou parametry. Výše uvedený vztah se používá pro parametrické vyjádření čtyřstěnu, kde pro parametry platí $u, v, w \in \langle 0, 1 \rangle$ & $0 \leq u + v + w \leq 1$. Je to vlastně lineární interpolace v rovnoběžnostěnu.

Poznámka

Pro lineární interpolaci vektorů se používá operace `lrep()`:

$$\mathbf{c}(t) = \text{lrep}(\mathbf{a}, \mathbf{b}, t) \quad (10.9)$$

Barycentrické souřadnice

V mnoha aplikacích se používají barycentrické souřadnice bodu:

- na úsečce v E^1 , kde určují poměr délek
- v trojúhelníku v E^2 , kde určují poměr ploch
- ve čtyřstěnu (tetrahedronu) v E^3 , kde určují poměr objemů

1 Je nutné upozornit, že barycentrické souřadnice bodu v trojúhelníku, který je v obecné poloze v E^3
 2 nejsou přímo definovány, neboť souřadnice jsou ještě „svázány“ rovinou, na které daný trojúhelník
 3 leží. Toto se většinou řeší projekcí do „vhodné“ základní roviny souřadného systému.

4

5 Barycentrické souřadnice bodu na úsečce

6 Barycentrické souřadnice jsou definovány takto:

$$\begin{aligned}\lambda_1 X_1 + \lambda_2 X_2 &= X \\ \lambda_1 + \lambda_2 &= 1\end{aligned}\tag{10.10}$$

7 Pokud dosadíme $\lambda_1 = 1 - \lambda_2$ do rovnice, pak:

$$X_1 + \lambda_2(X_2 - X_1) = X\tag{10.11}$$

8 Vidíme tedy přímou souvislost mezi lineární interpolací a barycentrickými souřadnicemi, neboť platí:

$$t = \lambda_2\tag{10.12}$$

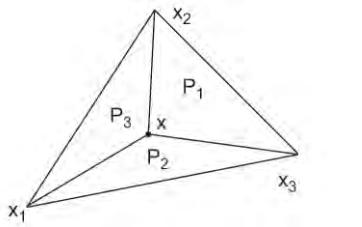
9

10 Barycentrické souřadnice bodu v trojúhelníku

11 Barycentrické souřadnice jsou dány vztahy:

$$\begin{aligned}\lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 &= X \\ \lambda_1 Y_1 + \lambda_2 Y_2 + \lambda_3 Y_3 &= Y \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1\end{aligned}\tag{10.13}$$

12 Pokud dosadíme $\lambda_1 = 1 - \lambda_2 - \lambda_3$ do rovnice, pak opět dostaneme lineární interpolaci na
 13 parametrické rovině $X(u, v)$. Barycentrická souřadnice λ_1 pak určuje poměr plochy trojúhelníka P_1 ,
 14 jehož vrcholem je daný bod X a plochy celého trojúhelníka.

$\lambda_1 = \frac{P_{XX_2X_3}}{P_{X_1X_2X_3}}$	$\lambda_2 = \frac{P_{X_1XX_3}}{P_{X_1X_2X_3}}$	$\lambda_3 = \frac{P_{X_1X_2X}}{P_{X_1X_2X_3}}$	
---	---	---	---

15 Je zřejmé, že určení hodnot barycentrických souřadnic vede obecně na řešení soustav lineárních
 16 rovnic $Ax = b$.

17

18 Výše uvedená lineární parametrická interpolace a interpolace barycentrickými souřadnicemi
 19 v Eukleidovském prostoru E^n je lineární vzhledem k parametru, resp. parametrům.

20

21 V počítačové grafice se používají homogenní souřadnice, tj. používáme projektivní rozšíření
 22 Eukleidovského prostoru. Otázkou tedy je, jak postupovat, pokud souřadnice bodů jsou dány
 23 v homogenních souřadnicích.

24

25 Interpolace v projektivním prostoru

26 Předpokládejme, že body jsou dány souřadnicemi v homogenních souřadnicích, a to:

- $x = [x: w]^T$ v případě E^1
- $x = [x, y: w]^T$ v případě E^2
- $x = [x, y, z: w]^T$ v případě E^3

30 a pro homogenní souřadnici *nemusí platit*, že $w = 1$. Obecně je možné převést souřadnice bodů do
 31 Eukleidovského prostoru, tj. vydělit souřadnice hodnotou homogenní složky w a následně použít
 32 lineární interpolaci nebo barycentrické souřadnice. Toto řešení je poměrně výpočetně náročné,

neboť se musí použít operace dělení. Otázkou je, zda je možné realizovat lineární interpolaci bez použití operace dělení.

Předpokládejme, že jsou dány dva body $\mathbf{x}_1 = [x_1, y_1: w_1]^T$ a $\mathbf{x}_2 = [x_2, y_2: w_2]^T$ v homogenních souřadnicích. Pak pro souřadnici X lze psát:

$$X(t) = \frac{x_1}{w_1} + \left(\frac{x_2}{w_2} - \frac{x_1}{w_1} \right) t \quad (10.14)$$

Pak zajisté je možné:

$$\mathbf{x}(t) = w_2 \mathbf{x}_1 + (w_1 \mathbf{x}_2 - w_2 \mathbf{x}_1) t \quad (10.15)$$

neboť homogenní souřadnice reprezentují „odložené“ dělení. Pak pro případ E^1 můžeme psát:

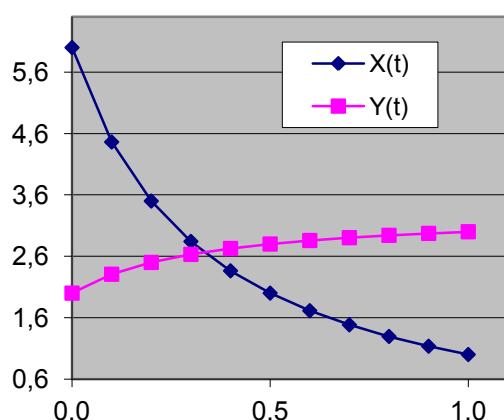
$$\begin{bmatrix} x \\ w \end{bmatrix}(t) = \begin{bmatrix} w_2 x_1 \\ w_1 w_2 \end{bmatrix} + \left(\begin{bmatrix} w_1 x_2 \\ w_1 w_2 \end{bmatrix} - \begin{bmatrix} w_2 x_1 \\ w_1 w_2 \end{bmatrix} \right) t \quad (10.16)$$

Analogicky i pro E^2 a E^3 . Parametrizace parametrem t je *lineární*, avšak je zapotřebí několik operací násobení.

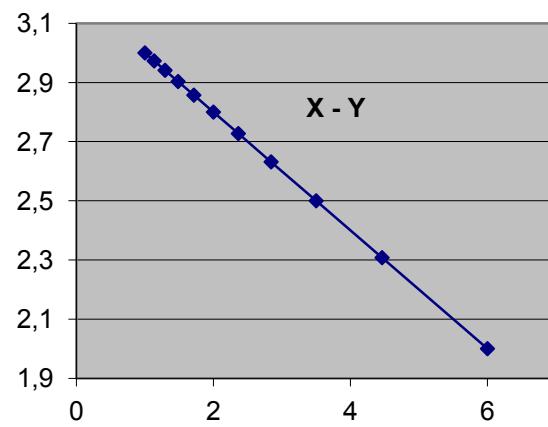
Otázkou je, zda je možné provést lineární interpolaci přímo v projektivním prostoru a jaké to má důsledky. Lineární interpolaci v projektivním prostoru, tj. body jsou dány v homogenních souřadnicích, můžeme zapsat:

$$\mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1) \tau = \mathbf{x}_1 + \mathbf{s} \tau \quad \mathbf{s} = [x_2 - x_1, y_2 - y_1: w_2 - w_1]^T \quad (10.17)$$

Lze ukázat, že výsledkem je přímka v projektivním prostoru s parametrizací danou parametrem τ . Z předchozího víme, zejména pak viz kap.5.1 (Perspektivní projekce), že obecně parametr $\tau \neq t$, kromě krajních bodů, neboť při převodu do Eukleidovského prostoru jde vlastně o perspektivní projekci. *Parametr τ monotónně roste nebo klesá, avšak **nelineárně**.* V tomto případě projektivní reprezentace bodů jde o *lineární interpolaci s nelineární monotónní parametrizací*.



Obr.10.1: $x(t), y(t)$ v závislosti na parametru t



Obr.10.2: Interpolace v prostoru (x, y)

Výše uvedený postup lze aplikovat obecně pro E^n a i pro rovinou interpolaci na ploše, tj.:

$$\mathbf{x}(u, v) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1) u + (\mathbf{x}_3 - \mathbf{x}_1) v \quad (10.18)$$

Takže lze vlastně lineární interpolace rozdělit takto:

lineární interpolace $\begin{cases} \text{v Eukleidovském prostoru s lineární parametrizací} \\ \text{v projektivním prostoru } \begin{cases} \text{s lineární parametrizací} \\ \text{s nelinární monotónní parametrizací} \end{cases} \end{cases}$

24

1 V mnoha algoritmech nepotřebujeme lineární parametrisaci, ale postačuje pouze monotónní
 2 parametrisace. Typickými ukázkami jsou:

- 3 metoda sledování paprsku (Ray tracing), viz kap.15.1 (Metoda sledování paprsku), kdy je
 4 nutné určit, který objekt protnutý polopřímkou, tj. paprskem, je nejblíže počátku polopřímky
- 5 • algoritmus Cyrus-Beck pro ořezávání přímky, resp. úsečky konvexním n-úhelníkem – vhodnou
 6 modifikací lze takto algoritmus zrychlit cca o 15%

7
 8 V předchozím výkladu byl ukázán výpočet barycentrických souřadnic, který vede na řešení soustavy
 9 lineárních rovnic $\mathbf{Ax} = \mathbf{b}$. Je tedy přirozenou otázkou, zda barycentrické souřadnice je možné počítat
 10 také přímo v projektivním prostoru. Vlastní odvození je nad rámec tohoto textu a lze jej najít v:

- 11 • Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers &
 12 Graphics, Elsevier, ISSN 0097-8493, Vol. 32, No.1, pp.120-127, 2008

13 Uvažme pro jednoduchost výpočet barycentrických souřadnic pro bod v trojúhelníku v E^2 . V kap. 3.3
 14 (Dualita a její aplikace) bylo ukázáno, že řešení soustav lineárních rovnic je ekvivalentní zobecněnému
 15 vektorovému součinu.

$$\lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 = X \quad \lambda_1 Y_1 + \lambda_2 Y_2 + \lambda_3 Y_3 = Y \quad \lambda_1 + \lambda_2 + \lambda_3 = 1 \quad (10.19)$$

17 18 Uvedenou soustavu rovnic lze přepsat do implicitní formy, tj.:

$$\begin{bmatrix} X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{aligned} \omega_1 X_1 + \omega_2 X_2 + \omega_3 X_3 + \omega_4 X &= 0 \\ \omega_1 Y_1 + \omega_2 Y_2 + \omega_3 Y_3 + \omega_4 Y &= 0 \\ \omega_1 + \omega_2 + \omega_3 + \omega_4 &= 0 \\ \lambda_i &= -\frac{\omega_i}{\omega_4} \end{aligned} \quad (10.20)$$

19 20 Řešení uvedené soustavy rovnic $\mathbf{Ax} = \mathbf{0}$ je ekvivalentní zobecněnému vektorovému součinu a lze
 21 tedy psát:

$$\boldsymbol{\omega} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \boldsymbol{\tau} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (10.21)$$

22 kde:

$$\boldsymbol{\xi} = [X_1, X_2, X_3, X]^T, \boldsymbol{\eta} = [Y_1, Y_2, Y_3, Y]^T, \boldsymbol{\tau} = [1, 1, 1, 1]^T, \boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$$

$$\mathbf{i} = [1, 0, 0, 0]^T, \mathbf{j} = [0, 1, 0, 0]^T, \mathbf{k} = [0, 0, 1, 0]^T, \mathbf{l} = [0, 0, 0, 1]^T$$

26 Je tedy zřejmé, že:

- 27 • pro výpočet barycentrických souřadnic není nutné řešit soustavu rovnic, např. Gaussovou
 28 eliminační metodou nebo nějakou jinou, a lze použít vektorový součin
- 29 • výpočet barycentrických souřadnic, pokud souřadnice jsou v homogenních souřadnicích je
 30 také jednoduchý, neboť determinant je multilineární

1 Výpočet barycentrických souřadnic pro body v projektivním prostoru:

2

$$\boldsymbol{\omega} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} = \frac{1}{w_1 w_2 w_3 w} \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ w_1 X_1 & w_2 X_2 & w_3 X_3 & wX \\ w_1 Y_1 & w_2 Y_2 & w_3 Y_3 & wY \\ w_1 & w_2 & w_3 & w \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ w_1 & w_2 & w_3 & w \end{bmatrix} \quad (10.22)$$

3 Také lze psát:

$$\boldsymbol{\omega} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \boldsymbol{\tau} \quad (10.23)$$

4 kde:

$$\boldsymbol{\xi} = [x_1, x_2, x_3, x]^T, \boldsymbol{\eta} = [y_1, y_2, y_3, y]^T, \boldsymbol{\tau} = [w_1, w_2, w_3, w]^T, \boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$$

$$\mathbf{i} = [1, 0, 0, 0]^T, \mathbf{j} = [0, 1, 0, 0]^T, \mathbf{k} = [0, 0, 1, 0]^T, \mathbf{l} = [0, 0, 0, 1]^T$$

7 Barycentrické souřadnice jsou pak dány:

$$\lambda_1 = -\frac{\omega_1}{\omega_2 \omega_3 \omega_4} \quad \lambda_2 = -\frac{\omega_2}{\omega_1 \omega_3 \omega_4} \quad \lambda_3 = -\frac{\omega_3}{\omega_1 \omega_2 \omega_4} \quad (10.24)$$

8

Poznámka

10 Z výše uvedeného je zřejmé, že výpočetní postup je jednoduchý, elegantní, nepotřebuje ke své
11 realizaci operaci dělení, pokud λ_1 vyjádříme projektivně, tj. např. $\lambda_1 = [-\omega_1 : \omega_2 \omega_3 \omega_4]^T$ atd.

12 Také je vhodný pro GPU aplikace, neboť vektorový součin je instrukcí GPU.

13

14 Dále pak uveďme:

- 15 • výše uvedený způsob výpočtu barycentrických souřadnic má *lineární parametrizaci*
- 16 • pro zjištění, zda bod je uvnitř trojúhelníka je vhodné použít modifikovaný test

$$0 \leq \omega_1 \leq -\omega_2 \omega_3 \omega \quad 0 \leq \omega_2 \leq -\omega_1 \omega_3 \omega \quad 0 \leq \omega_3 \leq -\omega_1 \omega_2 \omega \quad (10.25)$$

17 Obdobně lze postupovat i pro výpočet barycentrických souřadnic bodu ve čtyřstěnu v E^3 .

Poznámka

19 Rozšířený vektorový součin může být implementován na GPU např. takto:

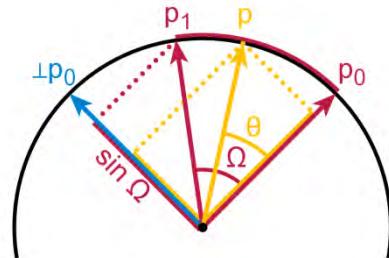
```
20
21 float4 cross_4D(float4 x1, float4 x2, float4 x3)
22 {
23     float4 a;
24     a.x = dot(x1.yzw, cross(x2.yzw, x3.yzw));
25     a.y = -dot(x1.xzw, cross(x2.xzw, x3.xzw));
26     a.z = dot(x1.xyw, cross(x2.xyw, x3.xyw));
27     a.w = -dot(x1.xyz, cross(x2.xyz, x3.xyz));
28     return a
29 }
```

31 Až dosud jsme se zabývali lineárními interpolacemi. Další specifickou interpolací je interpolace
32 sférická, která je interpolací „po oblouku“, resp. po kulové ploše, s **téměř lineární parametrizací**.

10.2. Sférická interpolace

2 Sférická interpolace je interpolací speciální a vychází
 3 z následujících předpokladů:

- 4 střed rotace je v počátku souřadného systému
- 5 pokud se interpolují normálové vektory (jsou
 6 vázány na příslušný vrchol), pak mají průsečík
 7 v počátku souřadného systému
- 8 interpoluje se po kulové ploše



Obr.10.3: Sférická interpolace

10 Sférická interpolace je interpolací s téměř lineární
 11 parametrizací. Interpolace *slerp()* je definována:

$$11 \quad \text{slerp}(\mathbf{X}_0, \mathbf{X}_1, t) = \frac{\sin[(1-t)\Omega]}{\sin \Omega} \mathbf{X}_0 + \frac{\sin[t\Omega]}{\sin \Omega} \mathbf{X}_1 \quad (10.26)$$

13 Je nutné upozornit, že pro $\Omega \rightarrow k\pi$, $k = 0, \pm 1, \dots$ je tato formule nestabilní, neboť se dělí hodnotou
 14 blízkou nule.

16 Proto je vhodné používat projektivní reprezentaci $\text{slerp}_p(\mathbf{x}_0, \mathbf{x}_1, t)$ a „odložit“ operaci dělení.

$$17 \quad \begin{aligned} \text{slerp}(\mathbf{X}_0, \mathbf{X}_1, t) &\triangleq \text{slerp}_p(\mathbf{X}_0, \mathbf{X}_1, t) = \left[\begin{array}{c} \sin[(1-t)\Omega] \mathbf{X}_0 + \sin[t\Omega] \mathbf{X}_1 \\ \sin \Omega \end{array} \right] \\ &= [\sin[(1-t)\Omega] \mathbf{X}_0 + \sin[t\Omega] \mathbf{X}_1 : \sin \Omega]^T \end{aligned} \quad (10.27)$$

18 Je určitě na místě se ptát, jak je sférická interpolace definována, pokud body jsou zadány obecně
 19 v homogenních souřadnicích. Pak lze psát pro x a w souřadnice (analogicky pro y, z):

$$19 \quad \begin{aligned} x(t) &= \frac{\sin[(1-t)\Omega]}{\sin \Omega} x_0 + \frac{\sin[t\Omega]}{\sin \Omega} x_1 \\ w(t) &= \frac{\sin[(1-t)\Omega]}{\sin \Omega} w_0 + \frac{\sin[t\Omega]}{\sin \Omega} w_1 \end{aligned} \quad (10.28)$$

21 Analogicky i pro souřadnice $x(t), y(t)$.

22 Pak interpolované souřadnice bodu vyjádřené v Eukleidovských souřadnicích jsou:

$$23 \quad \mathbf{X}(t) = \frac{\frac{\sin[(1-t)\Omega]}{\sin \Omega} x_0 + \frac{\sin[t\Omega]}{\sin \Omega} x_1}{\frac{\sin[(1-t)\Omega]}{\sin \Omega} w_0 + \frac{\sin[t\Omega]}{\sin \Omega} w_1} = \frac{\sin[(1-t)\Omega] x_0 + \sin[t\Omega] x_1}{\sin[(1-t)\Omega] w_0 + \sin[t\Omega] w_1} \quad (10.29)$$

24 Pokud sférickou interpolaci vyjádříme projektivně, pak:

$$25 \quad \mathbf{x}(t) = \begin{bmatrix} \sin[(1-t)\Omega] x_0 + \sin[t\Omega] x_1 \\ \sin[(1-t)\Omega] y_0 + \sin[t\Omega] y_1 \\ \sin[(1-t)\Omega] z_0 + \sin[t\Omega] z_1 \\ \sin[(1-t)\Omega] w_0 + \sin[t\Omega] w_1 \end{bmatrix} \quad (10.30)$$

26 To znamená, že máme lepší numerickou stabilitu.

10.3. RBF interpolace

2 V mnoha oblastech dříve uvedené metody nejsou aplikovatelné vzhledem k relativně specifickým
 3 předpokladům. Až dosud jsme se zabývali *interpolací uspořádaných dat* v různém kontextu. Nicméně,
 4 zejména v úlohách s vyšší dimenzí nebo v případě časově proměnných dat, kdy se musí vytvářet
 5 čtyřúhelníková nebo trojúhelníková síť na daném definičním oboru, se v současné době
 6 upřednostňují tzv. bezsíťové (Meshless, Meshfree) interpolační techniky.

7 Jednou z takových interpolačních technik je interpolace pomocí radiálních bázových funkcí (Radial
 8 Basis Functions), tzv. *RBF interpolate*. RBF interpolace je vhodnou metodou pro interpolaci
 9 *neuspořádaných dat*, např. skalárního potenciálového nebo vektorového pole. Jde tedy o úlohu, kdy
 10 jsou zadány souřadnice bodů $\mathbf{x}_i \in E^n$ a hodnoty $\mathbf{h}_i \in E^p$, které se mají interpolovat.

11 RBF interpolace je v postatě úlohou řešení soustavy lineárních rovnic $\mathbf{Ax} = \mathbf{b}$, kde \mathbf{b} je vektor
 12 zadaných hodnot a \mathbf{x} je vektor vah pro jednotlivé interpolační funkce. Výsledná interpolovaná
 13 hodnota je pak dána výrazem:

$$f(\mathbf{x}) = \sum_{i=1}^N \lambda_i \phi(r_i) \quad (10.31)$$

14 kde λ_i jsou váhy, $\phi(r_i) = \phi(\|\mathbf{x} - \mathbf{x}_i\|)$ jsou hodnoty interpolačních funkcí a \mathbf{x}_i jsou zadané hodnoty
 15 souřadnic i -tého bodu a $r_i = \|\mathbf{x} - \mathbf{x}_i\|$ je vzdálenost bodu \mathbf{x} od bodu \mathbf{x}_i .

16 V následujícím výkladu si ukážeme princip RBF interpolace, tj. kdy interpolační křivka nebo plocha
 17 prochází danými body. V případě RBF approximace approximační křivka nebo plocha nemusí nutně
 18 procházet danými body.

19 RBF interpolace má následující významné vlastnosti:

- 20 • RBF je určena především pro interpolaci neuspořádaných dat
- 21 • RBF interpolace je vhodná pro interpolaci d -rozměrných dat, $d \geq 2$
- 22 • RBF je založena na „parametrizaci“ závisející na vzdálenostech bodů $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, a tedy
 23 jde o interpolaci v principu *neseparabilní*, tj. není možné interpolovat po jednotlivých
 24 dimenzích

25 RBF interpolace používají rozdílné interpolační bázové funkce $\phi(r)$, tj. $\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. RBF interpolace
 26 lze rozdělit do dvou základních skupin, a to dle typů funkcí, které se pro interpolaci používají:

- 27 • **globální funkce**, které mají vliv na celý interval a které lze rozdělit na funkce, jejichž hodnota:
 - 28 ○ roste s rostoucí vzdáleností r , např. $\phi(r) = r^2 \lg r$, nebo $\phi(r) = r^3$
 - 29 ○ klesá s rostoucí vzdáleností r , např. $\phi(r) = \frac{1}{r^2}$, nebo $\phi(r) = e^{-\varepsilon r^2}$, kde ε je
 30 parametr

31 Při použití globálních funkcí je matice \mathbf{A} poměrně hodně zaplněna nenulovými hodnotami a
 32 podmíněnost soustavy rovnic se zhoršuje s rostoucím řádem matice.

- 33 • **lokální funkce** („Compactly Supported“ RBF-CSRBF), jejichž hodnoty jsou nenulové jen pro
 34 určitý interval, většinou pro $r \in (0,1)$. Některé typické CSRBF funkce jsou uvedeny dále.

35 Výhodou použití CSRBF je především to, že matice \mathbf{A} je maticí řídkou, což následně vede
 36 k podstatnému snížení výpočetní složitosti. Nicméně, vzhledem k tomu, že jejich vliv je

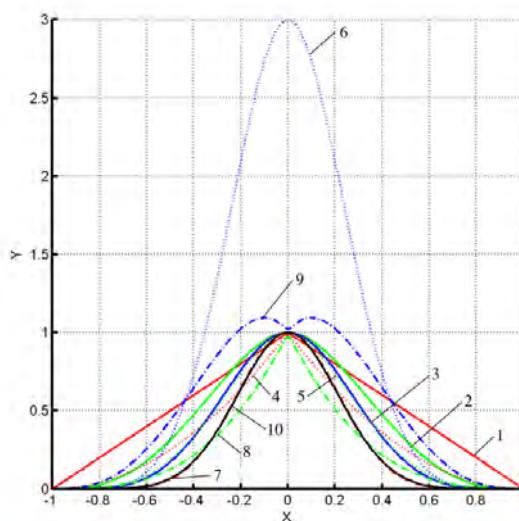
1 omezen na interval $r \in (0,1)$, je nutno všechny vzdálenosti r zvětšit, resp. zmenšit tak, aby
2 v intervalu pro r bylo více zadaných bodů.

3 Nevýhodou je tendence k „vytváření“ lokálního zvlnění vlivem lokálního vlivu funkcí $\phi(r)$.

5 Uveďme základní „globální“ funkce používané při RB interpolacích:

Rostoucí s hodnotou r			
Thin-Plate Spline (TPS)	$r^2 \lg r$ $\lim_{r \rightarrow 0_+} r^2 \lg r = 0$	Multiquadric (MQ)	$\sqrt{1 + \varepsilon r^2}$
Klesající s hodnotou r			
Gauss	$e^{-\varepsilon r^2}$	Inverse multiquadric (IMQ)	$1/\sqrt{1 + \varepsilon r^2}$
Inverse Quadric	$1/(1 + \varepsilon r^2)$		

6 Tab x: Globální RBF funkce



7 Obr.10.4: Geometrické vlastnosti CSRBF

ID	CSRBF	ID	CSRBF
1	$(1 - r)_+$	6	$(1 - r)_+^6(35r^2 + 18r + 3)$
2	$(1 - r)_+^3(3r + 1)$	7	$(1 - r)_+^8(32r^3 + 25r^2 + 8r + 3)$
3	$(1 - r)_+^5(8r^2 + 5r + 1)$	8	$(1 - r)_+^3$
4	$(1 - r)_+^2$	9	$(1 - r)_+^3(5r + 1)$
5	$(1 - r)_+^4(4r + 1)$	10	$(1 - r)_+^7(16r^2 + 7r + 1)$

9 Tab. x: Typické CSRBF funkce

10 V roce 1971 navrhl Hardy [Hardy 1971] interpolaci pomocí RBF s použitím funkce „multiquadric“
11 a metodu nazval metodou radiálních bázových funkcí, která je založena na interpolačním vzorci:

$$f(\mathbf{x}) = \sum_{i=1}^N \lambda_i \phi(r_i) \quad (10.32)$$

12 kde: $\phi(r_i) = \phi(\|\mathbf{x} - \mathbf{x}_i\|)$ a \mathbf{x} je d -dimensionální vektor a λ_i jsou váhy. V roce 1977 Duchon
13 [Duchon 1977] použil funkci typu $\phi(r) = r^2 \lg r$, kterou nazval „Thin-Plate Spline“ (TPS), Shagen
14 [Shagen 1979] v roce 1979 navrhl funkci $\phi(r) = e^{-(\varepsilon r)^2}$ a Wetland [Wetland 2005] v roce 2005
15 navrhl použití CSRBF („compactly supported“) funkci:

$$\phi(r) = \begin{cases} (1-r)^q P(r), & 0 \leq r \leq 1 \\ 0, & r > 1 \end{cases} \quad (10.33)$$

1 kde: $P(r)$ je polynom a q je parametr. Otázky řešitelnosti a stability jsou řešeny např.
2 v [Micchelli 1986] a [Wright 2003]. Wright rozšířil z důvodů stability řešení původní RBF interpolaci
3 polynomem k -tého stupně takto:

$$f(\mathbf{x}) = \sum_{i=1}^N \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) + P_k(\mathbf{x}) = \sum_{i=1}^N \lambda_i \phi_i(\mathbf{x}) + P_k(\mathbf{x}) \quad (10.34)$$

4 a zavedl dodatečné podmínky pro lineární polynom, t.j. $k = 1$:

$$\sum_{i=1}^N \lambda_i = 0 \quad \sum_{i=1}^N \lambda_i \mathbf{x}_i = \mathbf{0} \quad (10.35)$$

5 Obvykle je místo polynomu k -tého stupně $P_k(\mathbf{x})$ použit polynom lineární a to:

$$P_1(\mathbf{x}) = a_0 + \mathbf{a}^T \mathbf{x} \quad (10.36)$$

6 Geometricky vzato, člen a_0 vlastně „nastavuje“ hodnotu pro $\mathbf{x} = \mathbf{0}$ a druhý člen $\mathbf{a}^T \mathbf{x}$, pak vlastně
7 reprezentuje v případě E^3 , tj. pro $\{\mathbf{x}, h\}$, „naklonění“ roviny, v obecném případě „naklonění“
8 nadroviny.

9 Protože hodnoty $h_j = f(\mathbf{x}_j)$ v bodech \mathbf{x}_j jsou známy pro $j = 1, \dots, N$, rovnice tvoří systém lineárních
10 rovnic $\mathbf{Ax} = \mathbf{b}$, jehož řešením dostaneme koeficienty λ_j a a_0, \mathbf{a} , tj.:

$$f(\mathbf{x}_j) = \sum_{i=1}^N \lambda_i \phi(\|\mathbf{x}_j - \mathbf{x}_i\|) + P_k(\mathbf{x}_j) = \sum_{i=1}^N \lambda_i \phi_{i,j} + P_k(\mathbf{x}_j) \quad j = 1, \dots, N \quad (10.37)$$

11 Je tedy zřejmé, že pro d -dimensionální interpolaci a N daných bodů dostáváme soustavu
 $(N + d + 1) \times (N + d + 1)$

12 rovnic.

13 Pro $d = 2$ jsou tedy vektory $\mathbf{x}_i = [x_i, y_i]^T$ a $\mathbf{a} = [a_x, a_y]^T$. Výše uvedenou formuli můžeme rozepsat
14 takto:

$$\left[\begin{array}{cccccc|c} \phi_{1,1} & \dots & \phi_{1,N} & 1 & x_1 & y_1 & \lambda_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{N,1} & \dots & \phi_{N,N} & 1 & x_N & y_N & \lambda_N \\ 1 & \dots & 1 & 0 & 0 & 0 & a_0 \\ x_1 & \dots & x_N & 0 & 0 & 0 & a_x \\ y_1 & \dots & y_N & 0 & 0 & 0 & a_y \end{array} \right] = \left[\begin{array}{c} f_1 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \end{array} \right] \quad (10.38)$$

15 Pak

$$\left[\begin{array}{cc|c} \mathbf{B} & \mathbf{P} & [\lambda] \\ \mathbf{P}^T & \mathbf{0} & [\mathbf{a}] \end{array} \right] = \left[\begin{array}{c} \mathbf{f} \\ \mathbf{0} \end{array} \right] \quad \mathbf{Ax} = \mathbf{b} \quad (10.39)$$

16

$$\mathbf{a}^T \mathbf{x}_i + a_0 = a_0 + a_x x_i + a_y y_i \quad (10.40)$$

17 Pro $d = 3$ jsou tedy vektory $\mathbf{x}_i = [x_i, y_i, z_i]^T$ a $\mathbf{a} = [a_x, a_y, a_z]^T$ dostáváme:

$$\left[\begin{array}{ccccccc|c} \phi_{1,1} & \dots & \phi_{1,N} & 1 & x_1 & y_1 & z_1 & \lambda_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{N,1} & \dots & \phi_{N,N} & 1 & x_N & y_N & z_N & \lambda_N \\ 1 & \dots & 1 & 0 & 0 & 0 & 0 & a_0 \\ x_1 & \dots & x_N & 0 & 0 & 0 & 0 & a_x \\ y_1 & \dots & y_N & 0 & 0 & 0 & 0 & a_y \\ z_1 & \dots & z_N & 0 & 0 & 0 & 0 & a_z \end{array} \right] = \left[\begin{array}{c} f_1 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right] \quad (10.41)$$

18

1 Dostáváme tak opět soustavu rovnic:

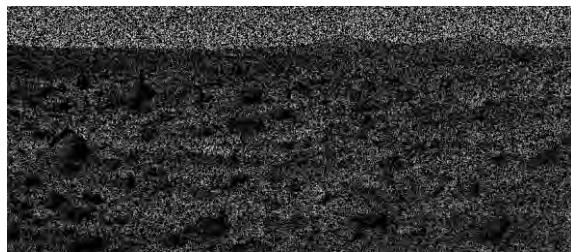
$$\begin{bmatrix} \mathbf{B} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \quad (10.42)$$

2 kde: $\mathbf{a}^T \mathbf{x}_i + a_0 = a_0 + a_x x_i + a_y y_i + a_z z_i$

3

4 Z výše uvedeného je zřejmé, že RBF interpolace je poměrně mocným nástrojem pro řešení
5 interpolačních úloh. RBF interpolací lze využít např.:

- 6 • v interpolacích skalárního 2D/3D pole, kdy z naměřených hodnot v různých neuspořádaných
7 bodech \mathbf{x}_i je nutné určit hodnoty h v jiných bodech \mathbf{x} než naměřených, které se dále
8 zpracovávají.
- 9 • při rekonstrukci poškozených obrazů



Obr.10.5: Původní obraz - 60% poškozených pixelů



Obr.10.6: Rekonstruovaný obraz

- 10 • pro odstranění nápisů na předlohách (inpainting), resp. trhlin na fotografiích a na nástěnných
11 malbách apod.

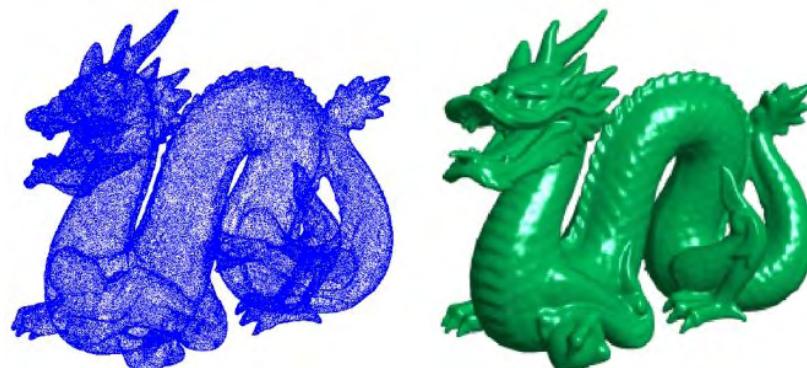


Obr.10.7: Originální obraz
[Bertalmio et al. 2000]



Obr.10.8: Rekonstruovaný obraz
[Uhlir & Skala 2006]

- 12 • při rekonstrukci 3D povrchu z nasnímaných bodů.



Obr.10.9: Rekonstrukce povrchu z nasnímaných bodů (438 000 bodů) [Carr et al. 2001]

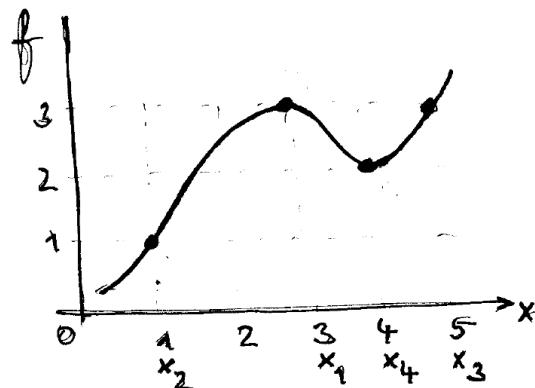
1
2 V případě rekonstrukce povrchu z nasnímaných bodů úloha vede na soustavu homogenních
3 rovnic, tj. $Ax = \mathbf{0}$, tj. nemáme žádnou informaci o orientaci plochy, a proto je nutné použít
4 speciálních přístupů.
5

6 Je zřejmé, že výše uvedenou techniku lze s výhodou použít k „prevzorkování“, např. prevzorkování
7 naměřených dat do strukturované pravoúhlé pravidelné datové struktury („orthogonal mesh“).
8

9 Až dosud jsme se zabývali interpolací neuspořádaných dat, tj. případem, kdy interpolovaná křivka či
10 plocha prochází danými body. V mnoha případech jsou vzorky interpolované veličiny příliš „husté“,
11 případně nepožadujeme naprostou přesnost, ale chceme snížit počet parametrů RBF. V tomto
12 případě můžeme použít *RBF approximace*.
13

Příklad

15 Pro jednoduchou demonstraci aplikace RBF interpolace uvažme následující příklad, který je pouze
16 jednorozměrným případem, tj. $y = f(x)$. Uvažme naměřené body, viz Obr.10.10, a nalezněme RBF
17 Interpolaci.



Obr.10.10: RBF interpolace pro 1&1/2 D - jednoduché zadání

21 Pro názornost použijme funkci $\phi(r_{ij}) = \phi(\|x_j - x_i\|) = |x_j - x_i|$. Pak dostáváme soustavu rovnic:

$$\begin{bmatrix} 0 & 2 & 2 & 1 & 1 & 3 \\ 2 & 0 & 4 & 3 & 1 & 1 \\ 2 & 4 & 0 & 1 & 1 & 5 \\ 1 & 3 & 1 & 0 & 1 & 4 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & 5 & 4 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ a_0 \\ a_x \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 3 \\ 2 \\ 0 \\ 0 \end{bmatrix} \quad (10.43)$$

22 Jejím řešením dostáváme RBF interpolaci, tj. hodnoty $\lambda_1, \dots, \lambda_4, a_0$ a a_x . Je nutné zdůraznit, že matice
23 soustavy se změní pouze tehdy, pokud se změní pozice bodů x_i .
24

10.4. Aproximace

2 Aproximace na rozdíl od interpolací prokládají danými daty křivku, která obecně neprochází
 3 zadanými body. Aproximace dat se používá při řešení mnoha problémů, kdy jsou dány vzorky dat a je
 4 nutné je přiblížně reprezentovat nějakým funkčním popisem.

5 V technické praxi se velmi často používá metoda nejmenších čtverců (Least Square Error - LSE) pro
 6 nalezení „optimálního“ řešení. Při její aplikaci je však nutné mít na paměti, že „standardní“ metoda
 7 nejmenších čtverců může poskytovat neadekvátní výsledky, neboť *chyba se měří „na svislici“ a vliv*
 8 *odchylky roste s kvadrátem*. Lepší výsledky dává ortogonální metoda, která měří vzdálenosti „na
 9 kolmici“, nicméně je podstatně výpočetně náročnější.

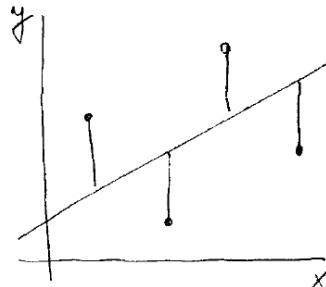
10 „Standardní“ metoda nejmenších čtverců je použitelná na situace, kdy výsledná approximace je
 11 explicitního charakteru, tj.:

$$y = f(x_1, \dots, x_n) = f(\mathbf{x}) \quad (10.44)$$

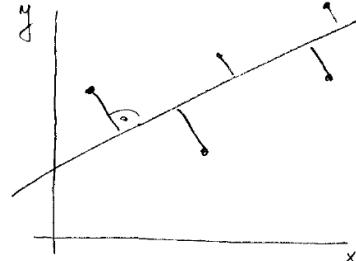
12 V případě, že výsledná approximace je implicitního charakteru, tj.:

$$F(x_1, \dots, x_n) = F(\mathbf{x}) = 0 \quad (10.45)$$

13 nelze výše uvedený postup aplikovat a musí se použít jiné postupy.



Obr.10.11: Standardní lineární regrese



Obr.10.12: Ortogonální lineární regrese

14
 15 Rozdíl mezi zadanou hodnotou a hodnotou approximovanou je pak chyba (Error), kterou se snažíme
 16 minimalizovat podle nějakého kritéria. Průměrnou chybu approximace lze pak definovat takto:

$$\varepsilon = \frac{1}{n} \sum_{i=1}^n |f(\mathbf{x}_i) - h_i| \quad (10.46)$$

17 kde: h_i jsou zadané hodnoty v bodech \mathbf{x}_i , $f(\mathbf{x}_i)$ jsou hodnoty v daných bodech vypočtené a n je
 18 počet zadaných bodů. Hodnota ε je hodnota určující průměrnou chybu na jeden bod, tj. jde o
 19 relativizující kritérium vůči počtu bodů.

20 Je nutné upozornit, že velmi často používaná metoda nejmenších čtverců nemusí být vhodná, neboť
 21 *penalizační váha odchylky roste s kvadrátem odchylky*. Toto může být problém při prokládání daných
 22 bodů přímkou v E^2 , resp. prokládání rovinou v případě E^3 , kdy potřebujeme minimální odchylku
 23 vzdáleností, nikoliv její kvadrát.

$$\min_{a, b} \left\{ \sum_{i=1}^n (f(\mathbf{x}_i) - h_i)^2 \right\}$$

Metoda nejmenších čtverců

$$\min_{a, b} \left\{ \sum_{i=1}^n \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}} \right\}$$

Minimalizace vzdáleností

$$\min_{a, b} \left\{ \sum_{i=1}^n \left(\frac{ax_i + by_i + c}{\sqrt{a^2 + b^2}} \right)^2 \right\}$$

Minimalizace čtverců vzdáleností

10.5. Metoda nejmenších čtverců

Asi nejčastěji používanou approximací je metoda nejmenších čtverců. Nechť jsou dány skalární hodnoty h_i v bodech \mathbf{x}_i , a chtejme těmito body proložit křivku $\phi(\mathbf{x})$, která je tvaru:

$$\phi(\mathbf{x}) = \xi_1 \varphi_1(\mathbf{x}) + \dots + \xi_p \varphi_p(\mathbf{x}) = [\xi_1, \dots, \xi_p] [\varphi_1(\mathbf{x}), \dots, \varphi_p(\mathbf{x})]^T \quad (10.47)$$

přičemž funkce $\varphi_i(\mathbf{x})$ nejsou závislé na žádném parametru a $n > p$. Hledáme takové koeficienty ξ_i , které minimalizují námi zadané kritérium, v našem případě použijeme nejmenší čtverce.

Lze nahlédnout, že pro jednotlivé body lze v maticové formě psát:

$$\begin{bmatrix} \varphi_1(\mathbf{x}_1) & \cdots & \varphi_p(\mathbf{x}_1) \\ \varphi_1(\mathbf{x}_2) & \cdots & \varphi_p(\mathbf{x}_2) \\ \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{x}_{n-1}) & \cdots & \varphi_p(\mathbf{x}_{n-1}) \\ \varphi_1(\mathbf{x}_n) & \cdots & \varphi_p(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_p \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n-1} \\ h_n \end{bmatrix} \quad (10.48)$$

Dostáváme přeurovenou soustavu rovnic $\mathbf{Ax} = \mathbf{b}$ a potřebujeme určit hodnoty vektoru \mathbf{x} . Nyní můžeme definovat chybu r a její kvadrát r^2 takto:

$$r = \|\mathbf{Ax} - \mathbf{b}\| \quad r^2 = (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) \quad (10.49)$$

Pak:

$$r^2 = (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) = (\mathbf{Ax})^T (\mathbf{Ax}) - (\mathbf{Ax})^T \mathbf{b} - \mathbf{b}^T \mathbf{Ax} + \mathbf{b}^T \mathbf{b} \quad (10.50)$$

a protože platí:

$$\mathbf{b}^T \mathbf{Ax} = (\mathbf{Ax})^T \mathbf{b} \quad (10.51)$$

můžeme tedy psát:

$$\begin{aligned} r^2 &= (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) = (\mathbf{Ax})^T (\mathbf{Ax}) - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b} \\ &= \mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b} \end{aligned} \quad (10.52)$$

Pro extrém musí platit podmínka:

$$\frac{\partial r^2}{\partial \mathbf{x}} = 2\mathbf{A}^T \mathbf{Ax} - 2\mathbf{A}^T \mathbf{b} = 0 \quad (10.53)$$

takže dostáváme soustavu lineárních rovnic:

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b} \quad (10.54)$$

Matice $\mathbf{A}^T \mathbf{A}$ je symetrická pozitivně semidefinitní. Ve většině reálných případů je pozitivně definitní.

Podrobnější informace o aplikaci metody nejmenších čtverců lze nalézt např. v:

- Schneider,P.J., Eberly,D.: Geometric Tools for Computer Graphics, Morgan Kaufmann Publ., 2003
- Eberly,D.: Least Squares Fitting of Data, Geometric Tools, 2008 ([CLICK off-line](#))
- Eberly,D.: Least Squares Fitting of Data with B-Spline Curves, Geometric Tools, 2008 ([CLICK off-line](#))
- Eberly,D.: Least Squares Fitting of Data with a Cylinder, Geometric Tools, 2008 ([CLICK off-line](#))

1 **Příklad**

2 Pro jednoduchost uvažme body reprezentující povrch koule získaných, např. odměřováním,
 3 skenováním apod. Je tedy dána množina bodů $\{x_i\}_{i=1}^n$ reprezentující povrch koule a počet bodů
 4 $n > 4$. Body na povrchu koule musí vyhovovat rovnici:

$$x^2 + y^2 + z^2 + ax + by + cz + d = 0 \quad (10.55)$$

5 Pro jednotlivé body x_i by měla platit tato rovnice, tj.:

$$x_i^2 + y_i^2 + z_i^2 + ax_i + by_i + cz_i + d = 0 \quad i = 1, \dots, n \quad (10.56)$$

6 Je zřejmé, že žádné měření není absolutně přesné a tedy rovnice zřejmě nebude splněna ani pro
 7 jeden daný bod. Úkolem je však nalézt parametry pro kouli „nejlépe“ approximující naměřený povrch.

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & \ddots & & 1 \\ & \ddots & & 1 \\ x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = - \begin{bmatrix} x_1^2 + y_1^2 + z_1^2 \\ x_2^2 + y_2^2 + z_2^2 \\ \vdots \\ x_n^2 + y_n^2 + z_n^2 \end{bmatrix} \quad (10.57)$$

8 Dostáváme opět přeurovenou soustavu rovnic $\mathbf{Ax} = \mathbf{b}$, kterou můžeme řešit různými numerickými
 9 postupy, např. metodou nejmenších čtverců, nebo dekompozicí (Singular Value Decomposition -
 10 SVD) apod.

11 Je nutné zdůraznit, že stabilita řešení prudce klesá s rozdílem výsledné matice $\mathbf{A}^T \mathbf{A}$.

14 **Příklad**

15 V mnoha případech potřebujeme posoudit chování algoritmu, tj. časovou nebo paměťovou složitost
 16 na základě experimentálních měření. Uvažme algoritmus, který zpracovává „typická data“ pro různý
 17 počet zpracovávaných dat. To znamená, že z experimentů dostáváme množinu $\Omega = \{\langle n_i, t_i \rangle\}_{i=1}^N$, kde:
 18 n_i je počet zpracovávaných primitiv, t_i je spotřebovaný výpočetní čas, N je celkový počet
 19 experimentů.

21 Úkolem je odhadnout výpočetní složitost algoritmu.

23 Předpokládejme, že základní uvažované složitosti jednotlivých částí algoritmu jsou např.:

$$O(\lg n), O(\sqrt{n}), O(n), O(n \lg n), \dots, O(n^2), \dots$$

25 Pak tyto funkce jsou vlastně funkce $\varphi_i(n)$, viz předchozí text, tj.:

$$\phi(n) = \xi_1 \varphi_1(n) + \dots + \xi_p \varphi_p(n) \quad (10.58)$$

26 Protože každý algoritmus má fixní počáteční složitost $O(1)$, položíme $\varphi_1(n) = 1$

27 Také dostáváme:

$$\varphi_1(n) = 1 \quad \varphi_2(n) = \lg n \quad \varphi_3(n) = \sqrt{n} \quad \varphi_4(n) = n \lg n \quad \dots \dots \dots \quad \varphi_p(n) = n^2$$

28 přičemž $p \ll N$.

30 Dostáváme tedy opět přeurovenou soustavu rovnic $\mathbf{Ax} = \mathbf{b}$, kterou vyřešíme dříve uvedenými
 31 způsoby.

32

10.6. Výpočet vzdálenosti bodu od trojúhelníka v E^3

2 Vzdálenost bodu od přímky, resp. roviny je dána:

$$dist = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}} \quad dist = \frac{|ax + by + cz + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (10.59)$$

3 nicméně v mnoha aplikacích potřebujeme např. vzdálenost bodu od trojúhelníka v E^2 nebo v E^3 .

4 Uvažme trojúhelník v E^3 a bod a úkolem je najít vzdálenost daného bodu od tohoto trojúhelníka, tento problém elegantně vyřešil Eberley v roce 1999 a je dobrým návodem, jak v obdobných případech postupovat.

5 Je dán bod \mathbf{P} a trojúhelník daný parametricky $\mathbf{T}(s, t) = \mathbf{B} + s\mathbf{E}_0 + t\mathbf{E}_1$, přičemž parametry $(s, t) \in D$, kde D je oblast $s, t \in \langle 0, 1 \rangle$ a $s + t \leq 1$.

6 Kvadrát vzdálenosti $Q(s, t)$ bodu od plochy, na které trojúhelník leží, je roven:

$$7 Q(s, t) = (\mathbf{T}(s, t) - \mathbf{P})^2 \quad (10.60)$$

8 což je rovnice kvadratické formy pro s, t , tj.

$$9 Q(s, t) = as^2 + 2bst + ct^2 + 2ds + 2et + f \quad (10.61)$$

10 kde: $a = \mathbf{E}_0 \cdot \mathbf{E}_0$, $b = \mathbf{E}_0 \cdot \mathbf{E}_1$, $c = \mathbf{E}_1 \cdot \mathbf{E}_1$, $d = \mathbf{E}_0 \cdot (\mathbf{B} - \mathbf{P})$,

11 $e = \mathbf{E}_0 \cdot (\mathbf{B} - \mathbf{P})$ a $f = (\mathbf{B} - \mathbf{P}) \cdot (\mathbf{B} - \mathbf{P})$

12 Kvadratické formy jsou klasifikovány znaménkem výrazu $ac - b^2$. Pro funkci $Q(s, t)$ dostaváme:

$$13 ac - b^2 = (\mathbf{E}_0 \cdot \mathbf{E}_0)(\mathbf{E}_1 \cdot \mathbf{E}_1) - (\mathbf{E}_0 \cdot \mathbf{E}_1)^2 = |\mathbf{E}_0 \times \mathbf{E}_1|^2 > 0 \quad (10.62)$$

14 Za předpokladu, že vektory \mathbf{E}_0 a \mathbf{E}_1 jsou nezávislé a nejsou na sebe kolmé. Pro minimum musí platit:

$$15 \nabla Q = \left[\frac{\partial Q}{\partial s}, \frac{\partial Q}{\partial t} \right]^T = [2(as + bt + d), 2(bs + ct + e)]^T = [0, 0]^T \quad (10.63)$$

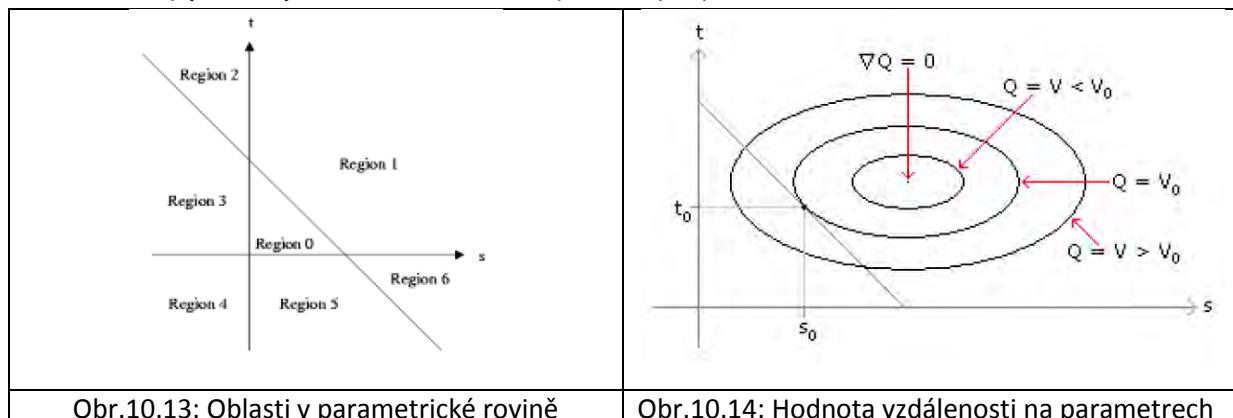
16 Dostaváme tedy soustavu lineárních rovnic:

$$17 \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} -d \\ -e \end{bmatrix} \quad (10.64)$$

18 a jejím řešením pak:

$$19 s = \frac{(be - cd)}{(ac - b^2)} / (ac - b^2) \quad t = \frac{(bd - ae)}{(ac - b^2)} / (ac - b^2) \quad (10.65)$$

20 Pokud $(s, t) \in D$ pak nejbližší bod na rovině je bod trojúhelníka, v opačném případě nejbližší bod leží na nějaké hraně trojúhelníka. Podle hodnoty parametrů (s, t) určíme, v které oblasti takový minimální bod leží. Např. pro oblast Region₁ hledáme minimum funkce $Q(s, t)$ pro $s + t = 1$, tedy dostaváme $Q(s, 1 - s)$ a hledáme minimum pro $s \in \langle 0, 1 \rangle$.



21 Podrobný popis lze nalézt např. v:

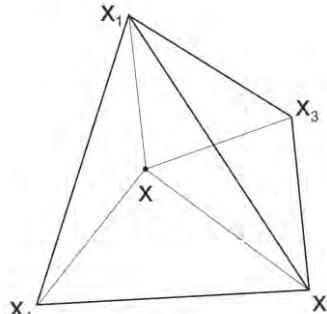
- 22 • Eberly,D.: Distance Between Point and Triangle in 3D, <http://www.geometrictools.com>, 1999

10.7. Metody výpočtu průsečíků

Výpočty průsečíků s elementárními objekty jako je trojúhelník, kružnice, resp. koule atd. jsou často používány nejen v počítačové grafice. V následujícím si uvedeme základní přístupy řešení vybraných problémů.

10.7.1. Výpočet průsečíku přímky a trojúhelníka v E^3

Výpočet průsečíku přímky v parametrickém tvaru s trojúhelníkem v obecné poloze v E^3 . Jsou dány souřadnice vrcholů $\mathbf{X}_q = [X_q, Y_q, Z_q]^T$ v Eukleidovském prostoru.



Obr.10.15: Čtyřstěn a bod

Pak:

$$\begin{aligned}\mathbf{X}(u, v) &= \mathbf{X}_1 + (\mathbf{X}_2 - \mathbf{X}_1) u + (\mathbf{X}_3 - \mathbf{X}_1) v = \mathbf{X}_1 + \mathbf{S}_1 u + \mathbf{S}_2 v \\ \mathbf{X}(t) &= \mathbf{X}_A + (\mathbf{X}_B - \mathbf{X}_A) t = \mathbf{X}_A + \mathbf{S} t\end{aligned}\quad (10.66)$$

Pro průsečík přímky s plochou musí platit:

$$\mathbf{X}(u, v) = \mathbf{X}(t) \quad (10.67)$$

a tedy:

$$\mathbf{X}_1 + \mathbf{S}_1 u + \mathbf{S}_2 v = \mathbf{X}_A + \mathbf{S} t \quad (10.68)$$

Jednoduchou úpravou pak dostáváme:

$$\mathbf{S}_1 u + \mathbf{S}_2 v - \mathbf{S} t = \mathbf{X}_A - \mathbf{X}_1 \quad (10.69)$$

Jde tedy o řešení soustavy lineárních rovnic s pravou stranou. Pro průsečík musí platit podmínka $u, v \in \langle 0,1 \rangle$ & $0 \leq u + v \leq 1$.

Úloha

Nakreslete si relevantní obrázek a odpovězte otázky:

- jaké bude řešení při řešení soustavy pomocí zobecněného vektorového součinu a jak bude vypadat programová sekvence pro GPU?
- jak bude vypadat řešení, pokud souřadnice vrcholů budou v homogenních souřadnicích, tj. obecně $w \neq 1$? Lze použít lineární interpolaci s nelineární parametrizací?

Předpokládejme opět:

$$\begin{aligned}\mathbf{x}(u, v) &= \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1) u + (\mathbf{x}_3 - \mathbf{x}_1) v = \mathbf{x}_1 + \mathbf{s}_1 u + \mathbf{s}_2 v \\ \mathbf{x}(t) &= \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A) t = \mathbf{x}_A + \mathbf{s} t\end{aligned}\quad (10.70)$$

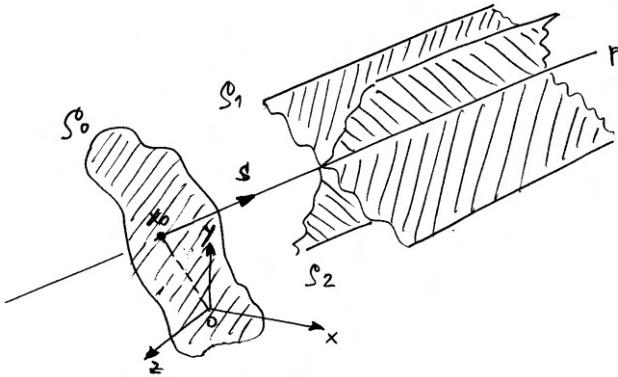
avšak body jsou nyní obecně v projektivním prostoru, tj. $\mathbf{x}_q = [x_q, y_q, z_q : w]^T$.

Úloha

Navrhněte algoritmus, který bude počítat průsečík přímky se čtyřstěnem.

1 **10.7.2. Určení přímky jako průsečnice dvou rovin**

2 V mnoha geometrických úlohách je nutné určit přímku jako průsečnici dvou rovin v E^3 . Tato zdánlivě
3 jednoduchá úloha je obvykle řešena v Eukleidovském prostoru.



Obr.10.16: Přímka jako průsečnice dvou rovin

Jsou dány dvě roviny ρ_1 a ρ_2 :

$$\rho_1 = [a_1, b_1, c_1; d_1]^T = [\mathbf{n}_1^T; d_1]^T \quad \rho_2 = [a_2, b_2, c_2; d_2]^T = [\mathbf{n}_2^T; d_2]^T \quad (10.71)$$

kde \mathbf{n}_1 a \mathbf{n}_2 jsou normály těchto rovin.

Pak směrový vektor přímky v parametrickém tvaru $\mathbf{x}(t) = \mathbf{x}_0 + st$ je určen vektorovým součinem:

$$\mathbf{s} = \mathbf{n}_1 \times \mathbf{n}_2 \equiv [a_3, b_3, c_3]^T$$

a bod \mathbf{x}_0 přímky je určen vztahy:

$$\begin{aligned} x_0 &= \frac{d_2 \begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix} - d_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix}}{DET} & y_0 &= \frac{d_2 \begin{vmatrix} a_3 & c_3 \\ a_1 & c_1 \end{vmatrix} - d_1 \begin{vmatrix} a_3 & c_3 \\ a_2 & c_2 \end{vmatrix}}{DET} \\ z_0 &= \frac{d_2 \begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} - d_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}}{DET} & DET &= \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \end{aligned} \quad (10.72)$$

tato formule je nejen velmi obtížně zapamatovatelná, ale zcela „průzračné“, jak se odvodí a ani není zřejmé, jaké vlastnosti bod \mathbf{x}_0 má. Je vhodné upozornit, na možné problémy se stabilitou, pokud $|DET| < \epsilon$. Uvedená formule není vhodná pro aplikace GPU a na architekturách podporující vektorové operace.

Alternativní řešení pomocí Plückerových souřadnic, které je výpočetně složitější, lze nalézt v:

- Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int.Journal on Image and Graphics, DOI No: [10.1142/S021946780800326X](https://doi.org/10.1142/S021946780800326X), ISSN 0219-4678, Vol.8, No.4, pp.615-628, 2008

Pokud je úloha řešena v projektivní reprezentaci, pak dostaneme jednoduchý algoritmus, který je také vhodný pro GPU aplikace. Uvažme opět roviny ρ_1 a ρ_2 . Pak směrový vektor přímky je určen opět vektorovým součinem normál, a to:

$$\mathbf{s} = \mathbf{n}_1 \times \mathbf{n}_2 \quad (10.73)$$

Až dosud je postup stejný, ale otázkou je jak určit bod \mathbf{x}_0 .

1 Uvažme rovinu ρ_0 procházející počátkem, jejíž normála \mathbf{n}_0 je stejná jako již určený směrový vektor \mathbf{s}
 2 přímky, viz Obr.10.17.

$$\rho_0 = [a_0, b_0, c_0; 0]^T = [\mathbf{s}^T; 0]^T \quad (10.74)$$

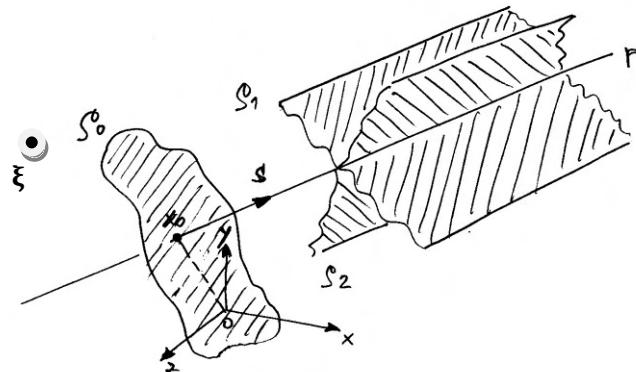
3 Pak bod x_0 je jednodušš určen jako průsečík 3 rovin, a to:

$$x_0 = \rho_1 \times \rho_2 \times \rho_0 \quad (10.75)$$

4 Z uvedeného je zřejmé, že daný postup je elegantní, jednoduchý a je vhodný pro aplikaci na GPU apod.
 5

6 10.7.3. Určení nejbližšího bodu na průsečnici dvou rovin

7 Úlohou je nalezení k danému bodu ξ nejbližší bod na průsečnici dvou rovin ρ_1 a ρ_2 , viz Obr.10.17



Obr.10.17: Nejbližší bod na průsečnici dvou rovin k danému bodu

Řešení této úlohy v Eukleidovském prostoru lze nalézt např. v:

- Krumm,J.: Intersection of Two Planes, Microsoft Research, May 2000
<http://research.microsoft.com/apps/pubs/default.aspx?id=68640> (CLICK off-line)

které je založeno na řešení soustavy (5×5) rovnic s Lagrangeovými multiplikátory:

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 2\xi_x \\ 2\xi_y \\ 2\xi_z \\ \mathbf{p}_1 \mathbf{n}_1 \\ \mathbf{p}_2 \mathbf{n}_2 \end{bmatrix} \quad (10.76)$$

kde: \mathbf{p}_1 , resp. \mathbf{p}_2 jsou body na rovině ρ_1 , resp. ρ_2 , s normálou \mathbf{n}_1 , resp. \mathbf{n}_2 . Souřadnice bodu $\mathbf{x} = [x, y, z]^T$, který je na průsečnici dvou rovin a je nejbliže bodu ξ , je pak určen řešením dané soustavy rovnic.

Uvedeme opět elegantní řešení v projektivní reprezentaci. Řešení má následující kroky:

- posunutí rovin ρ_1 , ρ_2 a bodu ξ tak, aby bod ξ byl v počátku, tj. transformace posunu s maticí T .
- výpočet průsečnice rovin ρ_1 a ρ_2 - výsledkem je přímka p daná směrovým vektorem s a bodem x_0
- posunutí bodu x_0 inverzní operací posuvu T^{-1}

Z uvedeného je zřejmé, že řešení daného problému je nejen jednodušší a rychlejší, ale je vhodné pro aplikace na GPU apod.

1 **10.7.4. Výpočet průsečíku přímky a koule v E^3**

2 Koule je nejen velmi častým používaným primitivem, ale používá se také jako ohraničující těleso pro
3 urychlování např. algoritmů pro nalezení kolize, detekce existence průsečíku se složitějšími objekty
4 atd.

6 Předpokládejme, že máme přímku p v Eukleidovském prostoru E^3 danou rovnicí:

$$\mathbf{x}(t) = \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A)t = \mathbf{x}_A + \mathbf{s}t \quad (10.77)$$

7 a povrch koule je dán vztahem:

$$(\mathbf{x} - \mathbf{x}_s)^T(\mathbf{x} - \mathbf{x}_s) - r^2 = 0 \quad (10.78)$$

9 Dosazením výrazu pro přímku do rovnice pro kouli pak dostaváme:

$$(\mathbf{x}_A + \mathbf{s}t - \mathbf{x}_s)^T(\mathbf{x}_A + \mathbf{s}t - \mathbf{x}_s) - r^2 = 0 \quad (10.79)$$

10 a tedy:

$$(\mathbf{s}t + \xi)^T(\mathbf{s}t + \xi) - r^2 = 0 \quad \xi = \mathbf{x}_A - \mathbf{x}_s \quad (10.80)$$

11 tedy kvadratickou rovnici.

$$\mathbf{s}^T \mathbf{s} t^2 + 2\mathbf{s}^T \xi t + \xi^T \xi - r^2 = 0 \quad (10.81)$$

13 Jejím řešením pak dostaváme:

$$at^2 + bt + c = 0 \quad t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (10.82)$$

14 Je zřejmé, že pokud $b^2 - 4ac < 0$, pak průsečík a ani dotyk přímky s koulí neexistuje. Pokud se
15 počítá větší počet průsečíků se stejnou přímkou, je výpočetně výhodné vektor \mathbf{s} normalizovat, tj.
16 $\|\mathbf{s}\| = 1$, čímž se výpočet zjednoduší.

18 **Úloha**

19 Modifikujte výše uvedený algoritmus pro případ:

- 20 • průsečíku přímky s elipsoidem
- 21 • když body \mathbf{x}_A a \mathbf{x}_B jsou zadány v projektivních souřadnicích
- 22 • pokuste se najít řešení pro případ, že body \mathbf{x}_A , \mathbf{x}_B a \mathbf{x}_s jsou zadány v projektivních
23 souřadnicích

24 a v odborné literatuře najděte efektivní algoritmy pro řešení průsečíků – viz např. Graphics GEMS.

25

10.7.5. Výpočet obalové koule opsané trojúhelníku v E^3

V některých aplikacích je vhodné určit kouli opsanou danému trojúhelníku (circumcircle of a triangle) v, přičemž trojúhelník se nalézá v obecné poloze. Níže uvedený postup je aplikovatelný obecně pro d -dimensionální případ. Nechť $\mathbf{A}, \mathbf{B}, \mathbf{C}$ jsou vrcholy daného trojúhelníka. Pak nechť:

$$a = A - C \quad b = B - C \quad (10.83)$$

Pak polomér koule r je pro případ E^3 určen:

$$r = \frac{\|a\|\|b\|\|a - b\|}{2\|a \times b\|} = \frac{\|a - b\|}{2 \sin \theta} \quad (10.84)$$

kde θ je úhel sevřený vektory a a b . Výše uvedená formule platí pouze pro případ E^3 .

Střed koule je pak určen vztahem:

$$x_0 = \frac{(\|a\|^2 b - \|b\|^2 a) \times (a \times b)}{2\|a \times b\|^2} + c \quad (10.85)$$

V případě obecném je pak nutné použít následujících identit:

$$\begin{aligned} (\mathbf{a} \times \mathbf{b}) \times \mathbf{c} &= (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{b} \cdot \mathbf{c})\mathbf{a} \\ \mathbf{a} \times (\mathbf{b} \times \mathbf{c}) &= (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c} \\ \|\mathbf{a} \times \mathbf{b}\|^2 &= \mathbf{a}^2\mathbf{b}^2 - (\mathbf{a} \cdot \mathbf{b})^2 \end{aligned} \quad (10.86)$$

Je tedy zřejmé, že výpočet je poněkud komplikovanější.

10.7.6. Výpočet průsečíku přímky a kvadratické plochy v E^3

2 Výpočet přímky s koulí je poměrně jednoduchý. V praxi se však často vyskytují kvadratické plochy,
3 které jsou dány rovnicí:

$$x^T Q x = 0 \quad (10.87)$$

4 kde: $x = [x, y, z; 1]^T$. Tato formulace zahrnuje kouli, elipsoid, válec, hyperboloid, paraboloid,
5 kuželovou plochu, válcovou plochu, hyperbolický paraboloid atd., včetně roviny. Matice kvadratické
6 formy Q (4×4) je obecně *symetrická*, resp. je možné případně nesymetrickou matici na symetrickou
7 převést. Jde tedy o obecnější formulaci a rovnice (10.87) vlastně popisuje plochu danou rovnicí:

$$x^T Q x = a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44} = 0 \quad (10.88)$$

8 To znamená, že jedna rovnice obecně popisuje více typů ploch, které nazýváme kvadrikami a jejich
9 klasifikaci lze nalézt např. v:

- 10 • Bartsch,H.-J.: Matematické vzorce – kvadratické plochy, SNTL, pp.324-334, 1971
- 11 • Rektorys,K. a kol.: Přehled užité matematiky – kvadratické plochy, SNTL, pp.206-218, 1981

12 Podívejme se nyní na výpočet průsečíku kvadriky s přímkou:

$$x(t) = x_A + s t$$

13 kde: $s = [s_x, s_y, s_z; 0]^T$ a $x_A = [x_A, y_A, z_A; 1]^T$.

14 Dosazením $x(t)$ do rovnice pro kvadriku (11.86) dostaváme (matice Q je symetrická):

$$(x_A + s t)^T Q (x_A + s t) = 0 \quad (10.89)$$

15 Lze tedy psát:

$$s^T Q s t^2 + 2s^T Q x_A t + x_A^T Q x_A = 0 \quad (10.90)$$

16 a dostaváme opět kvadratickou rovnici a jejím řešením dostaneme průsečíky:

$$at^2 + 2bt + c = 0 \quad t_{1,2} = \frac{-b \pm \sqrt{b^2 - ac}}{a} \quad (10.91)$$

17 kde: $a = s^T Q s$, $b = s^T Q x_A$ a $c = x_A^T Q x_A$.

18 Je zřejmé, že pokud diskriminant $D = b^2 - ac < 0$, pak průsečík a ani dotyk přímky s kvadrikou
19 neexistuje. Je nutné zdůraznit, že výše uvedeným postupem dokážeme reprezentovat i plochy, které
20 jsou nekonečné, např. paraboloid.

10.7.7. Detekce existence průsečíku s kvadratickou plochou

21 V mnoha aplikacích se používá test existence průsečíku přímky (resp. paprsku) s kvadratickou
22 plochou, resp. koulí. Pro existenci průsečíku s kvadratickou plochou musí být diskriminant D
23 pozitivně semidefinitní, tj. $D \geq 0$.

24 Z rovnice (10.90) lze pro diskriminant D lze psát:

$$D = (s^T Q x_A) (s^T Q x_A) - (s^T Q s) (x_A^T Q x_A) = s^T Q^T [x_A s^T - s x_A^T] Q x_A \\ = s^T Q^T [x_A \otimes s - s \otimes x_A] Q x_A = s^T Q^T R Q x_A \quad (10.92)$$

25 kde matice Q je symetrická, matice R je antisymetrická s nulovou diagonálou.

$$R = \begin{bmatrix} 0 & x_A s_y - y_A s_x & x_A s_z - z_A s_x & -s_x \\ -(x_A s_y - y_A s_x) & 0 & y_A s_z - z_A s_y & -s_y \\ -(x_A s_z - z_A s_x) & -(y_A s_z - z_A s_y) & 0 & -s_z \\ s_x & s_y & s_z & 0 \end{bmatrix} \quad (10.93)$$

26 Je zřejmé, že matice R reprezentuje vlastnosti přímky, resp. paprsku.

1 Pro $\sigma = \mathbf{x}_B - \mathbf{x}_A$ lze ukázat, že platí:

$$\mathbf{R} = \begin{bmatrix} 0 & x_A y_B - x_B y_A & x_A z_B - x_B z_A & x_A - x_B \\ -(x_A y_B - x_B y_A) & 0 & y_A z_B - y_B z_A & y_A - y_B \\ -(x_A z_B - x_B z_A) & -(y_A z_B - y_B z_A) & 0 & z_A - z_B \\ -(x_A - x_B) & -(y_A - y_B) & -(z_A - z_B) & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{B} & -\sigma \\ \sigma^T & 0 \end{bmatrix} \quad (10.94)$$

2 Vektorový součin $\mathbf{w} \times \mathbf{v}$ je možné zapsat i v maticové formě:

$$\mathbf{w} \times \mathbf{v} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} \mathbf{v} \quad (10.95)$$

3 Je tedy zřejmé, že submatice matice \mathbf{B} je vlastně maticovým vyjádřením pro vektorový součin, tj.:

$$\mathbf{B}\mathbf{q} = \begin{bmatrix} 0 & x_A s_y - y_A s_x & x_A s_z - z_A s_x \\ -(x_A s_y - y_A s_x) & 0 & y_A s_z - z_A s_y \\ -(x_A s_z - z_A s_x) & -(y_A s_z - z_A s_y) & 0 \end{bmatrix} \mathbf{q} = [\sigma \times \xi_A] \times \mathbf{q} \quad (10.96)$$

4 kde: $\mathbf{s} = [s_x, s_y, s_z; 0]^T = [\sigma^T; 0]^T$ a $\mathbf{x}_A = [x_A, y_A, z_A; 1]^T = [\xi_A^T; 1]^T$.

5 To znamená, že prvky matice \mathbf{B} jsou vlastně dány vektorovým součinem $\sigma \times \xi_A$.

6 Rovnici (10.92) lze tedy přepsat do tvaru vhodného pro SSE instrukce nebo GPU implementaci takto:

$$D = \mathbf{s}^T \mathbf{Q}^T [\mathbf{x}_A \otimes \mathbf{s} - \mathbf{s} \otimes \mathbf{x}_A] \mathbf{Q} \mathbf{x}_A = \mathbf{s}^T \mathbf{Q}^T \mathbf{R} \mathbf{Q} \mathbf{x}_A \quad (10.97)$$

7
8 Je zřejmé, že pro specifické kvadratické plochy lze výše uvedenou formuli zdognodušit. Nicméně při
9 použití SSE instrukcí, nebo implementaci na GPU toto nemusí vést ke zrychlení výpočtu.

10
11 **Zobecnění pro zadání v projektivním prostoru**

12 Z výše uvedené formulace je zřejmé, že pro detekci existence průsečíku lze s výhodou použít i lineární
13 interpolaci na přímce s *monotonní parametrizací*, pokud body \mathbf{x}_A a \mathbf{x}_B jsou dány v projektivních
14 souřadnicích, tj. $\mathbf{x}_A = [x_A, y_A, z_A; w_A]^T$ a $\mathbf{x}_B = [x_B, y_B, z_B; w_B]^T$. Pak:

$$\mathbf{R} = \begin{bmatrix} 0 & x_A s_y - y_A s_x & x_A s_z - z_A s_x & -s_x w_A \\ -(x_A s_y - y_A s_x) & 0 & y_A s_z - z_A s_y & -s_y w_A \\ -(x_A s_z - z_A s_x) & -(y_A s_z - z_A s_y) & 0 & -s_z w_A \\ s_x w_A & s_y w_A & s_z w_A & 0 \end{bmatrix} \quad (10.98)$$

15 Po dosazení za vektor \mathbf{s} dostáváme:

$$\mathbf{R} = \begin{bmatrix} 0 & x_A y_B - x_B y_A & x_A z_B - z_A x_B & x_A w_B - x_B w_A \\ -(x_A y_B - x_B y_A) & 0 & y_A z_B - z_A y_B & y_A w_B - y_B w_A \\ -(x_A z_B - z_A x_B) & -(y_A z_B - z_A y_B) & 0 & z_A w_B - z_B w_A \\ -(x_A w_B - x_B w_A) & -(y_A w_B - y_B w_A) & -(z_A w_B - z_B w_A) & 0 \end{bmatrix} \quad (10.99)$$

16 Pokud matice \mathbf{M} je dána:

$$\mathbf{M} = \begin{bmatrix} x_A & y_A & z_A & w_A \\ x_B & y_B & z_B & w_B \end{bmatrix} \quad (10.100)$$

17 pak je zřejmé, že prvky matice \mathbf{R} jsou určeny determinanty submatic $M_{ij} = \det[\mathbf{m}_{i*} | \mathbf{m}_{j*}]$ matice \mathbf{M} ,
18 kde: \mathbf{m}_{i*} , resp. \mathbf{m}_{j*} je i -tý, resp. j -tý sloupec matice \mathbf{M} . Pak matice \mathbf{R} je určena:

$$\mathbf{R} = \begin{bmatrix} 0 & M_{12} & M_{13} & M_{14} \\ -M_{12} & 0 & M_{22} & M_{24} \\ -M_{13} & -M_{12} & 0 & M_{34} \\ -M_{14} & -M_{24} & -M_{34} & 0 \end{bmatrix} \quad (10.101)$$

19
20 *Upozornění* – matice kvadratické plochy \mathbf{Q} je vlastně určena maticí kvadratické plochy \mathbf{Q}_0 v základní
21 poloze a následnou pozičně-rotační transformací \mathbf{T} , tj.:

$$\mathbf{Q} = \mathbf{T}^T \mathbf{Q}_0 \mathbf{T} \quad (10.102)$$

1 **Odvození**

2 Uvažme kvadratickou plochu ve fundamentální poloze $\mathbf{x}^T \mathbf{Q}_0 \mathbf{x} = 0$ a transformujme souřadný systém
3 tak, že $\bar{\mathbf{x}} = \mathbf{T} \mathbf{x}$. Pak je zřejmé, že platí:

$$\mathbf{x}^T \mathbf{Q}_0 \mathbf{x} = (\mathbf{T}^{-1} \bar{\mathbf{x}})^T \mathbf{Q}_0 (\mathbf{T}^{-1} \bar{\mathbf{x}}) = \bar{\mathbf{x}}^T \mathbf{T}^{-T} \mathbf{Q}_0 \mathbf{T}^{-1} \bar{\mathbf{x}} \quad (10.103)$$

4 a tedy kvadratická plocha po transformaci je určena maticí $\mathbf{Q} = \mathbf{T}^{-T} \mathbf{Q}_0 \mathbf{T}^{-1}$. Pokud transformace \mathbf{T} je
5 rotace a posuv, pak $\mathbf{T}^{-T} = \mathbf{T}$.

6 V mnoha algoritmech se určuje průsečík přímky, resp. paprsku, s danou plochou parametrem t , který
7 je vlastně určen zlomkem a který je možno reprezentovat v homogenních souřadnicích. Existenci
8 průsečíku lze určit vyhodnocením diskriminantu D přímo v projektivní reprezentaci, včetně výpočtu
9 parametru t pro průsečík, pokud existuje.

10 V případě metody sledování paprsku je matice \mathbf{R} pro daný paprsek **konstantní**, a to pro výpočet
11 průsečíku se všemi kvadratickými plochami.

12 **Úloha**

13 Nalezněte efektivní postup pro výpočet průsečíku přímky s kvadratickými plochami v základní poloze.

Kulová plocha	$x^2 + y^2 + z^2 - r^2 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix}$	Elipsoid	$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 c^2 & 0 & 0 & 0 \\ 0 & a^2 c^2 & 0 & 0 \\ 0 & 0 & a^2 b^2 & 0 \\ 0 & 0 & 0 & -a^2 b^2 c^2 \end{bmatrix}$
Hyperboloid jednodílný	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 c^2 & 0 & 0 & 0 \\ 0 & a^2 c^2 & 0 & 0 \\ 0 & 0 & -a^2 b^2 & 0 \\ 0 & 0 & 0 & -a^2 b^2 c^2 \end{bmatrix}$	Hyperboloid dvojdílný	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} + 1 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 c^2 & 0 & 0 & 0 \\ 0 & a^2 c^2 & 0 & 0 \\ 0 & 0 & -a^2 b^2 & 0 \\ 0 & 0 & 0 & a^2 b^2 c^2 \end{bmatrix}$
Paraboloid * eliptický	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 2z = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 & 0 & 0 & 0 \\ 0 & a^2 & 0 & 0 \\ 0 & 0 & 0 & -a^2 b^2 \\ 0 & 0 & -a^2 b^2 & 0 \end{bmatrix}$	Paraboloid* hyperbolický	$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 2z = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 & 0 & 0 & 0 \\ 0 & -a^2 & 0 & 0 \\ 0 & 0 & 0 & -a^2 b^2 \\ 0 & 0 & -a^2 b^2 & 0 \end{bmatrix}$
Kuželová plocha	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 c^2 & 0 & 0 & 0 \\ 0 & a^2 c^2 & 0 & 0 \\ 0 & 0 & -a^2 b^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$		
Válcová plocha	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 & 0 & 0 & 0 \\ 0 & a^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -a^2 b^2 \end{bmatrix}$	Válcová plocha hyperbolická	$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0$ $\mathbf{Q}_0 = \begin{bmatrix} b^2 & 0 & 0 & 0 \\ 0 & -a^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -a^2 b^2 \end{bmatrix}$

17 * Upozornění - matice \mathbf{Q} pro tyto kvadratické plochy není diagonální.

1 **10.7.8. Detekce existence průsečíku s koulí**

2 V případě průsečíku přímky s koulí lze výpočet diskriminantu D opět podstatně zjednodušit.
 3 Předpokládejme opět, že $\mathbf{s} = [s_x, s_y, s_z: 0]^T$ a $\mathbf{x}_A = [x_A, y_A, z_A: 1]^T$ a že scénu transformujeme tak, že
 4 že koule má střed v počátku souřadného systému. Pak pro případ koule dostaváme matici \mathbf{Q}_0 :

$$\mathbf{Q}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} \quad (10.104)$$

5 Matice \mathbf{R} je konstantní pro daný paprsek, tj.:
 6 $\mathbf{R} = [\mathbf{x}_A \otimes \mathbf{s} - \mathbf{s} \otimes \mathbf{x}_A] \quad (10.105)$

7 Protože koule je v základní poloze, je nutné transformovat bod \mathbf{x}_A . Směrový vektor \mathbf{s} transformovat
 8 nemusíme, neboť na existenci průsečíku nemá délka vektoru \mathbf{s} vliv, podstatný je směr a ten zůstává
 zachován, neboť koule je rotačně invariantní, tj. jde vždy jen o posuv.

$$\begin{aligned} D = \mathbf{s}^T \mathbf{Q}^T \mathbf{R} \mathbf{Q} \mathbf{T}^{-1} \mathbf{x}_A &= \\ \begin{bmatrix} s_x, s_y, s_z: 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} \begin{bmatrix} \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_s \\ 0 & 1 & 0 & -y_s \\ 0 & 0 & 1 & -z_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} &= \\ \begin{bmatrix} s_x, s_y, s_z: 0 \end{bmatrix} \mathbf{R} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} \begin{bmatrix} x_A - x_s \\ y_A - y_s \\ z_A - z_s \\ 1 \end{bmatrix} &= \begin{bmatrix} s_x, s_y, s_z: 0 \end{bmatrix} \mathbf{R} \begin{bmatrix} x_A - x_s \\ y_A - y_s \\ z_A - z_s \\ -r^2 \end{bmatrix} \end{aligned} \quad (10.106)$$

9 Protože:

$$\mathbf{R} = \begin{bmatrix} \mathbf{B} & -\boldsymbol{\sigma} \\ \boldsymbol{\sigma}^T & 0 \end{bmatrix} \quad (10.107)$$

10 Pokud: $\boldsymbol{\sigma} = [s_x, s_y, s_z]^T$ a $\boldsymbol{\delta}_A = [x_A - x_s, y_A - y_s, z_A - z_s]^T$ a využitím vztahu (10.96) lze pak psát:

$$D = [\boldsymbol{\sigma}^T: 0] \begin{bmatrix} \mathbf{B} & -\boldsymbol{\sigma} \\ -\boldsymbol{\sigma}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_A \\ -r^2 \end{bmatrix} = [\boldsymbol{\sigma}^T \mathbf{B}: -\boldsymbol{\sigma}^T \boldsymbol{\sigma}] \begin{bmatrix} \boldsymbol{\delta}_A \\ -r^2 \end{bmatrix} = \boldsymbol{\sigma}^T \mathbf{B} \boldsymbol{\delta}_A + r^2 \boldsymbol{\sigma}^T \boldsymbol{\sigma} = \\ \boldsymbol{\sigma}^T [(\boldsymbol{\sigma} \times \boldsymbol{\xi}_A) \times \boldsymbol{\delta}_A] + r^2 \boldsymbol{\sigma}^T \boldsymbol{\sigma} \quad (10.108)$$

11

12 Je nutné zdůraznit, že:

- výraz $\boldsymbol{\sigma}^T \mathbf{B}$, resp. $(\boldsymbol{\sigma} \times \boldsymbol{\xi}_A)$ je konstantní pro danou přímku, resp. paprsek. Modifikace pro kvadratické křivky v E^2 je triviální
- projektivní modifikace je triviální, tj. $\mathbf{x}_A = [x_A, y_A, z_A: w_A]^T$ a $\mathbf{s} = [s_x, s_y, s_z: 0]^T$, pokud je parametrizace přímky s nelineární monotónní parametrizací, je směrový vektor \mathbf{s} dán jako $\mathbf{s} = [s_x, s_y, s_z: s_w]^T$, lze odvodit obdobné výpočetní postupy, viz:

$$\begin{aligned} D &= [\boldsymbol{\sigma}^T: s_w] \begin{bmatrix} \mathbf{B} & -\boldsymbol{\sigma} \\ \boldsymbol{\sigma}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_A \\ -r^2 \end{bmatrix} = [\boldsymbol{\sigma}^T \mathbf{B} - s_w \boldsymbol{\sigma}^T: -\boldsymbol{\sigma}^T \boldsymbol{\sigma}] \begin{bmatrix} \boldsymbol{\delta}_A \\ -r^2 \end{bmatrix} \\ &= \boldsymbol{\sigma}^T \mathbf{B} \boldsymbol{\delta}_A - s_w \boldsymbol{\sigma}^T \boldsymbol{\delta}_A + r^2 \boldsymbol{\sigma}^T \boldsymbol{\sigma} \end{aligned} \quad (10.109)$$

18

19 **Úloha**

20 Modifikujte výše uvedený postup pro případ výpočtu průsečíku přímky a elipsoidu, resp. dalších kvadratických ploch.

10.7.9. Výpočet průsečíku dvou trojúhelníků v E^3 – čtenář si doplní sám

1 Postup lze nalézt např. v:

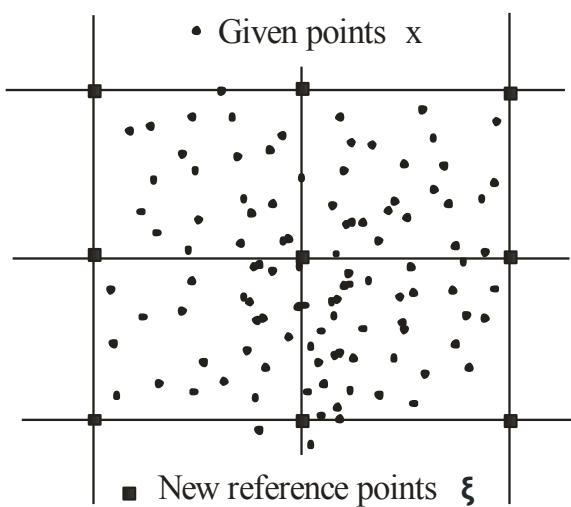
- 2
- 3 • Moller,T.A.: A Fast Triangle-Triangle Intersection Test, Journal of Graphics Tools, 1997 -
4 Taylor & Francis, ([CLICK off-line](#))
 - 5 • Wei,L.-Y.: A faster triangle-to-triangle intersection test algorithm, Computer Animation and
6 Virtual Worlds, Vol.25, pp.553-559, John Wiley&Sons, 2014

7

10.8. RBF approximace

Interpolace RBF v zásadě spoléhá na řešení soustavy lineárních rovnic $\mathbf{Ax} = \mathbf{b}$ řádu $(N + d + 1)$, kde N je počet daných bodů, d je dimenze nezávisle proměnných. Ať už v případě použití „globálních“ nebo CSRBF funkcí, matice jsou řádově velké a v mnoha případech jsou data příliš hustě vzorkována. Je tedy legitimní otázka, zda je možné „redukovat“ problém, tj. snížit počet hodnot λ_i s dobrou approximační přesností. Typickou úlohou může být reprezentace terénu, kdy máme vlastně zadané body (x, y) s výškou h .

Zkusme se proto podívat na RBF interpolaci z jiného hlediska. Předpokládejme, že máme neuspořádaná data v E^2 (dále uvedené také obecně platí pro d -dimensionální případ) a do datového setu vložme „virtuálně“ body ξ_j , viz Obr.10.18. „Virtuálně“ vložené body ξ_j , které nemusí být v pravoúhlé mřížce, může uživatel vložit tak, aby co nejlépe vystihovaly daný povrch, např. profil terénu apod. Počet vložených bodů ξ_j bude M a bude podstatně menší než počet zadaných bodu N , tj. $M \ll N$.



Obr.10.18: RBF approximace a redukce bodů

Aplikujeme-li RBF, pak interpolovaná hodnota může být určena obdobně definovanou funkcí, přičemž hodnota r_{ij} je určena nyní vztahem $r_{ij} = \|x_i - \xi_j\|$. Jde tedy o vzdálenost mezi body danými a body virtuálně vloženými. Interpolovaná hodnota je tedy určena vzorcem:

$$f(x) = \sum_{j=1}^M \lambda_j \varphi(\|x - \xi_j\|) \quad (10.110)$$

přičemž interpolační funkce $\varphi(r)$ jsou stejné jako v případě RBF interpolací. Je zřejmé, že opět pro dané hodnoty dostáváme soustavu rovnic:

$$f(x_i) = \sum_{j=1}^M \lambda_j \varphi(\|x_i - \xi_j\|) \quad (10.111)$$

$$h_i = f(x_i) \quad i = 1, \dots, N$$

kde ξ_j nejsou dané body, ale body „virtuálně“ vložené. Pro jednoduchost nyní uvažme, např. že pokud poměr počtu zadaných hodnot a počtu „virtuálně“ vložených je $N/M = 10^2$, pak rychlosť vlastního výpočtu pouze jedné approximované hodnoty bude také cca $10^2 \times$ rychlejší.

Výše uvedená formulace vede k řešení soustavy lineárních rovnic $\mathbf{Ax} = \mathbf{b}$, kde počet řádek $N \gg M$ a kde M je počet neznámých vah $[\lambda_1, \dots, \lambda_M]^T$.

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,M} \\ \vdots & \ddots & \vdots \\ \varphi_{i,1} & \cdots & \varphi_{i,M} \\ \vdots & \ddots & \vdots \\ \varphi_{N,1} & \cdots & \varphi_{N,M} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_N \end{bmatrix} \quad (10.112)$$

1
2 Soustava rovnic je tedy přeurovenou soustavou, tj. je více rovnic než neznámých. Takovou soustavu
3 můžeme řešit např. metodou nejmenších čtverců (Least Square Error - LSE) nebo dekompozicí
4 singulárních hodnot (Singular Value Decomposition - SVD).

5
6 Výhodou tohoto přístupu je, že volba „virtuálně“ vložených bodů je libovolná, tj. nemusí být
7 v pravidelné síti, a tedy lze snadno zlepšit přesnost approximace v oblasti uživatelského zájmu, resp. lépe
8 vystihnout vlastnosti a chování dat. Toto je zejména výhoda vlastnost v inženýrských aplikacích a
9 aplikacích GIS.

10
11 Radiální bázové funkce se dnes používají i v jiných oblastech, např. pro řešení parciálních
12 diferenciálních rovnic (Partial Differential Equations – PDE). Jejich výhodou je, že nepotřebují
13 vytváření sítí, na druhé straně však vznikají komplikace s ostrými přechody apod.

14
15

11. Parametrické křivky a plochy

Parametrický popis je poměrně mocným a flexibilním nástrojem k popisu objektů, resp. jejich povrchů. Nicméně některé „běžné“ operace, např. průsečík přímky s parametrickou křivkou nebo rovinou, průsečík roviny s parametrickou plochou atd., jsou komplikovanější. Parametrický popis dat je nedílnou součástí mnoha systémů, zejména pak CAD/CAM a GIS systémů.

Parametrická křivka v E^2 , resp. v E^3 dána svými souřadnicemi v parametrickém tvaru $x(t), y(t)$, resp. $x(t), y(t), z(t)$. Pro jednoduchost se pak souřadnice souhrnně značí $\mathbf{P}(t) = (x(t), y(t))$, resp. $\mathbf{P}(t) = (x(t), y(t), z(t))$.

Parametrická křivka může být vyjádřena různými způsoby. Pro jednoduchost uvažme kružnici $x^2 + y^2 - r^2 = 0$. Parametrické vyjádření pak např. může být

$$\begin{aligned} x &= r \cos t & x &= r \frac{1-t^2}{1+t^2} \\ y &= r \sin t & y &= r \frac{2t}{1+t^2} \\ t &\in \langle 0, 2\pi \rangle & t &\in \langle 0, 1 \rangle \end{aligned} \quad (11.1)$$

Je tedy zřejmé, že jednu parametrickou křivku lze vyjádřit různými způsoby.

Uveďme nyní některé pojmy, určující vlastnosti parametrických křivek, a to:

- parametrická křivka $\mathbf{P}(t)$ se nazývá křivkou *regulární*, pokud pro její derivaci platí

$$|\mathbf{P}'(t)| \neq 0 \quad (11.2)$$

pro $\forall t$ na daném intervalu. $\mathbf{P}'(t)$ je pak tečným vektorem křivky v daném bodě.

Poznámka – pokud je křivka dána v implicitním tvaru jako průsečnice dvou ploch, tj.

$$f(x, y, z) = 0 \quad g(x, y, z) = 0 \quad (11.3)$$

pak pokud matice

$$\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix} \quad (11.4)$$

má hodnost dva pro všechny body křivky, pak jde o regulární implicitně definovanou křivku.

- spojitost (hladkost) křivek*

- křivka má *parametrickou spojitost* k -tého stupně, značí se C^k , pokud je parametrická křivka spojitá ve všech bodech na daném intervalu, včetně všech derivací až do stupně k . Pokud $k \geq 1$, pak se tato vlastnost označuje též jako *hladkost parametrické křivky*.
- geometrická spojitost* označovaná G^k – jde o slabší podmítku, kdy je jen požadována rovnoběžnost tečných vektorů, což se využívá zejména při napojování dvou křivek, tzv. segmentů, obvykle s definičním intervalom $\langle 0, 1 \rangle$. Z podmínky spojitosti vyplývá, že koncový bod $\mathbf{P}(1)$ prvního segmentu musí být totožný s počátečním bodem $\mathbf{Q}(0)$ segmentu druhého, tj.

$$\mathbf{P}(1) = \mathbf{Q}(0) \quad (11.5)$$

1 pak pro geometrickou spojitost G^n musí platit

$$\frac{d^k \mathbf{P}(1)}{dt^k} = q_k \frac{d^k \mathbf{Q}(0)}{dt^k} \quad q_k > 0 \quad (11.6)$$

2 pro všechna $k = 1, \dots, n$.

- 3 • *reparametrisace křivky* slouží obecně ke změně „rychlosti“ pohybu bodu po regulární
4 parametrické křivce vzhledem ke změně parametru t . Pokud je $g(t)$ reparametrisující
5 funkce, pak by měla splňovat následující podmínky
 - 6 ○ funkce $g(s)$ je monotónně rostoucí
 - 7 ○ $g(0) = 0$ a $g(1) = 1$, resp. $g(0) = a$ a $g(1) = b$, pokud uvažujeme obecný interval
8 $\langle a, b \rangle$ pro parametr křivek.

9 speciálním případem je *reparametrisace obloukem*, kdy parametr vlastně vyjadřuje délku
10 křivky. Předpokládejme funkci délky oblouku

$$g(t) = \int_0^t |\mathbf{P}'(\tau)| d\tau \quad (11.7)$$

11 Funkce $g(t)$ je funkci rostoucí a k ní existuje inverzní funkce $t(s)$. Pak bod na
12 reparametrisované křivce je dán

$$\mathbf{P}(t(s)) = \mathbf{Q}(s) \quad (11.8)$$

13 Lze ukázat, že tečný vektor křivky reparametrisované obloukem bude v každém bodě
14 jednotkový.

15 V případě E^3 je situace složitější, neboť je nutné zmínit základní pojmy a Frenetovy vzorce
16 (podrobněji viz „Diferenciální geometrie“), a to:

- 17 • *oskulační rovina* $\mathbf{\tau}(u, v)$ křivky $\mathbf{P}(t)$ v bodě t_0 je dána vztahem

$$\mathbf{\tau}(u, v) = \mathbf{P}(t_0) + \mathbf{P}'(t_0)u + \mathbf{P}''(t_0)v \quad (11.9)$$

18 pokud bod t_0 je inflexním bodem, pak $\mathbf{P}'(t_0)$ a $\mathbf{P}''(t_0)$ jsou lineárně závislé a tedy vektory
19 jsou rovnoběžné.

20 Je zřejmé, že tečnou $\mathbf{P}'(t_0)$ v bodě t_0 křivky $\mathbf{P}(t)$ prochází vlastně svazek tečných rovin.
21 Tečný vektor $\mathbf{P}'(t_0)$ se většinou označuje jako \mathbf{t} . Rovina, která prochází daným bodem křivky
22 $\mathbf{P}(t_0)$ a má směrový vektor společný pro svazek rovin.

- 23 • *binormála* \mathbf{b} - vektor procházející bodem $\mathbf{P}(t_0)$ a který je kolmý na oskulační rovinu se
24 nazývá binormálou a označuje se jako \mathbf{b} . Je tedy zřejmé, že binormála v daném bodě t_0 je
25 kolmá na oba vektory $\mathbf{P}'(t_0)$ a $\mathbf{P}''(t_0)$ definující oskulační rovinu $\mathbf{\tau}(u, v)$. Binormála \mathbf{b} je
26 definována vektorovým součinem

$$\mathbf{b} = \mathbf{P}'(t_0) \times \mathbf{P}''(t_0) \quad (11.10)$$

- 27 • *hlavní normála* \mathbf{n} – jeden z vektorů kolmých na tečný vektor $\mathbf{P}'(t_0)$, který je též kolmý na
28 binormálu, a tedy $\mathbf{n} = \mathbf{b} \times \mathbf{t}$. Je tedy zřejmé, že platí tzv. Frenetovy vzorce:

$$\mathbf{n} = \mathbf{b} \times \mathbf{t} \quad \mathbf{b} = \mathbf{t} \times \mathbf{n} \quad \mathbf{t} = \mathbf{n} \times \mathbf{b} \quad (11.11)$$

1 Výše uvedené vektory je vhodné *normalizovat*.

- 2
- 3 • *křivosti* – křivost křivky je vlastností, která intuitivně říká, jak se křivka odchyluje od přímky, a
4 rozlišují se dvě křivosti

$$k_1 = \frac{|\mathbf{P}'(t_0) \times \mathbf{P}''(t_0)|}{|\mathbf{P}'(t_0)|^3} \quad k_2 = \frac{(\mathbf{P}'(t_0) \times \mathbf{P}''(t_0)) \cdot \mathbf{P}'''(t_0)}{|\mathbf{P}'(t_0) \times \mathbf{P}''(t_0)|^2} \quad (11.12)$$

5 Křivost k_1 se nazývá flexe (curvature) a v inflexním bodě je nulová. Křivost k_2 se nazývá torze
6 (torsion)

- 7
- 8 • *oskulační kružnice* – oskulační kružnice v bodě $\mathbf{P}(t_0)$ je kružnice ležící v oskulační rovině, jejíž
9 tečna je stejná jako tečna v daném bodě křivky mající střed na hlavní normále a poloměr
daný vztahem

$$r = 1/k_1 \quad (11.13)$$

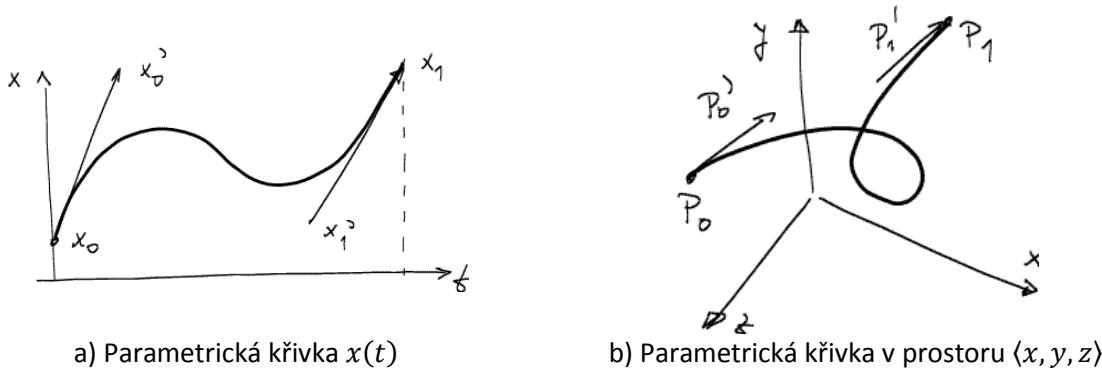
10 11.1. Parametrické kubické křivky

11 Jedna třída běžných parametrických interpolací je založena na kubických parametrických křivkách a
12 bikubických parametrických plátech. Pravděpodobně nejčastěji jsou používány křivky a pláty založené
13 na Hermitově, Bézierově a Coonsově formulaci.

14

15 Hermitova (Fergusonova) křivka

16 Hermitova křivka je určena souřadnicemi počátečního a koncového bodu křivky a tečnými vektory
17 v těchto bodech. Na Obr.11.1.a je křivka nakreslena pro souřadnici x . Obr.11.1.b pak zobrazuje křivku
18 v prostoru E^3 . V následujícím budeme používat značení $P = (x, y, z)$.



Obr.11.1: Hermitova parametrická křivka

19 Pro pochopení uveďme jednoduché odvození Hermitovy křivky. Pokud chceme prokládat 3 body
20 hladkou křivkou, pak křivka musí být alespoň 2. stupně. Pro proložení více body potřebujeme
21 polynom vyššího stupně, nebo polynomy nižšího stupně, které budeme napojovat. Lze ukázat, že 2
22 křivky 2. stupně nelze obecně hladce napojit tak, aby v místě napojení nebyl „zlom“, tj. aby alespoň
23 existovala první derivace v bodě napojení. Proto křivky musí být alespoň 3. stupně, tj. mající inflexní
24 bod.

25

26 Hermitova formulace vychází ze znalosti koncových bodů a směrnic křivky v koncových bodech. Pro
27 parametrickou křivku $x(t)$ 3. stupně lze tedy psát:

$$x(t) = at^3 + bt^2 + ct + d \quad x'(t) = 3at^2 + 2bt + c \quad (11.14)$$

1 kde: $x'(t) = \frac{dx(t)}{dt}$. Dosazením za $t = 0$ a $t = 1$ dostáváme 4 rovnice pro 4 neznámé pro a, b, c, d

$$\begin{aligned} x(0) &= d & x'(0) &= c \\ x(1) &= a + b + c + d & x'(1) &= 3a + 2b + c \end{aligned} \quad (11.15)$$

2

3 Pro určení hodnot a, b, c, d musíme vyřešit soustavu rovnic:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix} \quad (11.16)$$

4 Řešením této soustavy, tj. $\mathbf{Ax} = \mathbf{b}$, dostáváme koeficienty Hermitovy formy a, b, c, d a můžeme psát

$$x(t) = at^3 + bt^2 + ct + d = (\mathbf{A}^{-1}\mathbf{x})^T \mathbf{t} = \mathbf{x}^T \mathbf{M}_H \mathbf{t} \quad (11.17)$$

5 kde: $\mathbf{t} = [t^3, t^2, t, 1]^T$ a $\mathbf{x} = [x(0), x(1), x'(0), x'(1)]^T$.

6

7 Inverzní matice \mathbf{A}^{-1} je pak v našem případě maticí Hermitovy formy \mathbf{M}_H , tj.

$$\mathbf{M}_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \quad (11.18)$$

8 Obdobně postupujeme i pro ostatní souřadnice, tj. pro y , resp. z atd. Pro usnadnění zápisu se používá symbol P , který pak vlastně zastupuje jednotlivé souřadnice obecně v d -rozměrném prostoru. Je vhodné zdůraznit, že matice \mathbf{M}_H pro danou formu je konstantní.

11

12 Interpolační funkce (blending functions) Hermitovy formy $\mathbf{g}_H(t)$, viz Obr.11.2, jsou pak dány výrazem:

$$\mathbf{g}_H(t) = [g_0(t), g_1(t), g_2(t), g_3(t)]^T = \mathbf{M}_H [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle \quad (11.19)$$

14

15 Rozepsáním maticové formy dostáváme:

$$\begin{aligned} g_0(t) &= 2t^3 - 3t^2 + 1 & g_2(t) &= t^3 - 2t^2 + t \\ g_1(t) &= -2t^3 + 3t^2 & g_3(t) &= t^3 - t^2 \end{aligned} \quad (11.20)$$

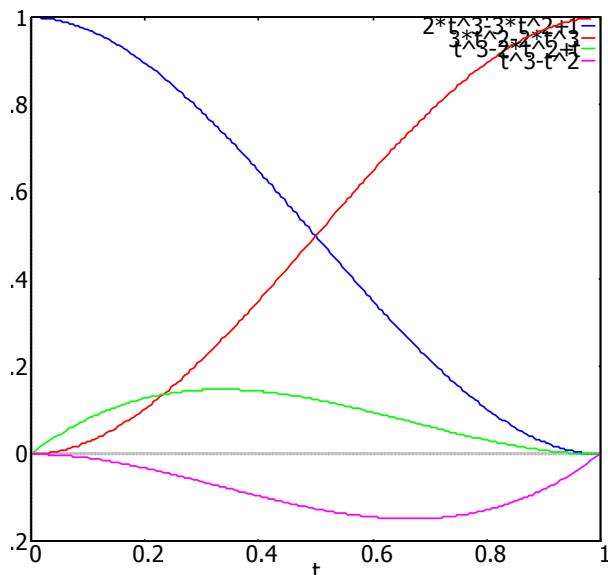
16

17 Bod křivky $\mathbf{P}(t) = (x(t), y(t), z(t))$ je určen:

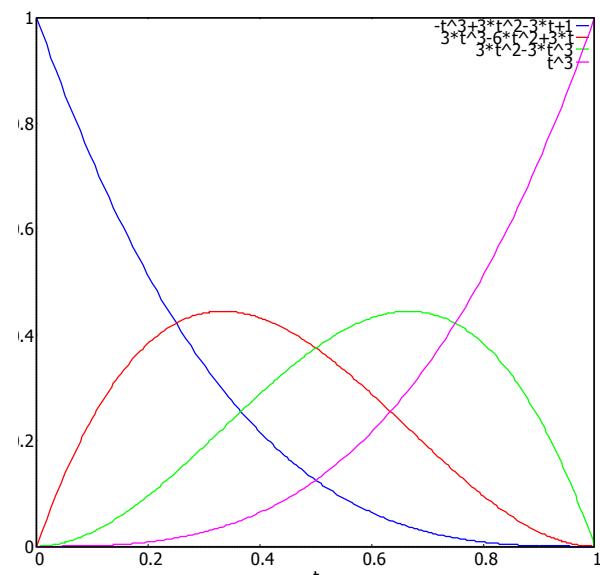
$$\mathbf{P}(t) = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}'_0, \mathbf{P}'_1] \mathbf{M}_H [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle \quad (11.21)$$

18

19 kde \mathbf{M}_H je matice Hermitovy formy.



Obr.11.2: Hermitovy interpolační funkce



Obr.11.3: Bézierovy interpolační funkce

1 **Bézierova křivka**

2 Bézierova křivka obecně n -tého stupně je určena:

$$\mathbf{P}(t) = \sum_{i=0}^n B_{n,i} \mathbf{P}_i = \sum_{i=0}^n \binom{n}{i} \mathbf{P}_i t^i (1-t)^{n-i} \quad B_{n,i} = \binom{n}{i} t^i (1-t)^{n-i} \quad (11.22)$$

4 kde: $B_{n,i}$ jsou Bernsteinovy polynomy.

6 Pro 3.stupeň je pak Bézierova křivka určena 4 body, viz Obr.11.3, a křivka je určena rovnicí:

$$\mathbf{P}(t) = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \mathbf{M}_B [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle \quad (11.23)$$

8 Matice Bézierovy formy je definována:

$$\mathbf{M}_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (11.24)$$

10 Interpolační funkce (blending functions) Bézierovy formy $\mathbf{g}_B(t)$, viz Obr.11.3, jsou dány výrazem:

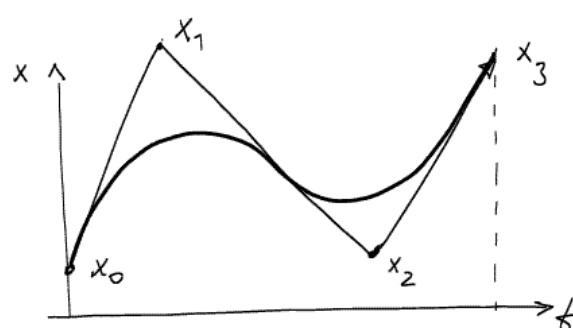
$$\mathbf{g}_B(t) = [g_0(t), g_1(t), g_2(t), g_3(t)]^T = \mathbf{M}_B [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle \quad (11.25)$$

12 Rozepsáním maticové formy dostáváme interpolační funkce pro Bézierovu křivku:

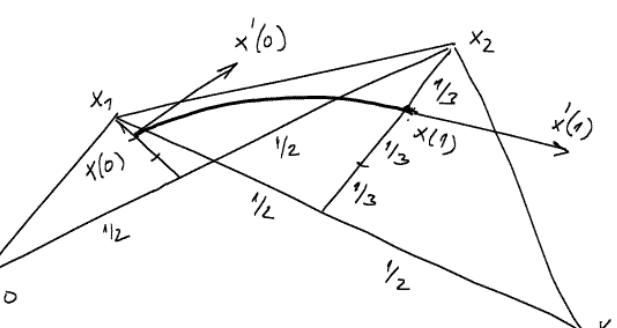
$$\begin{aligned} g_0(t) &= -t^3 + 3t^2 - 3t + 1 & g_1(t) &= 3t^3 - 6t^2 + 3t \\ g_2(t) &= -3t^3 + 3t^2 & g_3(t) &= t^3 \end{aligned} \quad (11.26)$$

14 **Upozornění**

15 Důležitou vlastností Bézierovy křivky je, že pro $t \in \langle 0,1 \rangle$ je křivka pro každou souřadnici vždy uvnitř konvexní obálky daných 4 řídících bodů. Tato vlastnost je dána tím, že součet interpolačních funkcí je roven hodnotě 1 pro všechny hodnoty $t \in \langle 0,1 \rangle$. To znamená, že všechny body křivky pro $t \in \langle 0,1 \rangle$ jsou uvnitř konvexní obálky dané řídícími body Bézierovy křivky. V případě křivky v E^3 je tedy křivka uvnitř tetrahedronu, který je dán čtyřmi řídícími body křivky.



Obr.11.4: Bézierova křivka



Obr.11.5: Coonsova křivka

22 **Poznámka**

23 Pozice řídících bodů v parametrickém prostoru, tj. (x, t) , jsou $(x_0, 0), (x_1, 1/3), (x_2, 2/3), (x_3, 1)$. Analogicky pro ostatní souřadnice y a z .

1 Lze ukázat, že pro převod řídících hodnot Hermitovy a Bézierovy formy platí:

$$\mathbf{x}'(0) = 3(\mathbf{x}_1 - \mathbf{x}_0) \quad \mathbf{x}'(1) = 3(\mathbf{x}_3 - \mathbf{x}_2) \quad (11.27)$$

2
3 Kubická Hermitova nebo Bézierova křivka je určena dvěma body, kterými křivka prochází, a další dvě
4 řídící hodnoty určují její tvar, jde tedy o *interpolační křivky*. Existují však i jiné formulace
5 parametrických křivek, založené na *aproximaci*, tj. křivka danými body obecně neprochází, např.
6 Coonsova křivka, která patří do B-Spline parametrických křivek.

8 Coonsova křivka

9 Další parametrickou křivkou je Coonsova křivka, která patří do skupiny B-Spline interpolací, nicméně
10 jde vlastně o approximační křivku. Křivka je opět dána čtyřmi řídícími body a je určena rovnicí:

$$\mathbf{P}(t) = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \mathbf{M}_C [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.28)$$

11
12 Tato křivka, na rozdíl od předchozích křivek, *neprochází počátečním a koncovým řídícím bodem*, viz
13 Obr.11.5. Jde tedy vlastně o *aproximační křivku*. Matice Coonsovy formy je definována:

$$\mathbf{M}_C = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (11.29)$$

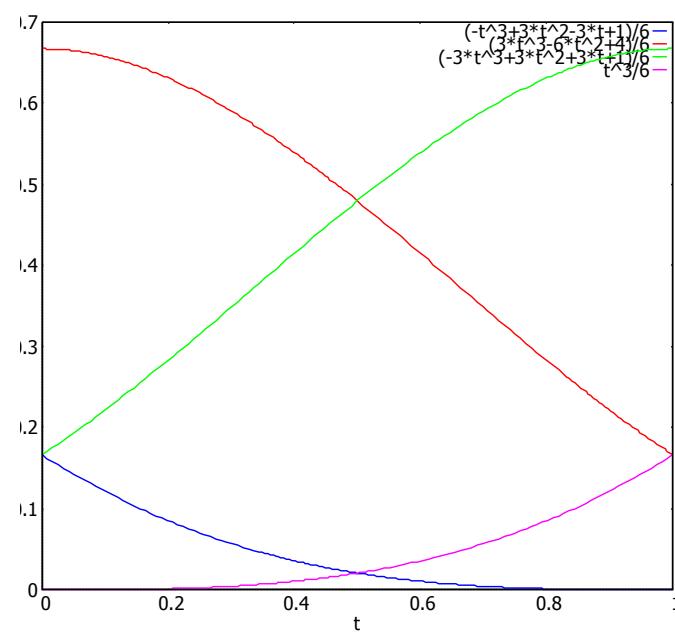
14 V případě interpolace pro více bodů se Coonsovy křivky hladce C^2 napojují. V případě Hermitovy a
15 Bézierovy křivky jde o výpočetní postup. Jednotlivé segmenty Coonsovy křivky se hladce napojují
16 vlastně pouhým „překrýváním“ po jednotlivých intervalech, tj. intervalů i a $i + 1$.

17 Interpolační funkce Coonsovy formy $\mathbf{g}_C(t)$ jsou dány výrazem:

$$\mathbf{g}_C(t) = \mathbf{M}_C [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.30)$$

18 Rozepsáním pak:

$$\begin{aligned} g_0(t) &= (-t^3 + 3t^2 - 3t + 1)/6 & g_1(t) &= (3t^3 - 6t^2 + 4)/6 \\ g_2(t) &= (-3t^3 + 3t^2 + 3t + 1)/6 & g_3(t) &= t^3/6 \end{aligned} \quad (11.31)$$



Obr.11.6: Coonsovy interpolační funkce

20

Závěr

Všechny výše uvedené křivky lze popsat jedním vztahem, a to:

$$\mathbf{P}(t) = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \mathbf{M}_F [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.32)$$

kde: \boldsymbol{M}_F je matici příslušné formy a P_i jsou řídící hodnoty podle specifikace příslušné formy.

Interpolaci (blending) funkce $g_F(t)$ jsou ve všech případech dány výrazem:

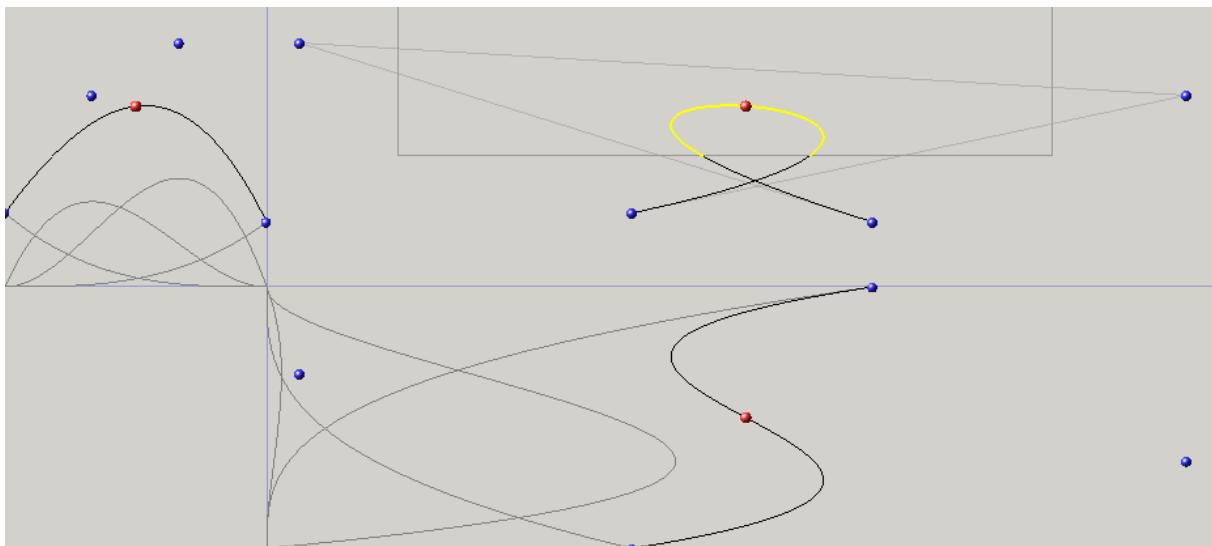
$$g_F(t) = M_F [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.33)$$

Je tedy zřejmé, že popis parametrické interpolace kubickými křivkami je poměrně jednoduchý.

Je vhodné upozornit, že je možná reparametrisace křivky, a to $t = \varphi(\tau)$, a tím ovlivnit „rychlosť“ pohybu po křivce. Musí platit $\varphi(0) = 0$, $\varphi(1) = 1$, $\varphi(\tau_1) < \varphi(\tau_2)$ pro $\tau_1 < \tau_2$ $\tau \in (0,1)$, tj. funkce $\varphi(\tau)$ musí být monotónně rostoucí pro $\tau \in (0,1)$.

Příklad

Ukázka chování Bézierovy křivky v závislosti na řídících bodech.



Obr.11.7: Béziérova křivka – manipulace (CLICK-EXE)

Adresář *Bezier* obsahuje soubory nutné ke spuštění programu, který umožňuje manipulaci s Bézierovou křivkou.

11.2. Vykreslování parametrických křivek

2 Vykreslování parametrických křivek se zdá být poměrně jednoduchou záležitostí. Naivní řešení, které
 3 je velmi často realizováno, je založeno na zvolení přírůstku parametru, např. $\Delta t = 0.01$, a kreslení
 4 lomené čáry. Je nutné si však uvědomit, že:

- 5 • pro konstantní přírůstek parametru Δt , délka jednotlivých segmentů lomené čáry není
 6 konstantní
- 7 • délka vykreslované křivky pro interval parametru $t \in \langle t_0, t_1 \rangle$ je určena nelineárním vztahem:

$$l = \int_{t_0}^{t_1} \sqrt{x'^2(t) + y'^2(t)} dt \quad (11.34)$$

8 V případě křivky v E^3 je délka křivky pro $t \in \langle t_0, t_1 \rangle$ určena:

$$l = \int_{t_0}^{t_1} \sqrt{x'^2(t) + y'^2(t) + z'^2(t)} dt \quad (11.35)$$

9 Pro celou křivku, kdy $t \in \langle 0, 1 \rangle$ dostáváme:

$$l = \int_0^1 \sqrt{x'^2(t) + y'^2(t) + z'^2(t)} dt \quad (11.36)$$

- 10 • pokud má být křivka vykreslena úseky o stejné délce, je nutné najít odpovídající hodnoty
 11 t_0, t_1, \dots, t_n . Nicméně, aby vykreslená křivka byla hladká, bude nutné volit intervaly $\langle t_i, t_{i+1} \rangle$
 12 dostatečně „malé“ a podle křivosti apod.

13 Je tedy zřejmé, že korektní a „dobré“ vykreslování křivky je poněkud složitější, než by se očekávalo.
 14 Podrobněji viz předměty *Geometrické modelování* apod.
 15 Pokud máme určeny intervaly $\langle t_i, t_{i+1} \rangle$, např. pomocí $\Delta t = 0.01$, pak je otázkou, jak křivku vykreslit.
 16 Z předchozího je známo, že kubická křivka je dána výrazem:

$$\mathbf{P}(t) = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \mathbf{M}_F [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.37)$$

18 Protože výraz $[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \mathbf{M}_F$ je konstantní pro všechny hodnoty t , dostáváme standardní
 19 mocninou řadu s konstantními parametry a její hodnota se vypočte Hornerovým schématem, a to:

$$S = \sum_{i=0}^n a_i t^i = ((a_n t + a_{n-1}) t + a_{n-2}) t + \dots + a_0 \quad (11.38)$$

20 Není tedy nutné počítat mocniny pro parametr t . Nicméně je nutné zdůraznit, že vyžaduje 9 násobení
 21 a 10 sčítání v případě 3D křivky na jeden krok, což je poměrně výpočetně náročné.

Další algoritmy pro vykreslování

24 Zajímavým algoritmem vykreslování Bézierových kubických parametrických křivek je algoritmus
 25 *de Casteljau*, který je založen na postupném dělení úseček řídícího n-úhelníka Bézierovy formy. Další
 26 možností je použití diferenčního schématu, které je založeno na tom, že druhá differenze pro křivku
 27 3. stupně je konstantní, viz doporučená literatura.

29 Je zřejmé, že různé formy kubických křivek mají jisté výhody a nevýhody. Je otázkou, zda a jak lze
 30 jednotlivé formy vzájemně převádět.

11.3. Vzájemný převod kubických parametrických křivek

2 Dříve uvedené kubické křivky jsou po formální stránce reprezentované jednou rovnicí, a to:

$$\mathbf{P}(t) = [P_0, P_1, P_2, P_3] \mathbf{M}_F [t^3, t^2, t, 1]^T \quad t \in \langle 0, 1 \rangle \quad (11.39)$$

3 nebo:

$$\mathbf{P}(t) = [P_0, P_1, P_2, P_3] \mathbf{M}_F \mathbf{t} \quad t \in \langle 0, 1 \rangle \quad (11.40)$$

4 kde: $\mathbf{t} = [t^3, t^2, t, 1]^T$ a $t \in \langle 0, 1 \rangle$. Je tedy zřejmé, že po formální stránce lze jednotlivé reprezentace
5 mezi sebou převádět, tj. transformovat. Pokud máme dvě křivky, např. Hermitovu a Bézierovu, pak
6 lze psát pro x souřadnici Hermitovy křivky:

$$x_H(t) = [{}^H x_0, {}^H x_1, {}^H x'_0, {}^H x'_1] \mathbf{M}_H [t^3, t^2, t, 1]^T \quad (11.41)$$

7 a pro Bézierovu křivku:

$$x_B(t) = [{}^B x_0, {}^B x_1, {}^B x_2, {}^B x_3] \mathbf{M}_B [t^3, t^2, t, 1]^T$$

8 Pokud chceme mít stejnou parametrickou křivku v obou případech, pak musí platit:

$$x_B(t) = x_H(t) \quad t \in \langle 0, 1 \rangle \quad (11.42)$$

9 tj.

$$[{}^B x_0, {}^B x_1, {}^B x_2, {}^B x_3] \mathbf{M}_B [t^3, t^2, t, 1]^T = [{}^H x_0, {}^H x_1, {}^H x'_0, {}^H x'_1] \mathbf{M}_H [t^3, t^2, t, 1]^T \quad (11.43)$$

10 a tedy:

$$[{}^B x_0, {}^B x_1, {}^B x_2, {}^B x_3] \mathbf{M}_B = [{}^H x_0, {}^H x_1, {}^H x'_0, {}^H x'_1] \mathbf{M}_H \quad (11.44)$$

11 Protože matice \mathbf{M}_B je regulární a rovnici lze násobit maticí inverzní \mathbf{M}_B^{-1} . Pro převod z Hermitovy do
12 Bézierovy formy pak dostáváme:

$$[{}^B x_0, {}^B x_1, {}^B x_2, {}^B x_3] = [{}^H x_0, {}^H x_1, {}^H x'_0, {}^H x'_1] \mathbf{M}_H \mathbf{M}_B^{-1} \quad (11.45)$$

13 Zkráceně můžeme psát:

$$\mathbf{M}_{H \rightarrow B} = \mathbf{M}_H \mathbf{M}_B^{-1} \quad (11.46)$$

14 Je zřejmé, že transformační matice $\mathbf{M}_{H \rightarrow B}$ pro převod z Hermitovy do Bézierovy formy je platná také
15 pro souřadnice y a z . Ostatní transformační matice lze snadno odvodit obdobným postupem.

16
17 Vzájemné převody jsou tedy možné a transformační matice jsou uvedeny v tabulce Tab.XXX.
18 Transformace jsou výhodné např. pro interaktivní modifikaci, kdy uživatelská interakce probíhá např.
19 v Bézierově formě, zatímco výpočty hladkosti probíhají v Hermitově formě apod.

	Z	Hermite	Bézier	Coons
Do	Hermite	-----	$\begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix}$	$\frac{1}{6} \begin{bmatrix} 1 & 0 & -3 & 0 \\ 4 & 1 & 0 & -3 \\ 1 & 4 & 3 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix}$
	Bézier	$\frac{1}{3} \begin{bmatrix} 3 & 3 & 0 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$	-----	$\frac{1}{6} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 4 & 2 & 1 \\ 1 & 2 & 4 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
	Coons	$\frac{1}{3} \begin{bmatrix} -3 & 6 & -3 & 6 \\ 6 & -3 & 6 & -3 \\ -7 & 2 & -1 & 2 \\ -2 & 1 & -2 & 7 \end{bmatrix}$	$\begin{bmatrix} 6 & 0 & 0 & 0 \\ -7 & 2 & -1 & 2 \\ 2 & -1 & 2 & -7 \\ 0 & 0 & 0 & 6 \end{bmatrix}$	-----

21 Tab.XXX: Matice vzájemných převodů

11.4. Hladké napojování kubických křivek

Představme si, že chceme interpolovat pohyb objektu po nějaké složitější prostorové křivce (trajektorii). Jako příklad uvedeme napojování křivek pro Hermitovu formu. Předpokládejme, že pohyb objektu je popsán po částech kubickými křivkami:

$$P(t) = \mathbf{P}^T \mathbf{M}_H \mathbf{t} \quad \mathbf{P} = [P_0, P_1, P'_0, P'_1]^T \quad \mathbf{t} = [t^3, t^2, t, 1]^T \quad (11.47)$$

Je zřejmé, že můžeme mít následující požadavky v bodě napojení dvou kubických křivek, a to:

- C^0 spojitost - objekt nemění skokem svoji **pozici**, tedy koncový bod 1. segmentu je stejný jako počáteční pozice 2. segmentu, tj. ${}^{(1)}P(1) = {}^{(2)}P(0)$, resp. obecně ${}^{(i-1)}P(1) = {}^{(i)}P(0)$
- C^1 spojitost - objekt nemění skokem svoji **rychlosť**, tedy že časová derivace pozice koncového bodu 1. segmentu je stejná jako počáteční časová derivace pozice 2. segmentu, tj. $\frac{d}{dt} {}^{(1)}P(1) = \frac{d}{dt} {}^{(2)}P(0)$, resp. obecně ${}^{(i-1)}P'(1) = {}^{(i)}P'(0)$ & $\frac{d}{dt} {}^{(i-1)}P(1) = \frac{d}{dt} {}^{(i)}P(0)$
- C^2 spojitost - **zrychlení** se nemění skokem, tj. $\frac{d^2}{dt^2} {}^{(1)}P(1) = \frac{d^2}{dt^2} {}^{(2)}P(0)$, resp. obecně $\frac{d^2}{dt^2} {}^{(i-1)}P(1) = \frac{d^2}{dt^2} {}^{(i)}P(0)$

Hladké napojení dvou Hermitových křivek

Hladké napojení Hermitovy křivky C^2 spojením segmentů ${}^{(1)}P(t)$ a ${}^{(2)}P(t)$:

$$P(t) = \mathbf{P}^T \mathbf{M}_H \mathbf{t} \quad \mathbf{P} = [P_0, P_1, P'_0, P'_1]^T \quad \mathbf{t} = [t^3, t^2, t, 1]^T \quad (11.48)$$

Pro:

$$P(t) = at^3 + bt^2 + ct + d \quad \frac{d}{dt} P(t) = 3at^2 + 2bt + c \quad \frac{d^2}{dt^2} P(t) = 6at + 2b \quad (11.49)$$

$${}^{(i-1)}P''(1) = {}^{(i)}P''(0) \quad \text{vede na podmínu:} \quad 6^{(i-1)}a + 2^{(i-1)}b = 2^{(i)}b \quad (11.50)$$

$$\begin{aligned} 2 \left[3 \left({}^{(i)}P - {}^{(i-1)}P \right) - 2 \left({}^{(i-1)}P' - {}^{(i)}P' \right) \right] + 6 \left[2 \left({}^{(i-1)}P - {}^{(i)}P \right) + {}^{(i-1)}P' + {}^{(i)}P' \right] \\ = 2 \left[3 \left({}^{(i+1)}P - {}^{(i)}P \right) - 2 {}^{(i)}P' - {}^{(i+1)}P' \right] \end{aligned} \quad (11.51)$$

Pokud podmínky spojíme dohromady spolu s podmínkou ${}^{(1)}P(1) = {}^{(2)}P(0)$ dostáváme:

Zjednodušením pak:

$${}^{(i-1)}P' + 4 {}^{(i)}P' + {}^{(i+1)}P' = 3 \left({}^{(i+1)}P - {}^{(i-1)}P \right) \quad (11.52)$$

a v maticové formě pak:

$$\begin{bmatrix} 2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \ddots & \vdots \\ 0 & 1 & 4 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & 1 & 4 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} {}^{(0)}P' \\ {}^{(1)}P' \\ \vdots \\ {}^{(m-2)}P' \\ {}^{(m-1)}P' \\ {}^{(m-1)}P' \end{bmatrix} = \begin{bmatrix} {}^{(0)}P' \\ 3 \left({}^{(2)}P - {}^{(0)}P \right) \\ \vdots \\ 3 \left({}^{(m-1)}P - {}^{(m-3)}P \right) \\ \vdots \\ {}^{(m-1)}P' \end{bmatrix} \quad (11.53)$$

Pokud ${}^{(0)}P''(0) = {}^{(m)}P''(0) = 0$ pak jde o tzv. přirozený kubický spline (*natural cubic spline*).

1 Coonsova křivka

2 Coonsova křivka je *aproximačního typu*, tj. křivka neprochází zadánymi body a má pouze řídící body.

3 Jednotlivé segmenty se napojují C^2 spojitě prostým překrýváním intervalů. Je však otázkou, jak:

- 4 • křivku „dotáhnout“ do koncových bodů
- 5 • docílit „uzavřené“ křivky

6 Oba problémy jsou snadno řešitelné, pokud si uvědomíme vlastnosti Coonsovy křivky a konstrukci
7 vykreslované křivky.

8 Předpokládejme, že jsou dány řídící body x_0, \dots, x_{n-1} .

11 Konstrukce uzavřené křivky

12 Konstrukce uzavřené křivky je poměrně jednoduchá, pokud si uvědomíme, že tvar výsledné křivky
13 nemůže záviset na pořadí zadávaných bodů. Takže uzavřená křivka daná Coonsovými kubikami je
14 určena řídícími body jednotlivých segmentů (pořadí jednotlivých segmentů je vhodné dodržet, ale
15 není to nutné) takto:

- 16 • standardní sekvence:
17 $[x_0, x_1, x_2, x_3], [x_1, x_2, x_3, x_4], \dots, [x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}]$
- 18 • „uzavírací“ sekvence:
19 $[x_{n-3}, x_{n-2}, x_{n-1}, x_0], [x_{n-2}, x_{n-1}, x_0, x_1], [x_{n-1}, x_0, x_1, x_2]$

20 Vidíme tedy, že postup je přímočarý.

21 Poznámka

22 Jde vlastně o aplikaci operace **mod** „přes indexy“.

24 Dotažení křivky do koncových bodů

25 Konstrukce křivky, kdy chceme „dotáhnout“ Coonsovou křivku do koncových bodů, vychází z její
26 konstrukce, kdy se vlastně vykresluje křivka „jen pod vnitřními řídícími body“. Postup je opět velmi
27 jednoduchý, neboť je pouze nutné krajní body vhodně „duplicovat“, viz následující sekvence:

- 28 • úvodní sekvence:
29 $[x_0, x_0, x_0, x_1]$ (křivka z bodu x_0 do $\frac{1}{2}$ hrany x_0x_1)
30 $[x_0, x_0, x_1, x_2]$ (křivka z $\frac{1}{2}$ hrany x_0x_1 do počátečního bodu standardně vykreslované křivky)
- 31 • standardní sekvence:
32 $[x_0, x_1, x_2, x_3], [x_1, x_2, x_3, x_4], \dots, [x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}]$
- 33 • „uzavírací“ sekvence analogická té úvodní:
34 $[x_{n-3}, x_{n-2}, x_{n-1}, x_{n-1}]$ a
35 $[x_{n-2}, x_{n-1}, x_{n-1}, x_{n-1}]$

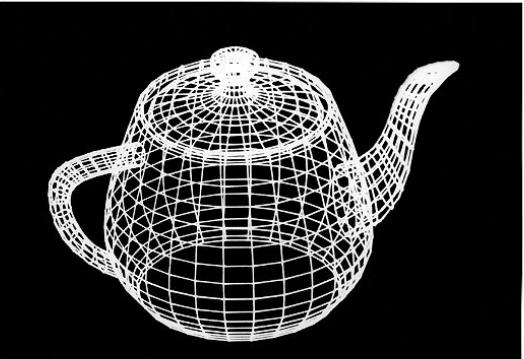
36 Zajisté je celá řada zajímavých problémů, např.::

- 36 • jak lze u křivky dané více segmenty redukovat počet segmentů s minimalizací chyby
- 37 • jak lze digitalizovanou křivku převést na parametrický tvar s minimalizací počtu segmentů
- 38 • jak lze reprezentovat křivky s nenulovým ryvem, tj. změny akcelerace

11.5. Parametrické plochy

Parametrické plochy umožňují realizaci hladkých ploch popsaných jednoduchým matematickým aparátem. Většinou jsou parametry plochy u, v omezeny na interval $u, v \in \langle 0,1 \rangle$ a pro vyváření složitějších tvarů se tyto plochy musejí napojovat. Proto se takové plochy označují termínem *pláty* (patches) a proces jejich napojování pak termínem plátování. Dnes již klasickou ukázkou je slavný The Utah Teapot (čajník z Utahu) Martina Newella, který je založen na Bézierově reprezentaci bikubického plátu.

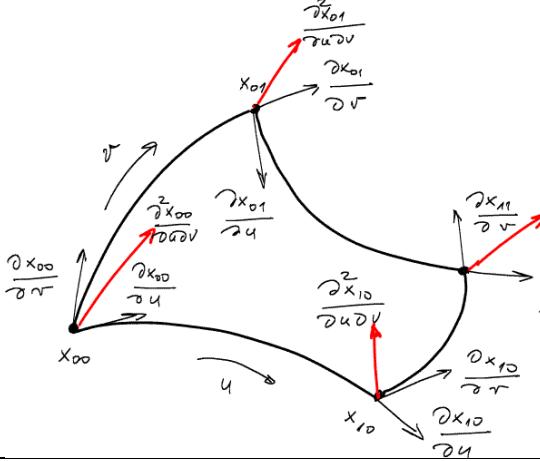
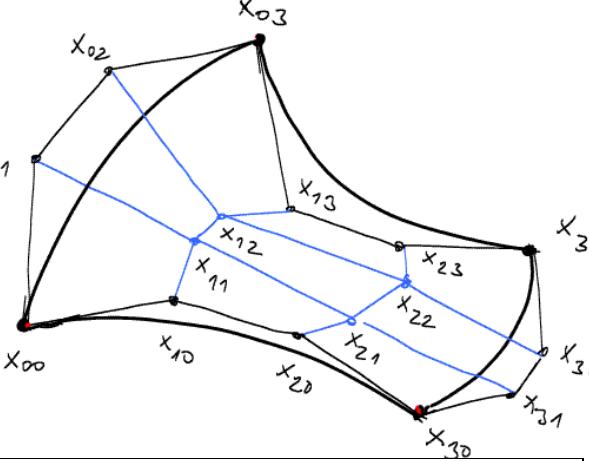
Čtyřúhelníkové pláty

 <small>University of Utah Computer Science</small>	
Obr.11.8: Teapot – drátěný model	Obr.11.9: Teapot – vystínovaný

Původní data viz <http://www.sibaker.org/teapot/teaset.tgz>

Java aplet viz <http://mrl.nyu.edu/~perlin/experiments/teapot/>

Parametrické bikubické pláty splňují podmínu, že pro $u = \text{konst}$ nebo $v = \text{konst}$ dostáváme kubické křivky dříve uvedené. Pro názornost uvedeme Hermitův a Bézierův bikubický plát:

	
Obr.11.10: Hermitův bikubický plát	Obr.11.11: Bézierův bikubický plát

Bikubické pláty můžeme popsat rovnicí:

$$P(u, v) = [u^3, u^2, u, 1] \mathbf{M}_F^T \mathbf{P} \mathbf{M}_F [v^3, v^2, v, 1]^T \quad u, v \in \langle 0,1 \rangle \quad (11.54)$$

kde: \mathbf{M}_F je matice příslušné formy, tj. Hermitovy, Bézierovy a Coonsovy, \mathbf{P} je matice 4×4 řídících hodnot pro každou souřadnici, tj. x, y, z , a $u, v \in \langle 0,1 \rangle$ jsou parametry bikubického plátu.

$\mathbf{P}_H = \begin{bmatrix} [P_{00} & P_{01}] & \frac{\partial}{\partial v} [P_{00} & P_{01}] \\ [P_{10} & P_{11}] & [P_{10} & P_{11}] \end{bmatrix}$	$\mathbf{P}_B = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}$
Matice řídících hodnot Hermitovy formy	Matice řídících hodnot Bézierovy formy
$\mathbf{P}_C = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}$	
Matice řídících hodnot Coonsovy formy	

1
2 V případě parametrických kubických křivek byla odvozena z podmínek jejich maticová forma, viz
3 kap.11.1 (Parametrické kubické křivky), kdy bod parametrické křivky $P(t) = (x, y, z)$ pro Hermitovu
4 formu je určen:

$$P(t) = [P_0, P_1, P'_0, P'_1] \mathbf{M}_H [t^3, t^2, t, 1]^T \quad t \in \langle 0,1 \rangle \quad (11.55)$$

5 kde \mathbf{M}_H je matice Hermitovy formy. Matice Hermitovy formy je definována

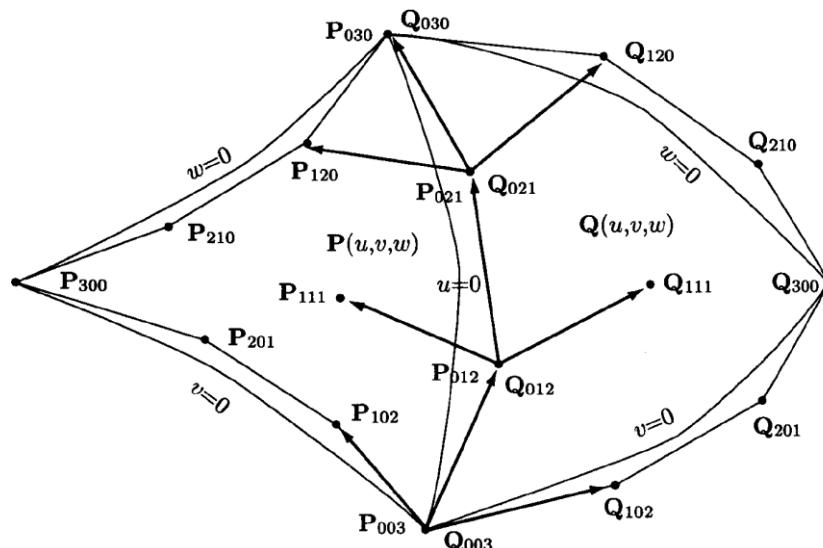
$$\mathbf{M}_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

6
7 **Trojúhelníkové pláty**
8 De Casteljau, který pracoval v Citroënu, vynil parametrický trojúhelníkový plát před plátem
9 čtyřúhelníkovým již v roce 1959. Kontrolní body jdou indexovány s indexy i, j, k přičemž platí, že
10 $0 \leq i, j, k \leq n$ a $i + j + k = n$ a body na plátu jsou určeny vztahem:

$$P(u, v, w) = \sum_{i+j+k=n} P_{ijk} \frac{n!}{i! j! k!} u^i v^j w^k \quad u + v + w = 1 \quad (11.56)$$

11 Hodnota n je určena uživatelem podle složitosti povrchu. V počítačové grafice je většinou $n = 3$. Plát
12 je vlastně funkci jen dvou parametrů, neboť platí podmínka $u + v + w = 1$ (je to vlastně analogie
13 barycentrických souřadnic, viz kap.10.1.(Lineární interpolace), podrobněji viz:

- 14 • Salomon,D.: Computer Graphics and Geometric Modeling, Springer, pp.483, 1999



Obr.11.12: Trojúhelníkový parametrický plát a jeho napojování.

15
16

1 Odvození rovnic bikubického plátu pro Hermitovu formu

2
3 V mnoha publikacích jsou jen uvedeny vzorce. Nicméně jejich odvození a motivace jsou velmi
4 důležité pro pochopení principů mnoha algoritmů. Při odvození rovnic pro parametrickou bikubickou
5 plochu v Hermitově formě budeme uvažovat pouze souřadnici $x(u, v)$, která je parametrizována nyní
6 dvěma parametry u a v , neboť jde nyní o plochu.

7 Základními požadavky na bikubickou parametrickou plochu v Hermitově formě jsou:

- 8 • pro parametry platí podmínka $u \in \langle 0,1 \rangle$ & $v \in \langle 0,1 \rangle$
- 9 • bikubický plát má 4 vrcholy
- 10 • všechny křivky pro $u \in \langle 0,1 \rangle$ & $v = konst$ a $v \in \langle 0,1 \rangle$ & $u = konst$ včetně křivek hraničních,
11 tj. pro $u = 0$ nebo $v = 0$, jsou také kubické křivky Hermitovy formy

12 Z popisu bikubického plátu $x(u, v)$ je zřejmé, že křivky pro $u = v$ a $u = -v$ jsou křivkami 6. stupně.
13 Hermitova křivka $x(u)$ pro danou hodnotu parametru v je dána vztahem:

$$x(u) = [x_0, x_1, x'_0, x'_1] M_H [u^3, u^2, u, 1]^T \quad u \in \langle 0,1 \rangle \quad (11.57)$$

14 nebo při použití transpozice:

$$x(u) = [u^3, u^2, u, 1] M_H^T [x_0, x_1, x'_0, x'_1]^T \quad u \in \langle 0,1 \rangle \quad (11.58)$$

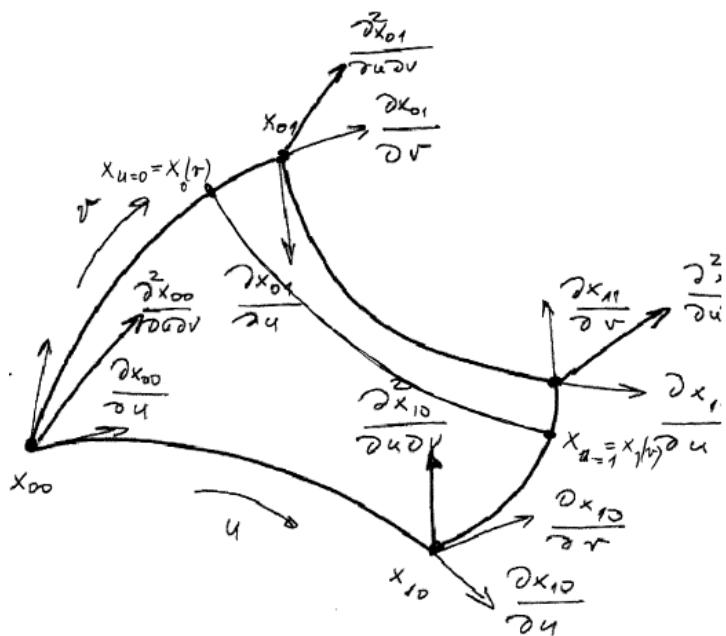
15 Křivka $x(u)$ je určena koncovými body x_0, x_1 a derivacemi x'_0, x'_1 , přičemž $x' = \frac{dx}{du}$, tj. derivace podle
16 u . V případě bikubického plátu pak tyto derivace jsou derivace parciální podle u , tj. $x^{(u)} = \frac{\partial x}{\partial u}$.

17 V případě bikubického plátu koncové body křivek $x(u)$ a jejich derivace $x'(u)$, tj. řídicí hodnoty
18 křivky, jsou parametrizovány parametrem v a tedy můžeme formálně psát:

$$x(u, v) = [u^3, u^2, u, 1] M_H^T [x_{u=0}(v), x_{u=1}(v), x_{u=0}^{(v)}(v), x_{u=1}^{(v)}(v)]^T \quad v \in \langle 0,1 \rangle \quad (11.59)$$

20 přičemž křivky $x_{u=0}(v), x_{u=1}(v), x_{u=0}^{(v)}(v), x_{u=1}^{(v)}(v)$ jsou opět kubické křivky v Hermitově formě.

21 V následujícím bude použito značení $x^{(u)} = \frac{\partial x}{\partial u}, x^{(v)} = \frac{\partial x}{\partial v}$ a $x^{(u,v)} = \frac{\partial^2 x}{\partial u \partial v}$.



22
23

Obr.11.13:: Hermitův bikubický plát

1 Také můžeme psát pro:

- 2 • pro $u = 0$

$$x_{u=0}(v) = [x_{00}, x_{01}, x_{00}^{(v)}, x_{01}^{(v)}] \mathbf{M}_H [v^3, v^2, v, 1]^T \quad (11.60)$$

- 3 • pro $u = 1$

$$x_{u=1}(v) = [x_{10}, x_{11}, x_{10}^{(v)}, x_{11}^{(v)}] \mathbf{M}_H [v^3, v^2, v, 1]^T \quad (11.61)$$

4 Nyní je nutné vyjádřit ještě parciální derivace pro křivky $x_{u=0}^{(u)}(v), x_{u=1}^{(u)}(v)$, tj. tečné vektory. Lze
5 nahlédnut, že:

- 6 • pro $u = 0$

$$x_{u=0}^{(u)}(v) = [x_{00}^{(u)}, x_{01}^{(u)}, x_{00}^{(u,v)}, x_{01}^{(u,v)}] \mathbf{M}_H [v^3, v^2, v, 1]^T \quad (11.62)$$

- 7 • pro $u = 1$

$$x_{u=1}^{(u)}(v) = [x_{10}^{(u)}, x_{11}^{(u)}, x_{10}^{(u,v)}, x_{11}^{(u,v)}] \mathbf{M}_H [v^3, v^2, v, 1]^T \quad (11.63)$$

8 Parametrickou bikubickou plochu v Hermitově formě lze tedy zapsat ve tvaru:

$$x(u, v) = [u^3, u^2, u, 1] \mathbf{M}_H^T [x_{u=0}(v), x_{u=1}(v), x_{u=0}^{(u)}(v), x_{u=1}^{(u)}(v)]^T \quad u, v \in \langle 0, 1 \rangle \quad (11.64)$$

9 resp.:

$$x(u, v) = [u^3, u^2, u, 1] \mathbf{M}_H^T \mathbf{X}_H \mathbf{M}_H [v^3, v^2, v, 1]^T \quad u, v \in \langle 0, 1 \rangle \quad (11.65)$$

10

11 kde \mathbf{X}_H je matice řídících hodnot Hermitovy formy:

$$\mathbf{X}_H = \begin{bmatrix} x_{00} & x_{01} & x_{00}^{(v)} & x_{01}^{(v)} \\ x_{10} & x_{11} & x_{10}^{(v)} & x_{11}^{(v)} \\ x_{00}^{(u)} & x_{01}^{(u)} & x_{00}^{(u,v)} & x_{01}^{(u,v)} \\ x_{10}^{(u)} & x_{11}^{(u)} & x_{10}^{(u,v)} & x_{11}^{(u,v)} \end{bmatrix} = \begin{bmatrix} \Sigma & \frac{\partial \Sigma}{\partial v} \\ \frac{\partial \Sigma}{\partial u} & \frac{\partial^2 \Sigma}{\partial u \partial v} \end{bmatrix} \quad (11.66)$$

12 kde matice Σ je definována $\Sigma = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}$

13 Matice $\frac{\partial^2 \Sigma}{\partial u \partial v}$ je vlastně maticí zkrutů (Twist). Analogicky pak pro x a y , neboť pro $\langle x, y, z \rangle$ jde vlastně
14 o vektor zkrutů v jednotlivých vrcholech Hermitova plánu, tj. $\left[\frac{\partial^2 x}{\partial u \partial v}, \frac{\partial^2 y}{\partial u \partial v}, \frac{\partial^2 z}{\partial u \partial v} \right]^T$. Rovnici pro
15 bikubický parametrický plát v Hermitově formě pak dostaváme ve formě:

$$x(u, v) = \mathbf{u}^T \mathbf{M}_H^T \mathbf{X}_H \mathbf{M}_H \mathbf{v} \quad u, v \in \langle 0, 1 \rangle \quad (11.67)$$

16 kde: $\mathbf{u} = [u^3, u^2, u, 1]^T$ a $\mathbf{v} = [v^3, v^2, v, 1]^T$

17 Obdobně pro ostatní souřadnice, tj. y, z :

$$y(u, v) = \mathbf{u}^T \mathbf{M}_H^T \mathbf{Y}_H \mathbf{M}_H \mathbf{v} \quad z(u, v) = \mathbf{u}^T \mathbf{M}_H^T \mathbf{Z}_H \mathbf{M}_H \mathbf{v} \quad u, v \in \langle 0, 1 \rangle \quad (11.68)$$

18 Matice $\mathbf{X}_H, \mathbf{Y}_H, \mathbf{Z}_H$ jsou matice 4×4 řídících hodnot daného bikubického parametrického plánu.

19 Zkráceně pro reprezentaci bodu $P(u, v) = [x(u, v), y(u, v), z(u, v)]^T$ pak lze formálně psát:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_H^T \mathbf{P}_H \mathbf{M}_H \mathbf{v} \quad u, v \in \langle 0, 1 \rangle \quad (11.69)$$

20 a tedy:

$$\mathbf{P}_H = \begin{bmatrix} [P_{00} & x_{01}] & \frac{\partial}{\partial v} [P_{00} & P_{01}] \\ [P_{10} & x_{11}] & [P_{10} & P_{11}] \\ \frac{\partial}{\partial u} [P_{00} & P_{01}] & \frac{\partial^2}{\partial u \partial v} [P_{00} & P_{01}] \\ [P_{10} & P_{11}] & [P_{10} & P_{11}] \end{bmatrix} \quad (11.70)$$

21 Pro reprezentaci plochy jedním bikubickým plátem je tedy zapotřebí $3 \times 16 = 48$ hodnot.

22

23 V předchozím jsme ukázali, že převod mezi jednotlivými formami, tj. Hermite, Bézier a Coons, je dán
24 lineárním vztahem, viz kap.11.3 (Vzájemný převod kubických parametrických křivek).

1 **Bézierův bikubický plát**

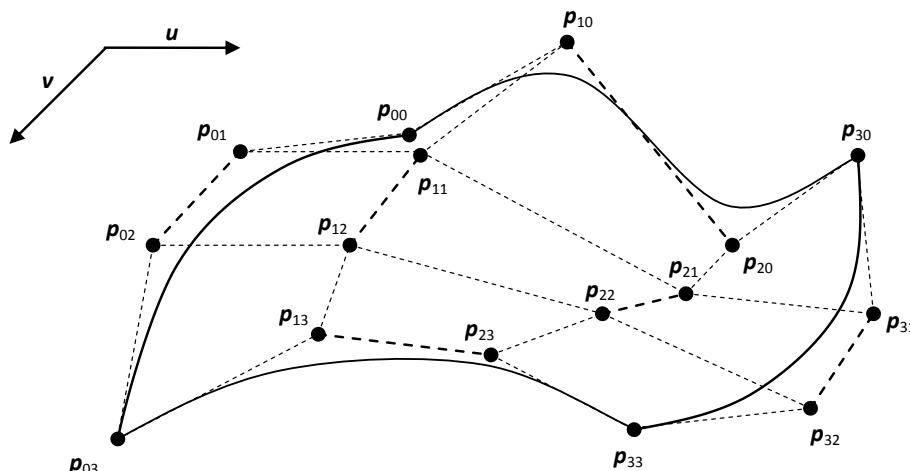
2 Bézierův bikubický plát je dán vztahem:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_B^T \mathbf{P}_B \mathbf{M}_B \mathbf{v} \quad (11.71)$$

3 přičemž matice řídících bodů pro Bézierovu formu je pak pro souřadnici x dána:

$$\mathbf{X}_B = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \quad (11.72)$$

4 a analogicky pro souřadnice y, z .



Obr.11.14: Bézierův čtyřúhelníkový plát

5 **Poznámka**

6 Béziérův plát je někdy popisován jako „tenzorový“ součin Bézierových křivek pro u a v takto:

$$P(u, v) = \sum_{k=0}^3 P_k(v) B_k^3(u) = \sum_{k=0}^3 \left(\sum_{i=0}^3 P_{i,k} B_i^3(v) \right) B_k^3(u) \quad (11.73)$$

9

10 **Coonsův bikubický plát**

11 Obdobně pro Coonsův plát dostaváme rovnici:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_C^T \mathbf{P}_C \mathbf{M}_C \mathbf{v} \quad (11.74)$$

12

13 **Závěr**

14 Je tedy zřejmé, že bikubický parametrický čtyřúhelníkový plát je popsán rovnicí:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_F^T \mathbf{P}_F \mathbf{M}_F \mathbf{v} \quad (11.75)$$

15

16 kde \mathbf{M}_F je matice příslušné formy a \mathbf{P}_F jsou matice řídících hodnot pro souřadnice x, y, z .

17

18 **Upozornění**

- Výhodou Bézierovy formy opět je, že bikubický plát je uzavřen v konvexním obalu řídících bodů Bézierova plátu, tj. v konvexním obalu 16 bodů v E^3 . Konvexní obálka však může mít méně než 16 bodů.
- U Coonsovy formy je dotažení plochy do hraničních křivek nebo „uzavření“ do uzavřené plochy realizováno analogicky jako u Coonsovy křivky. Je vhodné uvést, že tímto se vlastně vytváří hraniční plocha objemu daného objektu.

25

11.6. Bilineární a bikubický Coonsův plát

Bilineární Coonsův plát

V některých případech se používá bilineární Coonsův plát, který je dán okrajovými křivkami, a to:

$$\begin{aligned} \mathbf{a}_0(u) & v = 0 & \mathbf{a}_1(u) & v = 1 & u \in \langle 0,1 \rangle \\ \mathbf{b}_0(v) & u = 0 & \mathbf{b}_1(v) & u = 1 & v \in \langle 0,1 \rangle \end{aligned} \quad (11.76)$$

Pak bilineární Coonsův plát je definován takto:

$$\mathbf{Q}(u, v) = [1 - v \quad 1 \quad v] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{a}_0(u) & \mathbf{P}_{01} \\ \mathbf{b}_0(v) & \mathbf{P}(u, v) & \mathbf{b}_1(v) \\ \mathbf{P}_{10} & \mathbf{a}_1(u) & \mathbf{P}_{11} \end{bmatrix} \begin{bmatrix} 1 - u \\ 1 \\ u \end{bmatrix} \quad (11.77)$$

pro jednoduchost uveďme přímklovou plochu, kde hraniční křivky jsou přímkami. Pak

$$\begin{aligned} \mathbf{a}_0(u) &= (1 - u)\mathbf{P}_{00} + u\mathbf{P}_{01} & \mathbf{a}_1(u) &= (1 - u)\mathbf{P}_{10} + u\mathbf{P}_{11} \\ \mathbf{b}_0(v) &= (1 - v)\mathbf{P}_{00} + v\mathbf{P}_{10} & \mathbf{b}_1(v) &= (1 - v)\mathbf{P}_{01} + v\mathbf{P}_{11} \end{aligned} \quad (11.78)$$

Pak také křivky pro $u = \text{konst}$ nebo $v = \text{konst}$ jsou opět přímkami. Pak tedy pro $v = \text{konst}$ dostáváme:

$$\mathbf{P}(u, v) = (1 - v)\mathbf{a}_0(u) + v\mathbf{a}_1(u) \quad (11.79)$$

Pokud místo lineární interpolace pro u, v použijeme kubickou interpolaci, dostaneme obecnější typ, který je též někdy nazýván bikubickým plátem.

Bikubický Coonsův plát

Tento plát je dán vztahem

$$\mathbf{Q}(u, v) = [F_0(v) \quad 1 \quad F_1(v)] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{a}_0(u) & \mathbf{P}_{01} \\ \mathbf{b}_0(v) & \mathbf{P}(u, v) & \mathbf{b}_1(v) \\ \mathbf{P}_{10} & \mathbf{a}_1(u) & \mathbf{P}_{11} \end{bmatrix} \begin{bmatrix} F_0(u) \\ 1 \\ F_1(u) \end{bmatrix} \quad (11.80)$$

kde $F_0(t)$ a $F_1(t)$ jsou „blending“ funkce definované takto:

$$F_0(t) = 2t^3 - 3t^2 + 1 \quad F_0(t) = -2t^3 + 3t^2 \quad (11.81)$$

Jak bylo dosud ukázáno, matematický popis vede vlastně na bilineární formu a je tedy otázkou zda, lze jednotlivé parametrické pláty mezi sebou převádět a jak.

11.7. Vzájemný převod bikubických plátů

Převod z Hermitovy formy do Bézierovy je dán vztahem:

$$\mathbf{M}_{H \rightarrow B} = \mathbf{M}_H \mathbf{M}_B^{-1} \quad (11.82)$$

a tedy:

$$\mathbf{M}_{H \rightarrow B} \mathbf{M}_B = \mathbf{M}_H \quad (11.83)$$

Rovnici pro Hermitovu formu můžeme přepsat do tvaru pro Bézierovu formu:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_B^T \mathbf{M}_{H \rightarrow B}^T \mathbf{P}_H \mathbf{M}_{H \rightarrow B} \mathbf{M}_B \mathbf{v} = \mathbf{u}^T \mathbf{M}_B^T \mathbf{P}_B \mathbf{M}_B \mathbf{v} \quad (11.84)$$

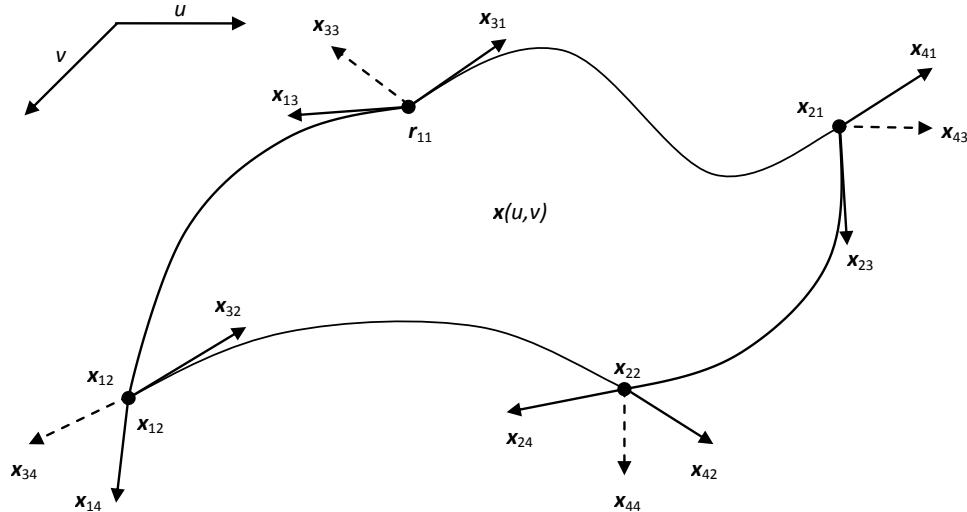
kde: $\mathbf{P}_B = \mathbf{M}_{H \rightarrow B}^T \mathbf{P}_H \mathbf{M}_{H \rightarrow B}$ je matice řídících bodů Bézierovy formy a \mathbf{M}_B je matice Bézierovy formy.

Příklad 1

V praxi je častý požadavek na převod Hermitovy formy plátu do formy Bézierovy. Lze ukázat, že převod z Hermitovy formy do Bézierovy formy je určen pro souřadnice x :

$$\begin{aligned} \mathbf{X}_B &= \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \\ &= \begin{bmatrix} x_{11} & x_{11} + \frac{1}{3}x_{13} & x_{12} - \frac{1}{3}x_{14} & x_{12} \\ x_{11} + \frac{1}{3}x_{31} & x_{11} + \frac{1}{3}(x_{13} + x_{31}) + \frac{1}{9}x_{33} & x_{12} + \frac{1}{3}(x_{32} - x_{14}) - \frac{1}{9}x_{34} & x_{12} + \frac{1}{3}x_{32} \\ x_{21} - \frac{1}{3}x_{41} & x_{21} + \frac{1}{3}(x_{23} - x_{41}) - \frac{1}{9}x_{43} & x_{22} - \frac{1}{3}(x_{24} + x_{42}) + \frac{1}{9}x_{44} & x_{22} - \frac{1}{3}x_{42} \\ x_{21} & x_{21} + \frac{1}{3}x_{23} & x_{22} - \frac{1}{3}x_{24} & x_{22} \end{bmatrix} \end{aligned} \quad (11.85)$$

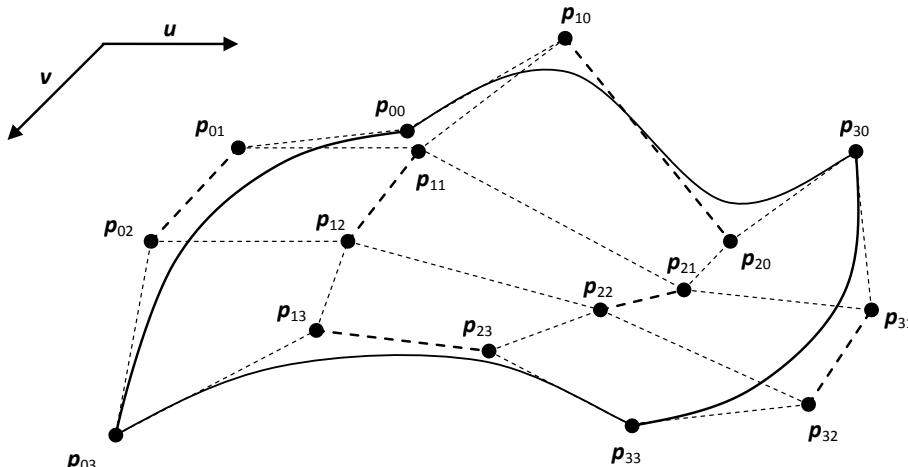
pokud použijeme indexaci, viz Obr.11.15. Pro ostatní souřadnice y a z postupujeme obdobně.



Obr.11.15: Řídící body Hermitova plátu

1 **Příklad 2**

2 Opačný převod, tj. převod z Bézierovy formy do Hermitovy formy, viz Obr.11.16, je určen:



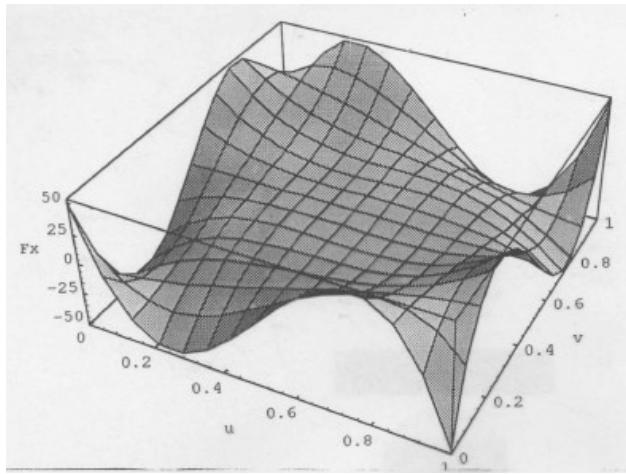
Obr.11.16:: Indexace Bézierovy plochy

3 Pak:

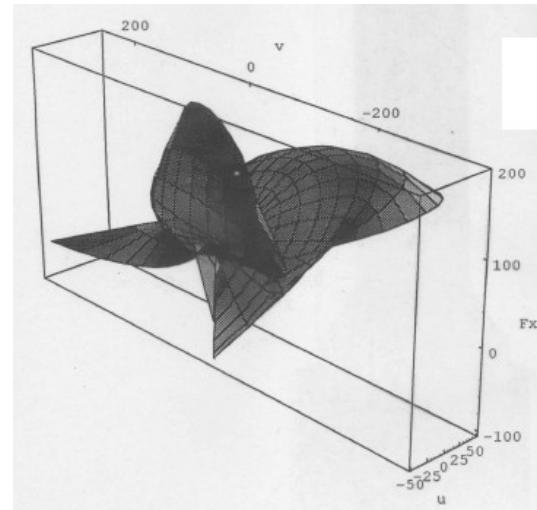
$$\begin{aligned}
 X_H &= \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \\
 &= \begin{bmatrix} x_{00} & x_{03} & 3(x_{01} - x_{00}) & 3(x_{03} - x_{02}) \\ x_{30} & x_{33} & 3(x_{31} - x_{30}) & 3(x_{33} - x_{32}) \\ 3(x_{10} - x_{00}) & 3(x_{13} - x_{03}) & 9(x_{00} - x_{01} - x_{10} + x_{11}) & 9(x_{02} - x_{03} - x_{12} + x_{13}) \\ 3(x_{30} - x_{20}) & 3(x_{33} - x_{23}) & 9(x_{20} - x_{21} - x_{30} + x_{31}) & 9(x_{22} - x_{23} - x_{32} + x_{33}) \end{bmatrix} \quad (11.86)
 \end{aligned}$$

6 Obdobné převodní vztahy je možné odvodit i pro další parametrické pláty, viz:
7

- 8 • Skala,V., Ondracka,V.: BS-Patch: Constrained Bezier Parametric Patch, Trans.on Mathematics,
-
- 9 2013

10 Nyní je otázkou jak vykreslit bikubický čtyřúhelníkový plát, když standardně zpracovávaná
11 geometrická primitiva jsou vlastně jen bod, úsečka, trojúhelník a jejich odvozeniny.

Obr.11.17: Plocha v parametrickém prostoru

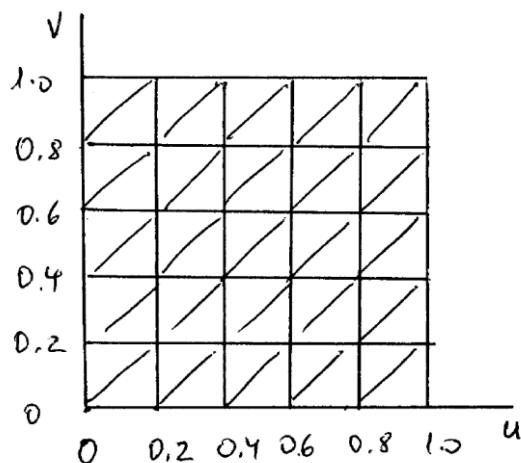


Obr.11.18: Plocha v geometrickém prostoru

12

11.8. Vykreslování parametrických ploch

Vykreslování kubických parametrických křivek pomocí lomené čáry bylo uvedeno v kap.11.2 (Vykreslování parametrických křivek). Pokud chceme vykreslit kružnici pomocí n-úhelníka, pak potřebujeme cca 100 vrcholů rovnoměrně rozložených na kružnici, abychom dostali vjem hladké křivky. Vykreslení bikubického plátu se většinou realizuje pomocí trojúhelníkové sítě, resp. množinou trojúhelníků, s normálovými vektory ve vrcholech. Parametrický prostor (u, v) se rozdělí na $n \times m$ obdélníků a každý obdélník se pak dále rozdělí na 2 trojúhelníky. To znamená, že pro dosažení vjemu hladkosti při vykreslování plátu budeme uvažovat $n, m \in \langle 50,100 \rangle$ a tedy počet generovaných bodů bude cca $\langle 2500, 10000 \rangle$ na jeden bikubický plát. Protože složité objekty se skládají z velkého počtu plátů, např. $10^2 - 10^4$ a více je nutné věnovat pozornost efektivitě generování souřadnic vrcholů trojúhelníků, které bikubický plát nahrazují. Poznamenejme, že volba úhlopříčky při dělení čtyřúhelníka je v zásadě libovolná.



Obr.11.19: Rozdělení intervalu $u, v \in \langle 0,1 \rangle$ na trojúhelníkovou síť pro $n, m = 5$

Uvažme opět rovnici bikubického plátu a pro jednoduchost pouze pro souřadnici x :

$$x(u, v) = \mathbf{u}^T \mathbf{M}_F^T \mathbf{X}_F \mathbf{M}_F \mathbf{v} \quad (11.87)$$

Je zřejmé, že hodnoty výrazu $\mathbf{M}_F^T \mathbf{X}_F \mathbf{M}_F$ jsou konstantní pro celý plát a mohou být předem uloženy do matice:

$$\mathbf{Q}_x = \mathbf{M}_F^T \mathbf{X}_F \mathbf{M}_F \quad (11.88)$$

Pak vlastní výpočet souřadnic vrcholů trojúhelníků nahrazující daný bikubický plát lze zapsat:

```

 $\mathbf{Q}_x = \mathbf{M}_F^T \mathbf{X}_F \mathbf{M}_F; \quad \mathbf{Q}_y = \mathbf{M}_F^T \mathbf{Y}_F \mathbf{M}_F; \quad \mathbf{Q}_z = \mathbf{M}_F^T \mathbf{Z}_F \mathbf{M}_F;$ 
 $u := 0;$ 
for  $i := 0$  to  $n$ 
  {  $\mathbf{g}_x := \mathbf{u}^T \mathbf{Q}_x; \quad \mathbf{g}_y := \mathbf{u}^T \mathbf{Q}_y; \quad \mathbf{g}_z := \mathbf{u}^T \mathbf{Q}_z;$  # Použít Hornerovo schéma #
     $v := 0;$ 
    for  $j := 0$  to  $m$ 
      {  $x := \mathbf{g}_x^T \mathbf{v}; \quad y := \mathbf{g}_y^T \mathbf{v}; \quad z := \mathbf{g}_z^T \mathbf{v};$  # Použít Hornerovo schéma #
        SAVE ( $x, y, z, i, j$ ); # uložení souřadnic vrcholů  $(x, y, z)$  do vhodné datové struktury #
      }
    }
  }
}

```

Ideový algoritmus pro výpočet bodů bikubického plátu

31

1 Vlastní vykreslení bikubického plánu lze realizovat:

- 2 • s konstantním stínováním, pak se zadává pouze normála trojúhelníka. Ta se určí např. jako
3 vektorový součin vektorů hran jednotlivých trojúhelníků
- 4 • pro stínování Gouraudovo nebo Phongovo je zapotřebí určit normálu v každém vrcholu
5 trojúhelníka. To je trochu obtížnější úloha, neboť:
- 6 ○ lze ji odhadnout jako průměr normál trojúhelníků sdílejících daný vrchol, nebo
- 7 ○ určit přesným výpočtem, a to vektorovým součinem derivací:

$$\begin{aligned} \mathbf{n} &= \left[\frac{\partial x(u, v)}{\partial u} \times \frac{\partial x(u, v)}{\partial v}, \frac{\partial y(u, v)}{\partial u} \times \frac{\partial y(u, v)}{\partial v}, \frac{\partial z(u, v)}{\partial u} \times \frac{\partial z(u, v)}{\partial v} \right]^T \\ &= \left[\frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} \right]^T \end{aligned} \quad (11.89)$$

8 Pokud máme bikubickou plochu zapsanou ve formě:

$$P(u, v) = \mathbf{u}^T \mathbf{M}_F^T \mathbf{P}_F \mathbf{M}_F \mathbf{v} \quad (11.90)$$

10 pak je zřejmé, že jednotlivé parciální derivace jsou dány:

$$\frac{\partial P(u, v)}{\partial u} = [3u^2, 2u, 1, 0] \mathbf{M}_F^T \mathbf{P}_F \mathbf{M}_F \mathbf{v} \quad \frac{\partial P(u, v)}{\partial v} = \mathbf{u}^T \mathbf{M}_F^T \mathbf{P}_F \mathbf{M}_F [3v^2, 2v, 1, 0]^T \quad (11.91)$$

11 Je nutné si uvědomit, že výpočet normály plánu je výpočetně náročný, neboť jde o funkci dvou
12 proměnných 5. stupně.

14

15

16 Poznámky

- 17 • Plát není obvykle vykreslován samostatně, ale spolu s dalšími pláty, které jsou na daný plát
18 napojeny. Proto je nutné brát v úvahu také sousední pláty při určování normálového vektoru
19 bodů ležících na hraničních křivkách, tj. pro $u = 0, u = 1$ a $v = 0, v = 1$.
- 20 • Výpočet normály může využít analytickou formu pro plát a počítat normálu přímo ze vzorce
21 pro daný bikubický plát.

22

23 Hladké napojování jednotlivých plátn, které je nutné k reprezentaci složitých povrchů, je netriviální
24 výpočetní operací a nazývá se *plátování*.

25

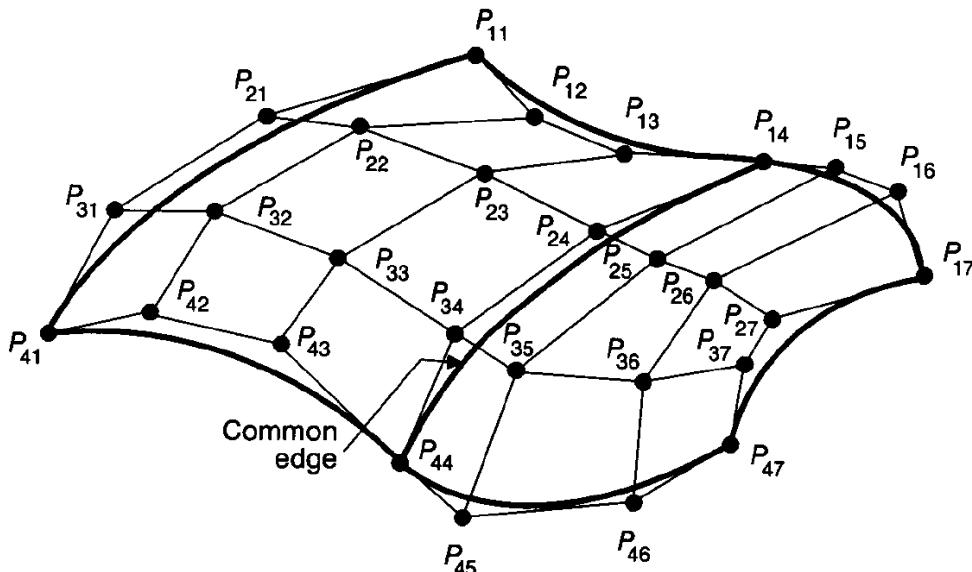
11.9. Hladké napojování

Uvažme jednoduchý případ napojení dvou bikubických plátů, viz Obr.11.20. Je zřejmé, že napojení musí splňovat určité podmínky, a to:

- pláty musejí mít společnou hranu, abychom nedostali „trhlinu“ na výsledném povrchu. To znamená, že hraniční sdílená křivka musí být identická – dostáváme tedy jednu omezující podmínsku
- další omezující podmínka je dána požadavkem hladkosti v místě napojení. Uvažme nyní pro jednoduchost Hermitovu formu. Pak je zřejmé, že na společné křivce, tj. v místě napojení, tangenciální vektory zleva a zprava musejí mít stejný směr, tj.:

$$\frac{\partial^1 P(u, v)}{\partial u} \Big|_{u=1} = q \frac{\partial^2 P(u, v)}{\partial u} \Big|_{u=0} \quad (11.92)$$

kde q je nenulová konstanta.



Obr.11.20: Napojení dvou bikubických plátů – Bézierova forma

Je nutné zdůraznit, že toto schéma napojování je platné v případech, kdy rohové body plátů jsou sdíleny 4 pláty. Pokud je bod, tj. „roh plátu“, sdílen 3 nebo 4 a více pláty jde o složitější výpočetní postup, kdy je nutné vlastně dodržet podmínku, že normálový vektor má stejný směr v daném bodě pro všechny pláty sdílející daný bod:

$$\frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} \Big|_k = q_i \left(\frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} \Big|_i \right) \quad (11.93)$$

kde q_i jsou nenulové libovolné konstanty a k a i jsou indexy plátů sdílející daný bod, přičemž hodnoty u, v jsou odpovídající pozici bodu v plátu, tj. $u, v \in \{0,1\}$.

Podrobnější informace jsou dostupné v příslušné odborné literatuře.

11.10. Výhody a nevýhody parametrické reprezentace

2 Nyní je asi vhodné porovnat výhody a nevýhody parametrické reprezentace vůči ostatním.

Výhody	Implicitní $F(\mathbf{x}) = 0$	<ul style="list-style-type: none"> • snadná klasifikace bodů-pozice (bod uvnitř-vně) • snadná reprezentace průsečíků, interference, množinové operace • jednoduchá reprezentace uzavřených křivek a ploch, multi-hodnotové křivky, „nekonečná“ směrnice
	Explicitní $y = f(\mathbf{x})$	<ul style="list-style-type: none"> • snadnost trasování • jednoduchý výpočet funkční hodnot
	Parametrická $\mathbf{x} = \mathbf{x}(u)$ $\mathbf{x} = \mathbf{x}(u, v)$	<ul style="list-style-type: none"> • snadnost transformací (nezávislost na souřadných osách) • snadná manipulace a formování tvarů – „free form shapes“ • jednoduchá generace složených křivek
Nevýhody	Implicitní	<ul style="list-style-type: none"> • obtížná manipulace a formování tvarů – „free form shapes“ • závislost na osách souřadného systému • složité trasování křivky, extrakce povrchu
	Explicitní	<ul style="list-style-type: none"> • závislost na osách souřadného systému • složitá reprezentace uzavřených křivek a ploch, multihodnotových křivek apod. • reprezentace „nekonečné“ směrnice pro polynomiální funkce
	Parametrická	<ul style="list-style-type: none"> • vysoká flexibilita podstatně komplikuje výpočet průsečíků, např. přímka-parametrická křivka, rovina-bikubická plocha atd.

- Odkazy:
- Bézier,P.: The mathematical basis of the UNISURF CAD system, Butterworths, 1986
 - Foley,J.D., van Dam,A., Feiner,S.K., Hughes,J.F: Computer Graphics: Principles and Practice, Addison Wesley Publ.Comp., 1996
 - Farin,G., Hoschek,J., Kim,M.-S. (Ed.): Handbook of Computer Aided Geometric Design, North Holland, 2002
 - Farin,G.E.: NURBS: From Projective Geometry to Practical Use, A.K.Peters, 1999
 - Boehm,W., Prautzsch,H.: Geometric Concept for Geometric Design, A.K.Peters, 1994
 - Warren,J., Weimer,H.: Subdivision Methods for geometric Design, Morgan Kaufmann Publ., 2002

12. Algoritmy řešení viditelnosti

Problém řešení viditelnosti je úlohou eliminace těch částí scény, které nejsou z dané pozice pozorovatele viditelné. Algoritmy pro řešení viditelnosti se dají rozdělit na:

- algoritmy eliminace neviditelných hran (Hidden Line Removal – HLR)
- algoritmy eliminace neviditelných ploch (Hidden Surface Removal – HRS)
- algoritmy eliminace neviditelných vrstevnic (Hidden Contour Removal - HCR) – jde vlastně o kombinaci eliminace neviditelných ploch a daných prostorových křivek, většinou vrstevnic.

Studiem problému řešení viditelnosti a eliminace neviditelných částí došlo k vývoji mnoha algoritmů, které mají své výhody a nevýhody a lze je v zásadě rozdělit na dvě skupiny, a to:

- algoritmy řešící viditelnost v prostoru objektů, tzv. Object Space Algorithms – např. Robertnův algoritmus, viz:

Skala,V.: Algoritmy počítačové grafiky II, 2011 ([CLICK off-line](#))

Algoritmy určují nové průsečíky, hrany atd. v přesnosti daného CPU, výsledek se dá libovolně zvětšovat atd. bez dalšího výpočtu viditelnosti

- algoritmy řešící viditelnost v obraze, tzv. Image Space Algorithms, viz např. z-buffer kap.12.3 (Hloubkový buffer – z-buffer) nebo algoritmus sledování primárního paprsku kap.15.2 (Základní algoritmus Ray-tracing).

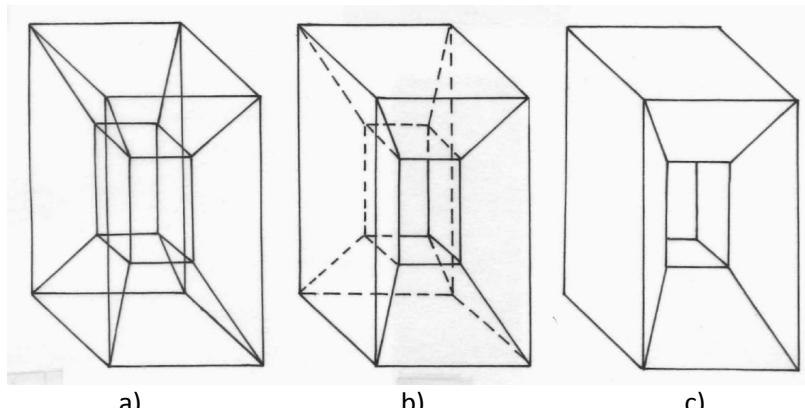
přesnost výpočtu a zobrazení je dána rozlišením generovaného obrazu, při zvětšení obecně dochází k novému řešení viditelnosti

Složitost algoritmů řešících viditelnosti v prostoru je obecně $O(k \lg k)$ nebo $O(k^2)$, kde: k je počet objektů ve scéně. Obecně lze asi říci, že tyto algoritmy jsou vhodnější pro scény s malým počtem objektů. Algoritmy operující nad obrazem jsou složitosti $O(nmk)$, kde: k je počet objektů ve scéně a nm je rozměr obrazu, tj. $n \times m$, resp. hodnota nm reprezentuje počet pixelů zobrazeného objektu. Jedním z prvních algoritmů eliminace neviditelných povrchů byl Warnockův algoritmus, který je v podstatě založena na dělení prostoru a quadtree.

Podrobněji viz:

- NewmanW.M., Sproull,R.F.: Principles of Interactive Computer Graphics, McGraw-Hill, 1973
- Skala,V.: Algoritmy počítačové grafiky II, 2011 ([CLICK off-line](#))

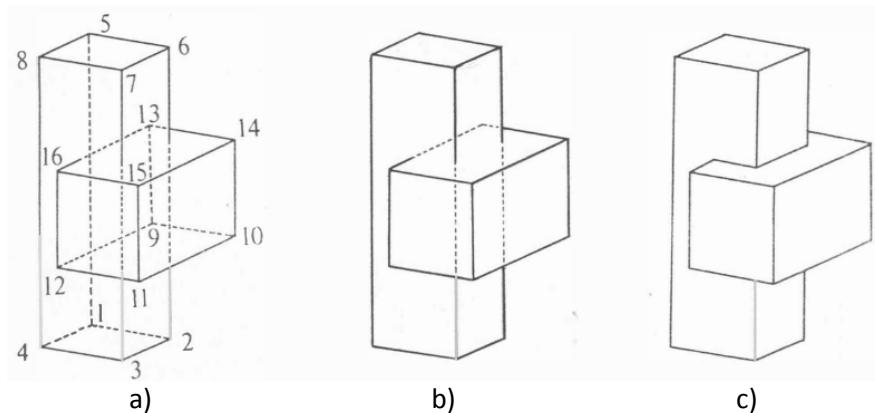
V současné době převážná většina aplikací používá "brutální" algoritmus *z-buffer*, který je založen na porovnávání vzdáleností, viz kap.12.3 (Hloubkový buffer – z-buffer). V mnoha aplikacích, zejména pak při kreslení čárových kreseb, nelze tento algoritmus použít. Pro řešení viditelnosti nestačí mít jen hranový (drátěný) model pro jednotlivé objekty, viz kap.7.3 (Reprezentace třírozměrných objektů a scén), neboť hranová reprezentace není jednoznačná, viz Obr.12.1.



Obr.12.1:: Drátěný model a jedna z možných interpretací

1
2
3
4

Situace však může být komplikovanější, neboť plochy se mohou navzájem protínat, viz Obr.12.2, a je nutné určit nejen průsečíky hran s plochou, ale také určit hrany nové, vzniklé průsečíkem ploch.



Obr.12.2: Průsečíky hran a ploch při řešení viditelnosti

5
6
7
8

12.1. Odstranění „zadních“ stěn

Řešení viditelnosti lze v případě konvexního tělesa kompletně vyřešit odstraněním ploch, které jsou „zadní“ a které jsou vždy zakryty viditelnými „předními“ plochami. Tato technika je použitelná i pro urychlení i pro eliminaci ploch u nekonvexních objektů. Předpokladem je, že plochy jsou orientované tak, že normála plochy v každém bodě směruje ven z tělesa (případně opačně) konzistentně pro všechna tělesa ve scéně. Tento proces se nazývá *Back-Face Removal* nebo *culling*. Řešení je poměrně jednoduché, neboť:

$$\mathbf{s}^T \mathbf{n} = \begin{cases} \leq 0 & \text{pak bod je viditelný} \\ > 0 & \text{pak bod je neviditelný} \end{cases}$$

kde: \mathbf{s} je směrový vektor daný zobrazovaným bodem a pozicí pozorovatele, \mathbf{n} je normálový vektor plochy v daném bodě. Je zřejmé, že pro planární plochy, např. trojúhelník je tento test platný pro všechny body dané plochy. V tomto případě pak lze přijmout nebo odmítnout celý trojúhelník.

13

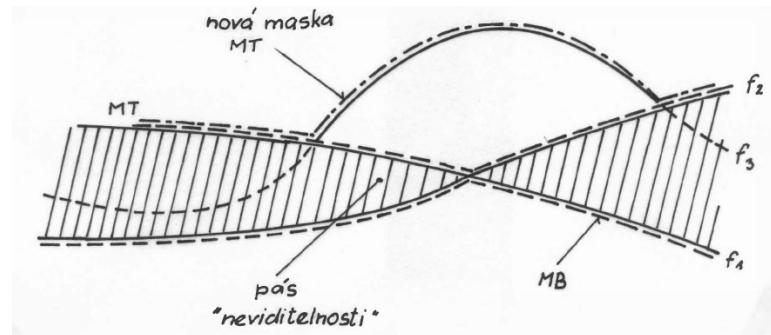
14

15

16

12.2. Řešení viditelnosti pro plochy 2½ D - $z=f(x,y)$

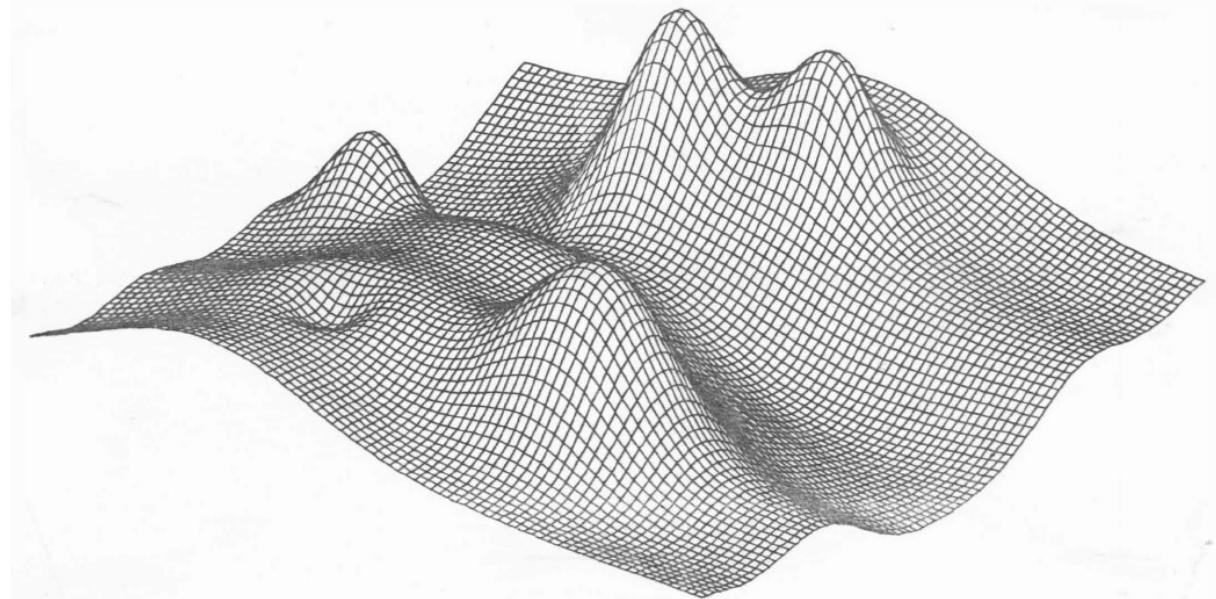
Specifickou úlohou je eliminace neviditelných částí funkcí dvou proměnných. Algoritmus, který je též nazýván algoritmem plovoucího horizontu, je založen na tom, že se kreslí postupně řezy od pozorovatele směrem dozadu. Pro jednoduchost si představme, že povrch je dán jako funkce $z = f(x, y)$. Označme nejbližší řez f_1 a další vzdálenější řez f_2 . Pak tyto křivky vlastně definují plochu, kterou pozorovatel vidí z podhledu, nebo nadhledu, kterou označíme jako *pás neviditelnosti*. Při kreslení dalšího vzdálenějšího řezu, který je označen jako f_3 se pak kreslí jen ty části této křivky, které jsou nad nebo pod pásem neviditelnosti.



Obr.12.3: Algoritmus plovoucího horizontu

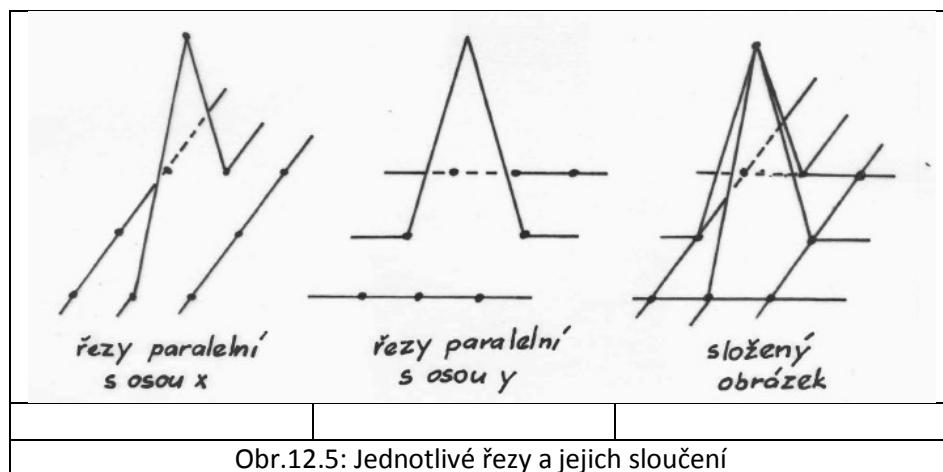
Po nakreslení viditelných částí řezu f_3 se horní a dolní horizont odpovídajícím způsobem upraví, takže se pás neviditelnosti výškově rozšíří. Takto se postupuje i pro další řezy.

Je zřejmé, že tento algoritmus je zejména výhodný, pokud jsou hodnoty x a y v obdélníkové síti. Datová struktura pak může obsahovat pouze funkční hodnoty a x_{min} , x_{max} , y_{min} a y_{max} v případě pravidelné sítě, v případě nepravidelné obdélníkové sítě jsou zapotřebí ještě hodnoty na osách x a y .



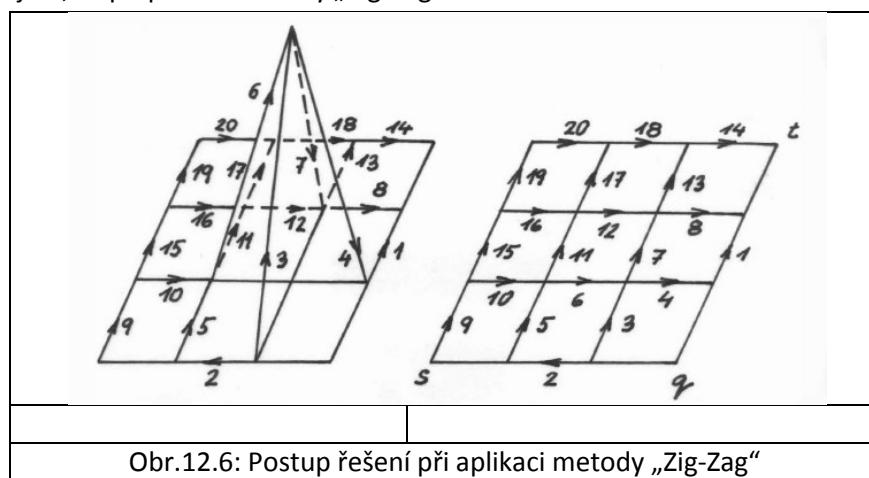
Obr.12.4: Vykreslená plocha algoritmem plovoucího horizontu

Protože se skládají nezávisle řezy pro x a y dohromady, vznikají nekorektní řešení viditelnosti, viz Obr.12.5. Pro realizaci algoritmu se proto používá vykreslování postupem „Zig-Zag“, který je založen na vykreslování části řezů, které jsou nejbližše pozorovateli, viz Obr.12.6. Algoritmus lze realizovat přímo v Bresenhamově algoritmu pro kreslení úsečky, viz kap. 19.2 (Bresenhamův algoritmus a algoritmus plovoucího horizontu).



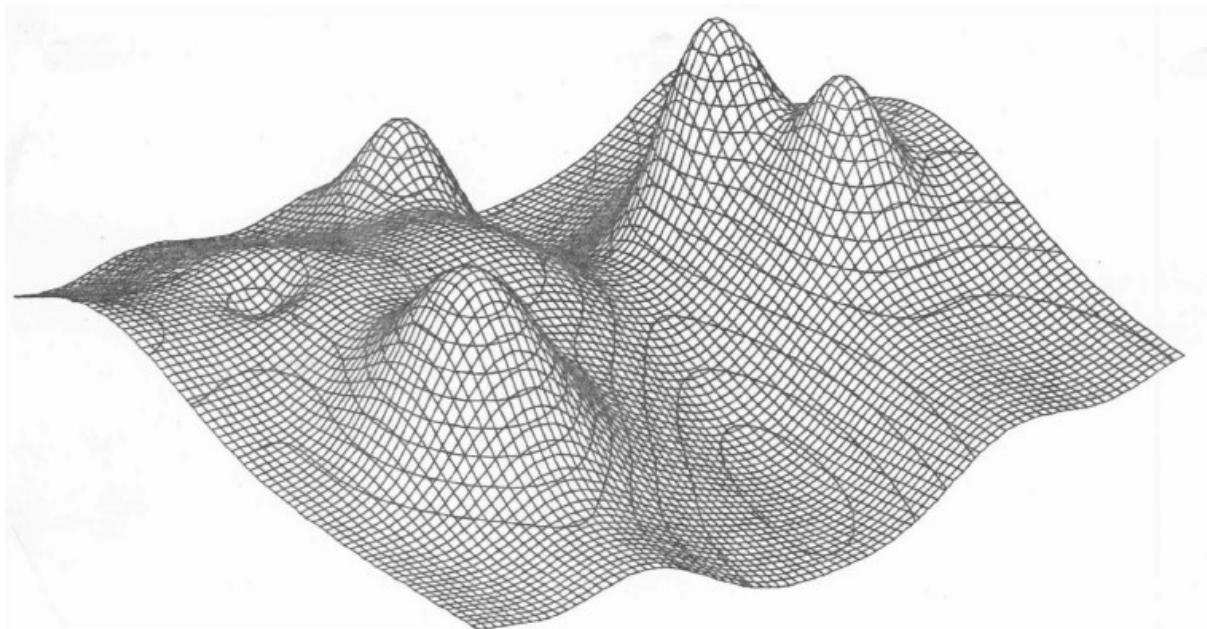
Obr.12.5: Jednotlivé řezy a jejich sloučení

- 1 Z Obr.12.6 je zřejmé, že při použití metody „Zig-Zag“ dochází ke korektnímu řešení viditelnosti.



Obr.12.6: Postup řešení při aplikaci metody „Zig-Zag“

- 2
 3 Metoda plovoucího horizontu je snadno modifikovatelná i pro kreslení vrstevnic s respektováním
 4 viditelnosti, viz Obr.12.7. Podrobnější informace lze nalézt např. v:
 5 • Skala,V.: Algoritmy počítačové grafiky II, 2011 ([CLICK off-line](#)).



Obr.12.7: Modifikace algoritmu plovoucího horizontu pro kreslení vrstevnic

6
 7
 8

12.3. Hloubkový buffer - z-buffer

2 Jedním z nejjednodušších algoritmů eliminace neviditelných povrchů je hloubkový buffer (depth
 3 buffer nebo z-buffer). Je založen na tom, že pro každý pixel je dána hloubková mapa $d[x, y]$, v které je
 4 zaznamenána nejmenší vzdálenost dosud vykreslované části plochy, tj. pixelu, na dané pozici $[x, y]$.
 5 Při rasterizaci povrchu geometrického objektu se pak porovnává vzdálenost právě vykreslovaného
 6 bodu s nejmenší vzdáleností z dosud kreslených bodů na dané pozici. Pokud nový bod je méně
 7 vzdálený, pak se bod vykreslí a hloubková mapa se pro danou pozici aktualizuje.

```

8
9   for každou plochu  $F$ 
10    for každý pixel  $(x, y)$  plochy  $F$ 
11      depth := hloubka v pozici  $(x, y)$ ;
12      if depth <  $d[x, y]$  /*  $F$  je nejbližše z dosud kreslených na dané pozici  $(x, y)$  */
13      {
14        color  $[x, y]$  := barva  $F$  na pozici  $(x, y)$ ;
15         $d[x, y]$  := depth
16      }
17

```

Algoritmus z-buffer

18 Algoritmus hloubkového bufferu je extrémně jednoduchý a ideální pro hardwarovou realizaci.
 19 Připomeňme, že po projekci není k dispozici vzdálenost, ale pseudo-vzdálenost, která je monotónní
 20 funkcí vzdálenosti.

21 Algoritmus hloubkového bufferu má navíc ještě další výhody, a to:

- 22 • plochy mohou být kresleny v libovolném pořadí, tj. není zapotřebí je uspořádat např. jako
 v algoritmu malíře, postup a způsob kreslení objektu je libovolný
- 23 • vzhledem k uspořádání v rovině (x, y) lze s výhodou využít hloubkové koherence, tj. hloubka
 se příliš nemění pro sousední pixely
- 24 • i když je důraz na zobrazování trojúhelníkových ploch, hloubkový buffer je schopen efektivně
 řešit i jiné typy, např. kvadratické a bikubické plochy apod.
- 25 • pokud uložíme pro danou scénu i hloubkovou mapu, lze při přidávání nových objektů
 původní mapu využít a řešit snadno a rychle viditelnost.

31 Nicméně vedle výhod má i jisté nevýhody, a to:

- 32 • může překreslovat daný pixel mnohonásobně, např. při zobrazování parametrických ploch
 atd., a počítá se výpočetně náročné osvětlení atd. Proto některé renderery provedou
 částečné setřídění ploch tak, aby se kreslily nejdříve plochy nejbližší. Tím se ušetří na výpočtu
 světelných poměrů.
- 33 • hloubkový buffer vyžaduje hodně paměti a každá buňka hloubkového bufferu má daný počet
 bitů, čímž je dána rozlišitelnost hloubkové mapy. To znamená, že v některých případech
 vlivem nepřesnosti reprezentace hloubky může docházet ke špatnému určení viditelnosti,
 zejména v oblasti blízké zadní stěny pohledové pyramidy.

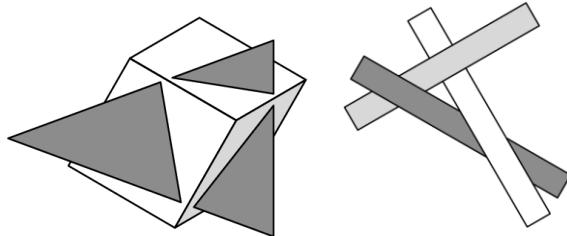
41

42

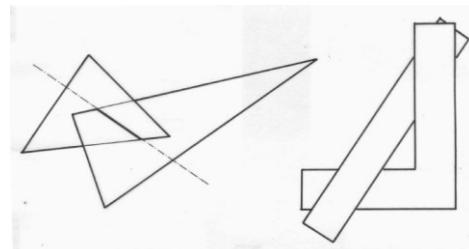
12.4. Algoritmus Malíře

2 Algoritmus malíře je typickým algoritmem řešení viditelnosti v prostoru. Princip je velmi jednoduchý
 3 a je založen na uspořádání ploch vzhledem ke vzdálenosti od pozorovatele a to tak, aby při
 4 postupném plošném překreslování odzadu dopředu se ve výsledku zobrazila daná scéna korektně.
 5 Nicméně tento zdánlivě jednoduchý princip má několik úskalí a to:

- 6 • jednotlivé plochy se nesmí vzájemně protínat, viz Obr.12.8.a
- 7 • plochy se nemohou vzájemně zakrývat, viz Obr.12.8.b
- 8 • Je zřejmé, že algoritmus pro uspořádání ploch vzhledem k hloubce bude složitější než běžné
 9 algoritmy pro uspořádání jako je *heap sort* nebo *quick sort*.



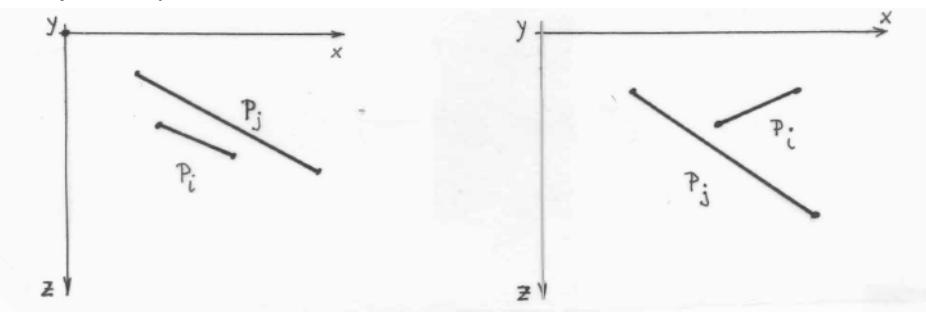
Obr.12.8: Vzájemná poloha ploch



Obr.12.9: Protínání a zakrývání ploch

10 Kritériem pro uspořádání je porovnání z-souřadnic a výsledek testu překrytí, tzv. *overlap test*. Pokud
 11 se plochy vzájemně protínají nebo překrývají, je nutné dané plochy vhodně rozdělit. V případě
 12 protínajících se ploch pak následně dochází ke generaci nových vrcholů a tím i nových ploch. Zejména
 13 pak na průsečnici dvou trojúhelníkových sítí dochází k „explozi“ malých trojúhelníků, viz Obr.12.9.a.
 14 Obr.12.9.b ukazuje nejjednodušší případ vzájemného překrytí dvou rovinných ploch, kdy je nutné
 15 jednu plochu rozdělit. V praxi jsou n-úhelníky realizovány pomocí trojúhelníků, přičemž „vnitřní“
 16 hrany nejsou zobrazovány, takže rozdělení na nezakrývající se plochy není obtížné, neboť dva
 17 trojúhelníky se nemohou zakrývat navzájem.

18
 19 Je zřejmé, že bude docházet k numerické nestabilitě výpočtu průsečíků zejména, pokud budou
 20 protínající se trojúhelníky téměř rovnoběžné a malé.



Obr.12.10: Vzájemná poloha ploch pro řešení viditelnosti

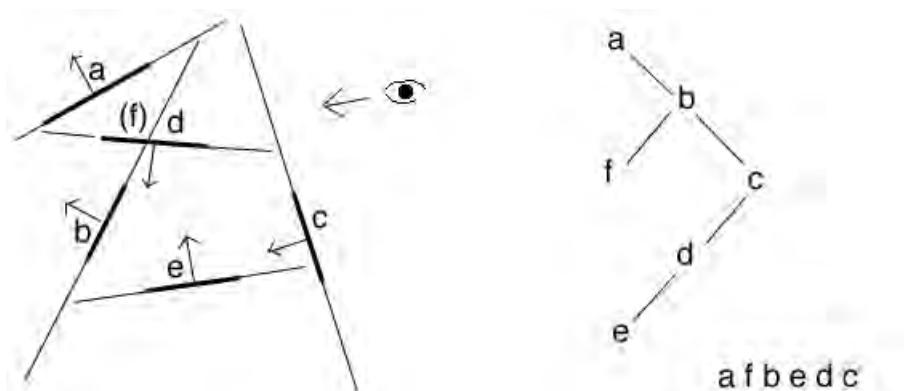
22 Pro uspořádání ploch ve směru osy z je nutné respektovat tzv. *overlap test*, který odliší případy
 23 Obr.12.10.a od případu Obr.12.10.b.

25

12.5. BSP strom

V předchozí části byly vysvětleny některé základní metody eliminace neviditelných částí. Jednou z elegantních metod řešení problému viditelnosti jsou BSP stromy (Binary-space Partition Trees). Metoda BSP stromů je založena na předzpracování ploch do binárního stromu tak, že některé plochy jsou vhodně rozděleny tak, aby pořadí jejich zobrazování již nebylo závislé na pozici pozorovatele. Je tedy zřejmé, že přímá aplikace v algoritmu malíře je jednoduchá. Konstrukce BSP stromu není obecně závislá na dimenzi řešené úlohy, a proto je aplikovatelná i pro řešení jiných problémů.

Pro jednoduchost uvažme jednoduchou scénu v E^2 . Objekty a, b, c, d, e ve scéně rozdělíme tak, aby žádná přímka, na které úsečka leží, neprotínala jiný objekt. Úsečka d se nám tedy rozdělí na d a f . Následně se pak vytvoří strom, který reprezentuje vlastní postup vykreslování.



Obr.12.11: Konstrukce BSP stromu

Modifikace pro E^3 je přímočará, neboť se pouze zamění pojem přímka, resp. úsečka, za pojem rovina, resp. plocha.

Podrobněji viz:

- Chen,S., Gordon,D.: Front-to-Back Display of BSP Trees, IEEE Computer Graphics & Algorithms, pp 79–85. September 1991.
 - Hill Jr.,F.S.: Computer Graphics Using OpenGL, pp.707-711, Prentice Hall, 2001

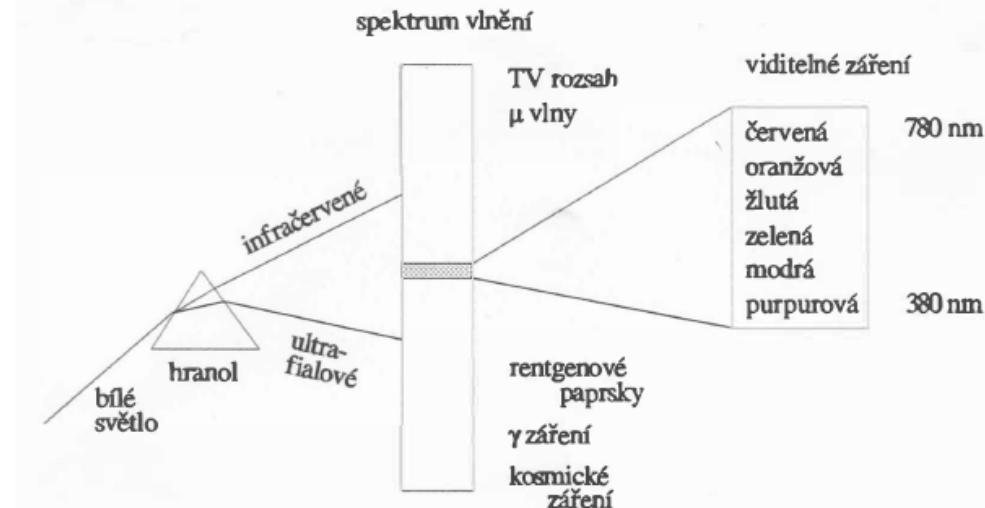
Obecněji:

- How the BSP Tree Works, ([CLICK off-line](#))
http://www.youtube.com/watch?feature=player_detailpage&v=Hebi641Ph7c

V předchozí části jsme se věnovali algoritmům, datovým strukturám, reprezentaci scény atd. Je pochopitelné, že scénu nakonec bude nutné nějakým způsobem zobrazit. Je tedy nutné se zabývat otázkou světelných poměrů ve scéně a způsoby, které nám umožní scénu efektivně zobrazovat.

13. Barvy a barevné systémy

2 V roce 1666 objevil Isaac Newton, že sluneční paprsek procházející skleněným hranolem není na
 3 výstupní straně bílý, ale obsahuje barevné spektrum obsahující barvy od červené až po fialovou. Jsou
 4 to spektrální barvy, které lze nalézt např. v duze po dešti. Proto se také někdy mluví o duhových
 5 barvách. Tento fyzikální fenomén je dán tím, že lom světla na rozhraní dvou opticky jiných materiálů
 6 je závislý na vlnové délce λ .



Obr.13.1: Princip rozkladu bílého světla

11 Vjem barvy závisí na vlnové délce vnímaného světla, resp. jejich kombinací. Lidské oko vnímá vlnovou
 12 délku od cca 380 nm (fialové barvy) do cca 780 nm (červené barvy). Kratší vlnové délky náleží
 13 k ultrafialovému světlu, které také způsobuje zánět spojivek a vzniká např. při sváření elektrickým
 14 obloukem. Na straně delších vlnových délek světlo přechází k infračervenému světlu, které vnímáme
 15 jako teplo.

17 Barva, kterou vnímáme je obecně dána:

- 18 • spektrálním složením světla, kterým je objekt, resp. scéna osvětlena
- 19 • spektrálními vlastnostmi pozorovaného objektu, na jeho optické povaze, např. propustnost,
průhlednost apod. (některé materiály pouze propouštějí, některé pouze odrážejí atd.)
- 21 • optickými vlastnostmi prostředí

22 Je nutné si uvědomit, že pokud povrch odráží pouze modré světlo a povrch osvítíme světlem mající
 23 barvu červenou, bude se nám povrch jevit jako černý, neboť povrch odráží pouze světlo barvy modré.

25 Také je nutné si uvědomit, že vjem pozorovatele má integrační charakter, takže při pozorování shluku
 26 barevných bodů je výsledný vjem vlastně optickou barevnou iluzí, např. barevný bod na obrazovce je
 27 vlastně shlukem barevných zdrojů světla, většinou červené, zelené a modré různé intenzity, které se
 28 pozorovateli jeví jako bod určité barvy.

30 Pokud světlo obsahuje všechny vlnové délky přibližně o stejně intenzitě, světlo se považuje za
 31 achromatické. Intenzitu světla je vhodné uvažovat v normalizovaném tvaru, tj. v intervalu $\langle 0,1 \rangle$.

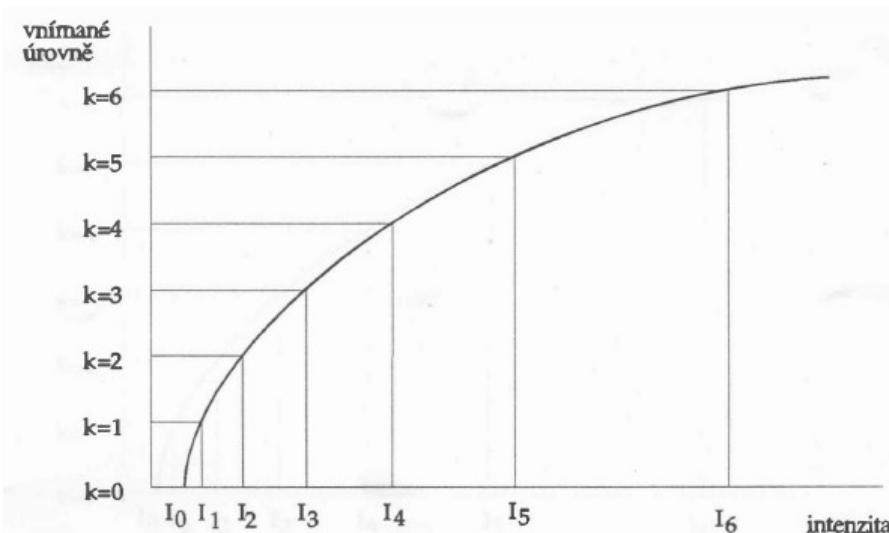
13.1. Achromatické světlo

2 Achromatické světlo se poměrně často vyskytuje v mnoha aplikacích, např. černobílá fotografie,
 3 monochromatický display, rentgenové snímky, tiskárny atd. V případě achromatického světla je jeho
 4 jedinou vlastností (atributem) intenzita, která se označuje jako úroveň šedi. Lidské vnímání je
 5 schopno běžně rozlišit cca 32 úrovní šedi. Na hranicích oblastí s různou úrovní šedi se projevuje tzv.
 6 Machův efekt (Ernst Mach http://en.wikipedia.org/wiki/Ernst_Mach), kdy pozorovatel vidí hranici
 7 oblastí světlejší nebo tmavší, než ve skutečnosti je.



Obr.13.2: Měnící se kontrast při kontaktu oblastí – animovaný GIF (Wikipedia)

8 Je nutné zdůraznit, že vnímaná úroveň šedi na intenzitě světla závisí *nelineárně*. Z Lambert-Beerova
 9 zákona vyplývá, že pokud úrovně šedi mají být vnímány „rovnoměrně“, pak intenzita světla musí být
 10 odstupňována logaritmicky.
 11



Obr.13.3: Závislost vnímané úrovně šedi na intenzitě světla

16 Obecně tedy lze psát:

$$I_1 = rI_0, I_2 = rI_1, \dots, I_k = rI_{k-1} = r^k I_0 \quad (13.1)$$

17 kde: I_0 je nenulová prahová hodnota.

19 S achromatickým světlem je spojen problém, jak vytisknout obraz, který má několik úrovní šedi, ale
 20 výstupní zařízení je černobílé, tj. mající pouze černou a bílou barvu, např. laserová tiskárna. K tomu
 21 slouží metody založené na půltónování. V následujícím textu si vysvětlíme základní principy
 22 jednotlivých metod a jejich vlastnosti.

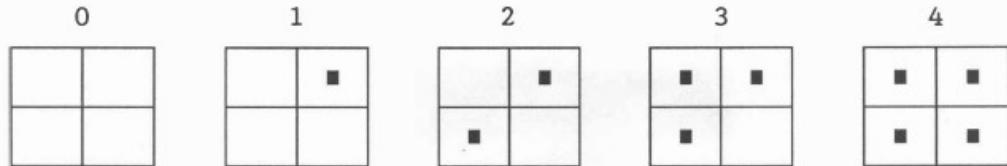
13.2. Půltónování

Techniky půltónování jsou založeny na integračním charakteru obrazového vnímání. Jde vlastně o transformaci obrázků s více úrovněmi šedi na obraz mající pouze dvě úrovně šedi, tj. černou a bílou, přičemž požadujeme, aby výsledný vjem byl co nejblíže vjemu původního obrazu s více úrovněmi šedi. Tyto techniky lze rozdělit v zásadě na dvě hlavní skupiny, a to:

- na metody, které zvětšují rozměr obrazu, např. metoda vzorů
- na metody zachovávající rozměr obrazu, např. rozmývání nebo dithering

13.2.1. Metoda vzorů

Metoda vzorů je založena na přímé nahradě daného pixelu vzorem, tj. skupinou pixelů, který vytváří vjem odpovídající požadované úrovní šedi. Např. pro 5 úrovní šedi lze použít vzory:



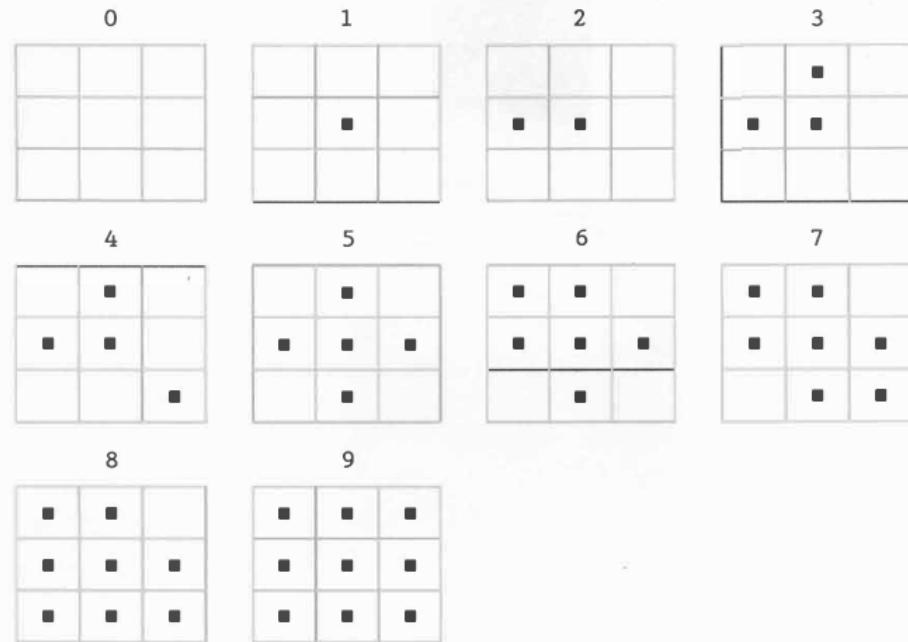
Obr.13.4: Typické vzory 2×2

Vzory typu:



Obr.13.5: Nevhodné typy vzorů

by se neměly používat, neboť lidské vnímání detekuje svislé a vodorovné struktury jako svislé nebo vodorovné úsečky. Pro více úrovní je nutné použít větší vzory, např. pro 10 úrovní je nutné použít vzor 3×3 , a to:



Obr.13.6: Typický vzor 3×3

19

20

21

- 1 Je tedy zřejmé, že technika vzorů zvětšuje rozměr obrazu, tj. faktorem 3 ve výše uvedeném případě.
 2 Navíc záznam vzoru formou obrázku je poměrně nepraktický. Výše uvedený vzor můžeme
 3 reprezentovat pomocí matice a pro uvedený vzor má matice tvar:

$$\begin{matrix} {}^{(3)}\mathbf{T} = \end{matrix} \begin{bmatrix} 7 & 9 & 5 \\ 2 & 1 & 4 \\ 6 & 3 & 8 \end{bmatrix}$$

- 4 přičemž její interpretace je jednoduchá. Pro danou intenzitu se aktivují ty pixely vzoru, jejichž
 5 hodnota na odpovídající pozici v matici vzoru je menší nebo rovna požadované intenzitě. Pro 16
 6 úrovní šedi lze použít vzor 4×4 definovaný maticí:

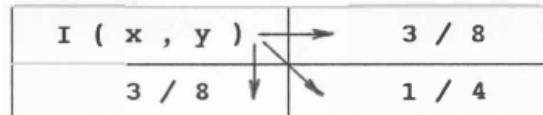
$$\begin{matrix} {}^{(4)}\mathbf{T} = \end{matrix} \begin{bmatrix} 1 & 9 & 3 & 15 \\ 13 & 5 & 14 & 7 \\ 4 & 10 & 2 & 12 \\ 16 & 8 & 11 & 6 \end{bmatrix}$$

- 7 V případě, že vertikální a horizontální vzdálenosti jsou na výstupním médiu rozdílné, lze toto částečně
 8 eliminovat pomocí obdélníkového vzoru.

- 9
 10 V některých případech není akceptovatelné zvětšení rozměru obrazu. V takových případech je nutné
 11 použít jiných technik, které jsou uvedeny níže.
 12

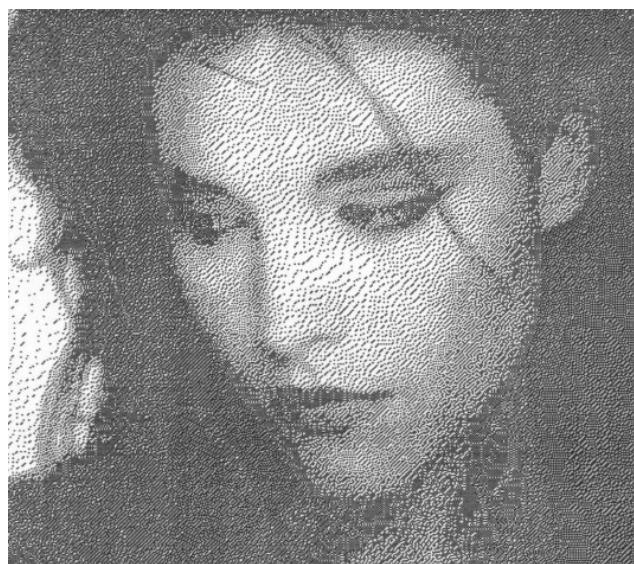
13.2.2. Metoda rozptylu chyb (Floyd-Steinberg)

- 14 Základní metodu rozptylu chyb navrhli Floyd a Steinberg v roce 1975. Metoda je založena vlastně na
 15 prahování, kdy daný pixel aktivujeme, pokud hodnota intenzity je větší než zvolený práh T . Nicméně
 16 při tomto postupu vzniká chyba, která je „distribuována“ do sousedních pixelů, např. podle pravidla:



Obr.13.7: Princip „rozmývání“ - distribuuje chyby

- 19 Výsledný obrázek je charakteristický tím, že se vytváří „duchy“ okolo oblastí s velkým rozdílem
 20 intenzit.



Obr.13.8: Výstup z Floyd-Steinbergova algoritmu

- 23 Je vhodné poznamenat, že existuje více schémat, jak vlastní chybu distribuovat.

1 Vlastní algoritmus pak lze popsát takto:

```

const xmin = 0; xmax = 1023; { rozsah rastru pro směr x }
    ymin = 0; ymax = 1023; { rozsah rastru pro směr y }
    Black_gray = 0.0; White_gray = 1.0;
var T, Error: real;
    x, y: integer;
begin T := (Imax + Imin) / 2;
    for y := ymax downto ymin do
        for x := xmin to xmax do
            begin { určení hodnoty pixelu pro práh T }
                { a chybou Error }
                if I(x,y) < T then
                    begin PutPixel(x,y,Black);
                        Error := I(x,y) - Black_gray
                    end
                else
                    begin PutPixel(x,y,White);
                        Error := I(x,y) - White_gray
                    end;
                if x < xmax then I(x+1,y) := I(x+1,y) + 3*Error/8;
                if y > ymin then I(x,y-1) := I(x,y-1) + 3*Error/8;
                if (x < xmax) and (y > ymin)
                    then I(x+1,y-1) := I(x+1,y-1) + Error/4
            end
    end

```

2

3 Algoritmus Floyd-Stenbergův

4

5 Vytváření duchů je poměrně nepřijemné a v některých aplikacích nepřijatelné. Problém duchů
6 částečně odstraňuje dithering.

7

8 13.2.3. Dithering

9 Dithering je metoda kombinující vlastně metodu vzorů a metodu rozptylu chyb. Metoda je založena
10 na vložení „náhodných“ chyb do obrazu, přičemž „náhodnost“ je řízena vzorem, který je dán maticí.
11 Je-li (x, y) pozice pixelu o intenzitě $I(x, y)$, pak odpovídající prvek ve vzoru na pozici (i, j) je určen
12 takto:

$$j = x \bmod n + 1$$

$$i = y \bmod n + 1$$

13

14 Daný pixel je pak aktivován jen tehdy, pokud:

$${}^{(n)}\mathbf{D}[i, j] < I(x, y)$$

15 kde: n je rozměr matice D .

16 Matice ${}^{(2)}\mathbf{D}$ pro vzor 2×2 je dána takto:

$${}^{(2)}\mathbf{D} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

1 Větší vzory pak mohou být získány pomocí rekurentního vztahu, a to:

$$\begin{pmatrix} (2n) \\ \end{pmatrix} \mathbf{D} = \begin{bmatrix} 4^{(n)} \mathbf{D} & 4^{(n)} \mathbf{D} + 2^{(n)} \mathbf{U} \\ 4^{(n)} \mathbf{D} + 3^{(n)} \mathbf{U} & 4^{(n)} \mathbf{D} + 4^{(n)} \mathbf{U} \end{bmatrix}$$

2 přičemž n je rozměr matice a matice $\begin{pmatrix} (n) \\ \end{pmatrix} \mathbf{U}$ rozměru $n \times n$ je definována:

$$\begin{pmatrix} (n) \\ \end{pmatrix} \mathbf{U} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

3 Pro $n = 4$ dostáváme:

$$\begin{pmatrix} (4) \\ \end{pmatrix} \mathbf{D} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

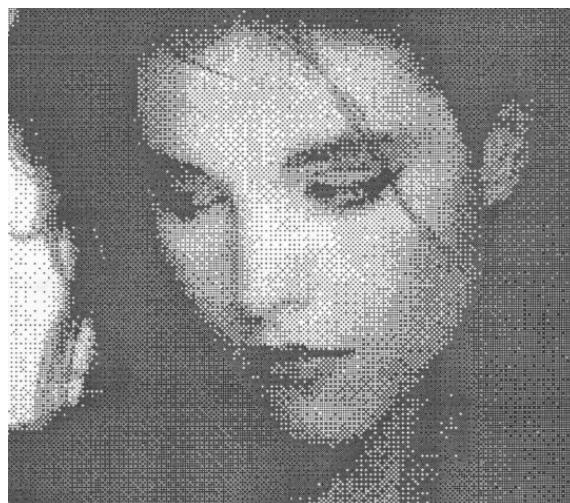
4 Níže uvedená sekvence pak realizuje daný postup, přičemž pro zjednodušení indexy matice \mathbf{D} začínají hodnotou **NULA**.

```

begin
    for y := ymax downto ymin do
        begin
            i := y mod n;
            for x := xmin to xmax do
                begin { určení polohy (i,j) v matici D }
                    j := x mod n;
                    if D[i,j] < I(x,y) then PutPixel(x,y,White)
                        else PutPixel(x,y,Black);
                end
        end
    end

```

Obr.13.9: Algoritmus Dithering



Obr.13.10: Výstup metody Dithering.

9
10 Až dosud jsme se zabývali světlem achromatickým. Principy výše uvedených technik je možné
11 modifikovat i pro chromatické světlo, tj. pro použití barev.
12
13
14

13.3. Chromatické světlo - barvy

Vjem světla určité vlnové délky nebo jejich kombinací vnímá člověk jako určitou barvu. Přirozené světlo je vždy složeno ze světel o různých vlnových délkách. Pro reprezentaci barev potřebujeme nějaký matematický model, který do určité míry bude dobře reprezentovat barevnost světla a potřebné operace, např. míchání barev atd. Je tedy zřejmé, že v modelu nelze postihnout všechny vlastnosti barevného vjemu a jakýkoliv matematický model bude jen z části dobře reprezentovat fyzikální realitu.

V praxi jsou používány dva základní barevné modely, a to:

- aditivní model – zařízení produkující světlo, např. obrazovka apod.; barvy se tvoří sčítáním barev.
- subtraktivní model – světlo se odráží, např. barevná tiskárna apod.; barvy se tvoří přetiskem.

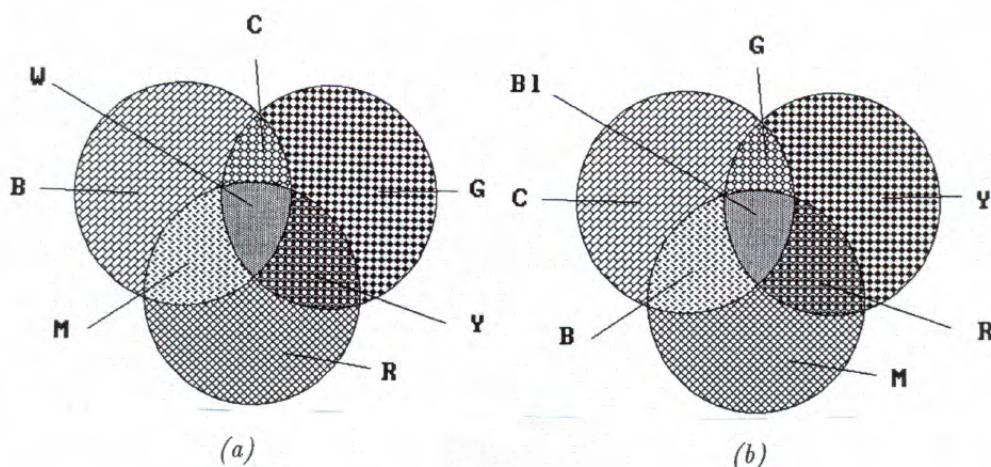
V případě aditivního modelu se používají základní barvy RGB. Mezinárodní komise CIE (Commission de l'Eclairage) stanovila referenční vlnové délky takto:

červená (R - Red)	zelená (G - Green)	modrá (B - Blue)
780.0 [nm]	546.1 [nm]	435.8 [nm]

Převážná část barev pak může být tvořena pomocí tří základních barev, a to:

- červenou (R - Red), zelenou (G - Green), modrou (B - Blue), viz Obr.13.11.a, v aditivním systému. Barva bílá (W - White) vzniká součtem barev R, G, B
- modrozelenou (C - Cyan), purpurovou (M - Magenta), žlutou (Y - Yellow), viz Obr.13.11.b, v subtraktivním systému. Barva černá (Bl - Black) vzniká přetiskem barev C, M, Y

Je nutné také zdůraznit, že u barev musíme rozlišovat intenzitu světla a jeho barevnost.



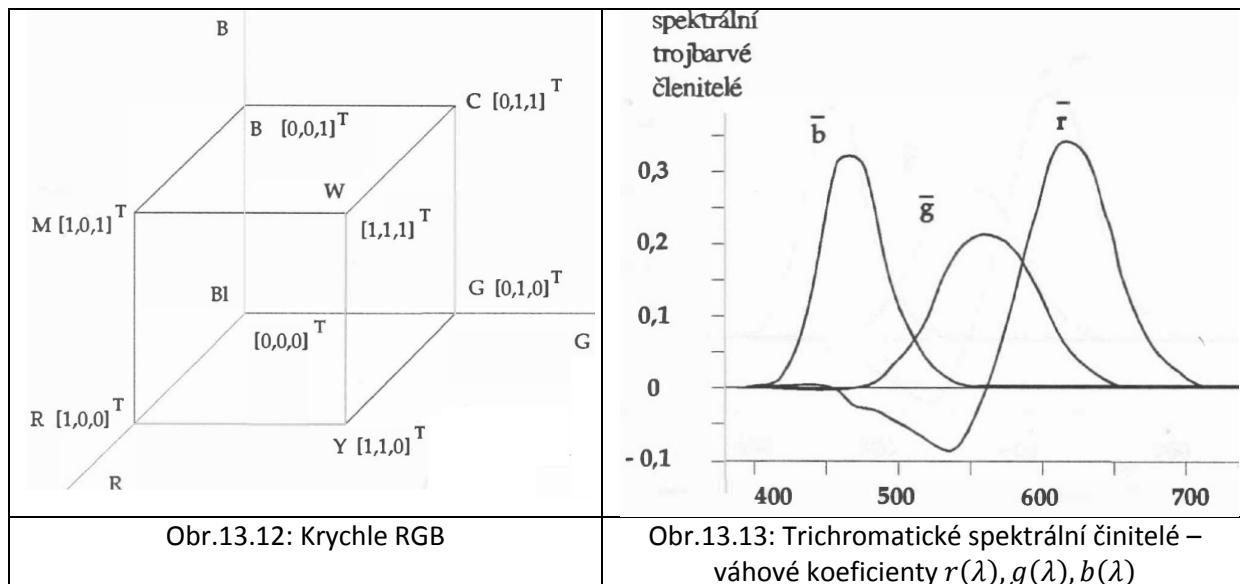
Obr.13.11: Míchání barev v systémech RGB a CMY

Poznámka

Část barev v subtraktivním systému není obsažena ve spektru bílého světla a nemá specifickou vlnovou délku. Proto se u barev CMY vlnové délky neuvádějí a některé barvy jsou charakterizovány tzv. doplňkovou vlnovou délkou, viz systém CIE-xy.

13.3.1. Systém RGB

2 Systém RGB umožňuje jednoduchou reprezentaci většiny barev ve spektru. Vezmeme-li viditelné
 3 spektrum, pak každé vlnové délce odpovídá určitá barva. Systém RGB si lze formálně představit jako
 4 krychli RGB, viz Obr.13.12, a je založen na představě, že vhodným mícháním základních barev RGB
 5 v poměru daném váhovými koeficienty r, g, b získáme požadovanou barvu.
 6



7 Hodnoty r, g, b byly získány pro jednotlivé vlnové délky kolorimetrickým porovnáním. Váhové
 9 koeficienty r, g, b se nazývají trichromatickými spektrálními členiteli a jejich hodnoty v závislosti na
 10 vlnové délce jsou znázorněny na Obr.13.13.

11 Z grafu na Obr.13.13 vyplývá, že pro některé vlnové délky jsou koeficienty r, g, b záporné, a tedy
 13 některé barvy v systému RGB nelze zobrazovat. Protože barva c je určena:

$$c = rR + gG + bB \quad (13.2)$$

14 pak pro oblasti se záporným trichromatickým činitelem bylo kolorimetrické porovnávání realizováno
 15 pomocí „přisvětlení“ zdrojem světla s referenční vlnovou délkou, např.:

$$c + rR = gG + bB \quad (13.3)$$

16 Je nutné zdůraznit, že citlivost lidského vnímání je odlišná pro různé vlnové délky a pro:

$$r: g: b = 1: 1: 1 \quad (13.4)$$

18 dostáváme poměr vnímaného jasu:

$$1: 4.6: 0.06 \quad (13.5)$$

19 Vidíme tedy, že citlivost oka na barvu modrou je velmi malá.

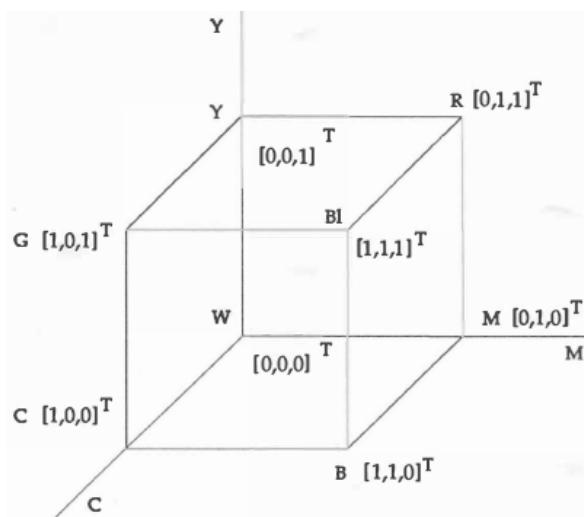
20

13.3.2. Systém CMY a CMYK

Subtraktivní barevný systém CMY se používá zejména pro zařízení s barevným přetiskem. Vlastní barevný prostor si lze opět představit jako CMY krychli, viz Obr.13.14. Vzájemné převody mezi systémem RGB a CMY jsou určeny vztahy:

$$\begin{bmatrix} c \\ m \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} c \\ m \\ y \end{bmatrix} \quad (13.6)$$

Poznamenejme, že vektor $[1,1,1]^T$ reprezentuje v systému RGB barvu bílou, v systému CMY pak barvu černou.



Obr.13.14: Krychle CMY

Se systémem CMY v praxi jsou trochu problémy, neboť zejména při přetisku barev, např. u inkoustových tiskáren, dochází k nepřesnému „usazení“ barevného pigmentu.

Teoreticky je možné barvu černou vytisknou přetiskem barev CMY, avšak výsledná barva by neměla požadovanou „tmavost“, resp. sytost a také cena za černobílý tisk takto realizovaný by byla velká. Proto se ještě přidává barva černá jako speciální složka a takový systém se pak označuje CMYK.

Až dosud jsme se zabývali reprezentací barev. Podíváme-li se na problematiku z jiné strany, pak barvu je možné také popsat pomocí intenzity a dvou dalších parametrů, které vyjadřují barevnost. Na této reprezentaci je založen systém CIE-xy a CIE-uv.

V současné době se u barevných tiskáren nejen používá více barev k přetisku, neboť se tím získá větší barevný gamut, ale používají se také barvy „kovové“, např. zlatavé, stříbrné atd., nebo barvy luminiscenční apod.

13.3.3. Systém XYZ a CIE-xy

2 Představme si opět jednotkovou krychli RGB a jednotkovou rovinu $r + g + b = 1$. Je zřejmé, že
 3 všechny barvy mající stejnou vlnovou délku, ale různou intenzitu se promítnou do stejného bodu na
 4 jednotkové rovině. Barva c je nyní reprezentována jasem a hodnotami \bar{r} , \bar{g} , \bar{b} , které jsou určeny
 5 vztahy:

$$\bar{r} = \frac{r}{r + g + b} \quad \bar{g} = \frac{g}{r + g + b} \quad \bar{b} = \frac{b}{r + g + b} \quad (13.7)$$

6 a tedy:

$$\bar{r} + \bar{g} + \bar{b} = 1 \quad (13.8)$$

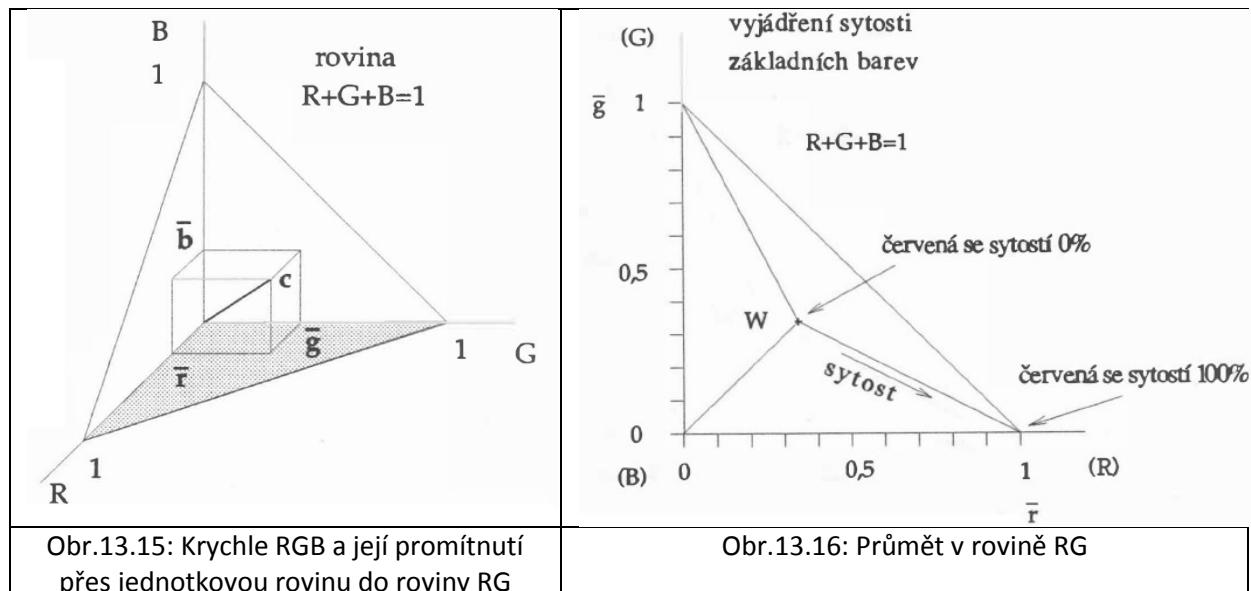
7 Protože hodnotu \bar{b} je možné tedy určit ze vztahu:

$$\bar{b} = 1 - \bar{r} - \bar{g} \quad (13.9)$$

8 je barevnost vlastně určena koeficienty \bar{r} , \bar{g} .

9
 10 Promítneme-li nyní jednotlivé barvy v krychli RGB z jednotkové roviny do roviny RG, viz Obr.13.15,
 11 pak dostáváme diagram na Obr.13.16, který obsahuje všechny barvy reprezentovatelné v systému
 12 RGB, tj. $r, g, b \in \langle 0,1 \rangle$, nehledě na intenzitu. Symbol W označuje pozici barvy bílé. Sytost barvy je
 13 dána poměrem dané barvy a barvy bílé.

14

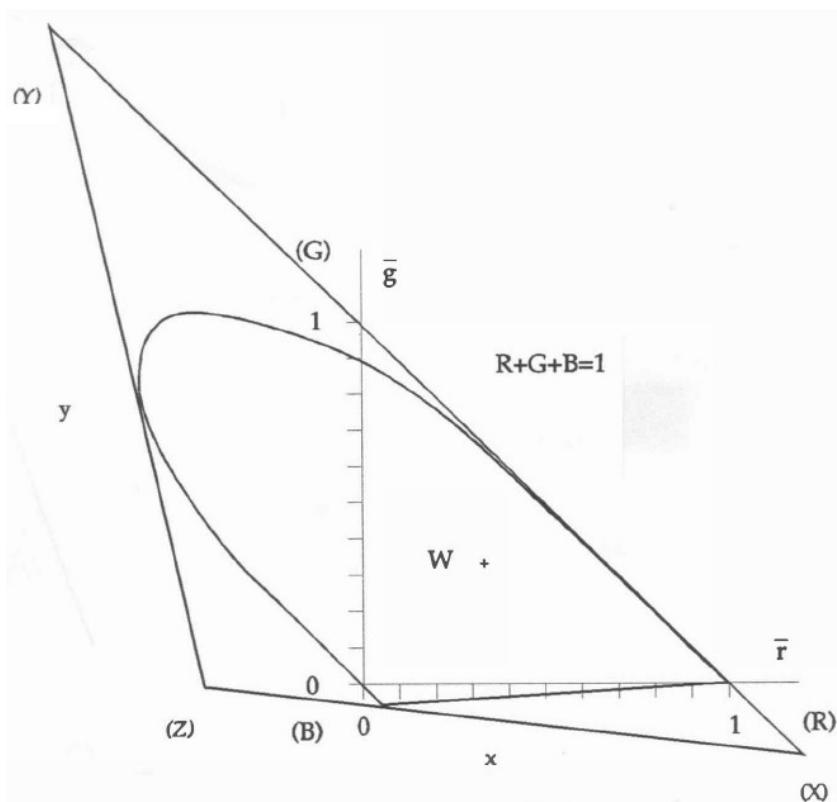


15
 16 U systému RGB jsme ukázali, jak závisí koeficient r, g, b na vlnové délce λ , přičemž pro některé
 17 vlnové délky koeficienty nabývaly záporných hodnot. Takže vzniká otázka, jak se hodnoty koeficientů
 18 r, g, b pro jednotlivé vlnové délky λ promítnou do diagramu $\bar{r}\bar{g}$, viz Obr.13.16. Výše uvedený postup
 19 výpočtu koeficientů \bar{r}, \bar{g} z hodnot r, g, b je nutné aplikovat pro jednotlivé vlnové délky λ . Výsledek je
 20 zobrazen na Obr.13.17.

21
 22 Je tedy zřejmé, že systémy založené na reprezentaci RGB neumožňují dobře reprezentovat barvu ve
 23 srovnání s fyzikální realitou.

24
 25 **Upozornění**
 26 Barvy, pro které je $\bar{r} < 0$, resp. $\bar{g} < 0$, jsou barvy v systému RGB nerealizovatelné.

27

1
23
4
5
6Obr.13.17: Průběh hodnot \bar{r} , \bar{g} pro vlnové délky viditelného spektra

1 Systém XYZ

2 Systém XYZ je systémem virtuálních souřadnic, které slouží k tomu, abychom se vyhnuli záporným
3 koeficientům. Systém virtuálních souřadnic XYZ byl definován takto:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 2.7689 & 1.7518 & 1.1302 \\ 1.0000 & 4.5907 & 0.0601 \\ 0.0000 & 0.0565 & 5.5943 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (13.10)$$

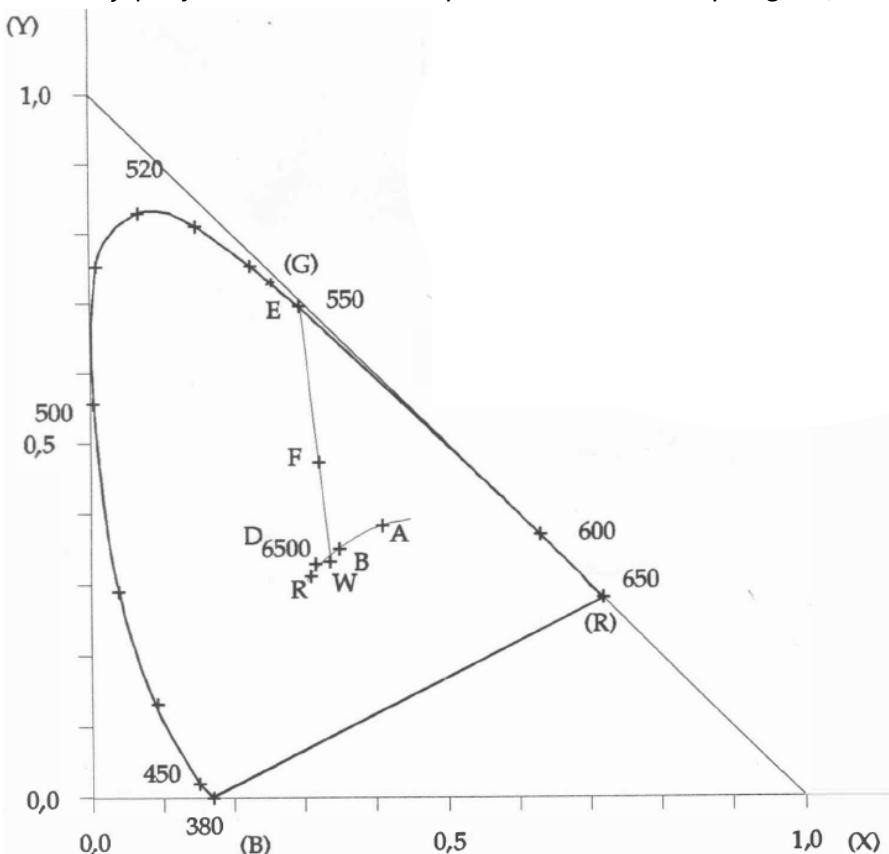
4 příčemž výsledný jas je dán hodnotou Y . Promítneme-li opět souřadnice XYZ na jednotkovou rovinu,
 5 dostáváme:

$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad z = \frac{Z}{X+Y+Z} \quad (13.11)$$

6 přičemž $x + y + z = 1$

8 CIE-xy diagram

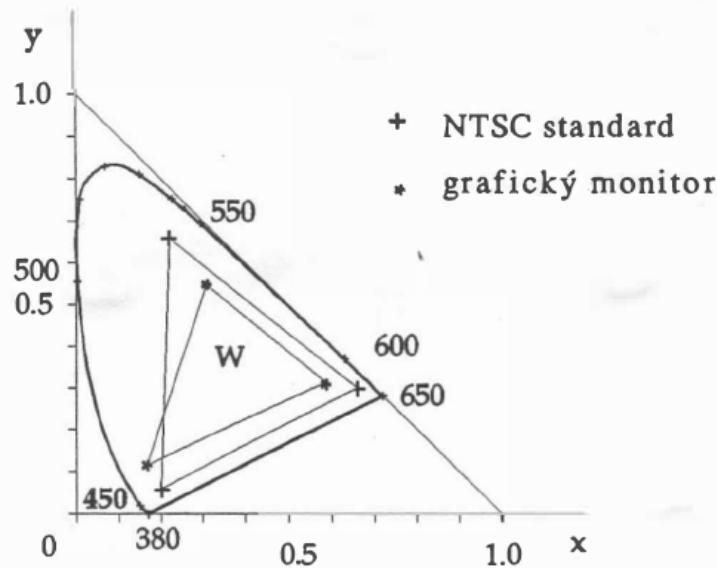
9 Zobrazením hodnot x, y pro jednotlivé vlnové délky dostáváme tzv. CIE-xy diagram, viz Obr.13.18.



Obr.13.18: Schematický CIE-xy diagram

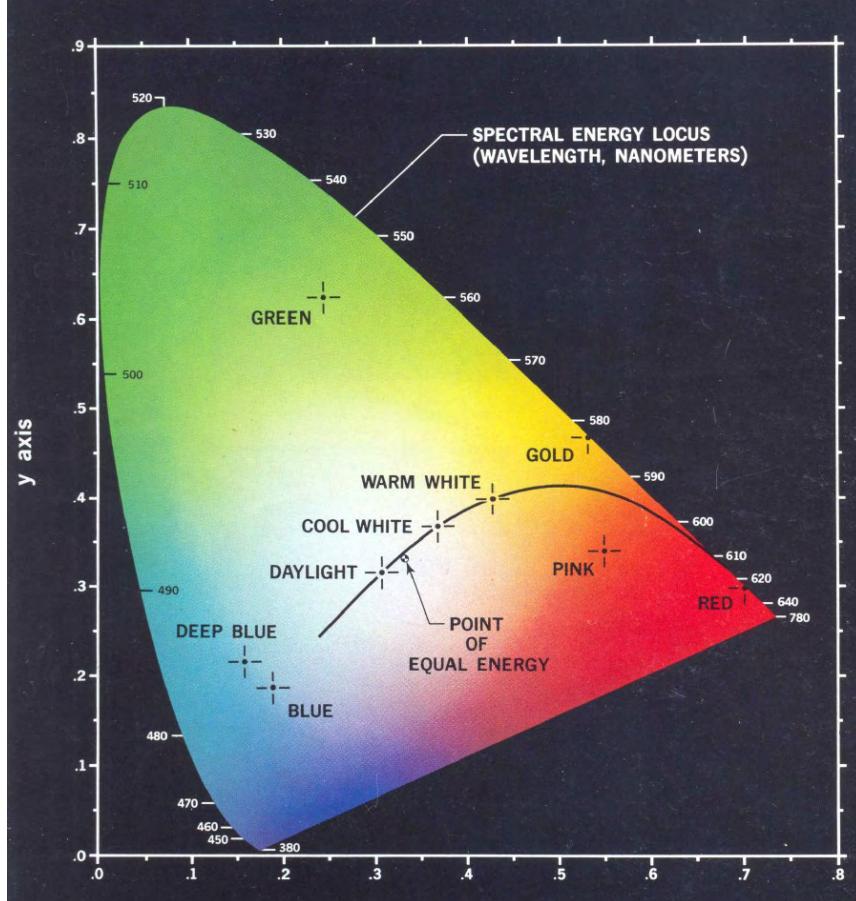
12 Na Obr.13.18 vidíme křivku reprezentující spektrální barvy se sytostí 100% pro jednotlivé vlnové
13 délky viditelného spektra, uprostřed oblast bílých barev (pozice barvy bílé závisí na její „teplotě“).
14 Bod dané barvy bílé (W), modré (380) a červené (650) tvoří trojúhelníkovou oblast barev
15 nespektrálních, tj. barev, které nejsou ve viditelném spektru a které jsou např. barvami tisknutými.
16 Spojnice WE reprezentuje stejnou barvu, avšak s různou sytostí. Sytost barvy na pozici F je dána
17 poměrem WF/WE , přičemž W je pozice aktuální bílé barvy. Je tedy zřejmé, že pokud F barva
18 s danou sytostí, pak při změně pozice barvy bílé, tj. např. při změně „teplé bílé“ na „studenou bílou“,
19 bude reprezentativní vlnová délka barvy se sytostí 100% jiná, a tedy i vjem barvy bude jiný.
20 Přímý důsledek: nutnost kalibrace barvy bílé na monitorech pro DTP.

- 1 Technologická omezení navíc neumožňují konstrukci zařízení umožňující sytost 100%. Každé zařízení
 2 má vlastní pozice pro reprezentaci barev r, g, b , které definují barevný gamut daného zařízení, který
 3 je na Obr.13.19 znázorněn trojúhelníkem. Různá zařízení mají různý gamut a oblast barev
 4 reproducovatelných na obou zařízeních je dána vlastně společnou částí obou gamutů, tj. průnikem
 5 daných trojúhelníků.



Obr.13.19: CIE-xy a barevný gamut

(C. I. E. CHROMATICITY DIAGRAM)



Obr.13.20: Barevný CIE-xy diagram

- 10 Hodnoty rgb a xyz pro jednotlivé vlnové délky jsou uvedeny v následující tabulce

1 Tabulka *rgb a xyz koeficientů*

λ [nm]	r	g	b	x	y	z
380	0,00003	-0,00001	0,00117	0,1741	0,0050	0,8209
385	0,00005	-0,00002	0,00189	0,1740	0,0050	0,8210
390	0,00010	-0,00004	0,00359	0,1738	0,0049	0,8213
395	0,00017	-0,00007	0,00647	0,1736	0,0049	0,8215
400	0,00030	-0,00014	0,01214	0,1733	0,0048	0,8219
405	0,00047	-0,00022	0,01969	0,1730	0,0048	0,8222
410	0,00084	-0,00014	0,03707	0,1726	0,0048	0,8226
415	0,00139	-0,00070	0,06637	0,1721	0,0048	0,8231
420	0,00211	-0,00110	0,11541	0,1714	0,0051	0,8235
425	0,00266	-0,00143	0,18575	0,1703	0,0058	0,8239
430	0,00218	-0,00119	0,24769	0,1689	0,0069	0,8245
435	0,00036	-0,00021	0,29012	0,1669	0,0086	0,8246
440	-0,00261	0,00149	0,31228	0,1644	0,0109	0,8247
445	-0,00673	0,00379	0,31860	0,1611	0,0138	0,8251
450	-0,01213	0,00678	0,31670	0,1566	0,0177	0,8257
455	-0,01874	0,01046	0,31166	0,1510	0,0227	0,8263
460	-0,02608	0,01485	0,29821	0,1440	0,0297	0,8263
465	-0,03324	0,01977	0,27295	0,1355	0,0399	0,8246
470	-0,03933	0,02538	0,22991	0,1241	0,0578	0,8181
475	-0,04471	0,03183	0,18592	0,1096	0,0868	0,8038
480	-0,04939	0,03914	0,14494	0,0913	0,1327	0,7760
485	-0,05364	0,04713	0,10968	0,0687	0,2007	0,7306
490	-0,05814	0,05689	0,08257	0,0454	0,2950	0,6595
495	-0,06414	0,06948	0,06246	0,0235	0,4127	0,5638
500	-0,07173	0,08536	0,04776	0,0082	0,5384	0,4534
505	-0,08120	0,10593	0,03688	0,0039	0,6548	0,3413
510	-0,08901	0,12860	0,02698	0,0139	0,7502	0,2359
515	-0,09356	0,15262	0,01842	0,0389	0,8120	0,1491
520	-0,09264	0,17468	0,01221	0,0743	0,8338	0,0919
525	-0,08473	0,19113	0,00830	0,1142	0,8262	0,0596
530	-0,07101	0,20317	0,00549	0,1547	0,8059	0,0394
535	-0,05316	0,21083	0,00320	0,1929	0,7816	0,0255
540	-0,03152	0,21466	0,00146	0,2296	0,7543	0,0161
545	-0,00613	0,21487	0,00023	0,2658	0,7243	0,0099
550	0,02279	0,21178	-0,00058	0,3016	0,6923	0,0061
555	0,05514	0,20588	-0,00105	0,3373	0,6589	0,0038
560	0,09060	0,19702	-0,00130	0,3731	0,6245	0,0024
565	0,12840	0,18522	-0,00138	0,4087	0,5896	0,0017
570	0,16768	0,17087	-0,00135	0,4441	0,5547	0,0012
575	0,20715	0,15429	-0,00123	0,4788	0,5202	0,0010
580	0,24526	0,13610	-0,00108	0,5125	0,4866	0,0009
585	0,27989	0,11686	-0,00093	0,5448	0,4544	0,0008
590	0,30928	0,09754	-0,00079	0,5752	0,4242	0,0006
595	0,33184	0,07909	-0,00063	0,6029	0,3965	0,0006
600	0,34429	0,06246	-0,00049	0,6270	0,3725	0,0005
605	0,34756	0,04776	-0,00038	0,6482	0,3514	0,0004
610	0,33971	0,03557	-0,00030	0,6658	0,3340	0,0002
615	0,32265	0,02583	-0,00022	0,6801	0,3198	0,0002
620	0,29708	0,01828	-0,00015	0,6915	0,3083	0,0002
625	0,26348	0,01253	-0,00011	0,7006	0,2993	0,0001
630	0,22677	0,00833	-0,00008	0,7079	0,2920	0,0001
635	0,19233	0,00537	-0,00005	0,7140	0,2859	0,0000
640	0,15968	0,00334	-0,00003	0,7190	0,2809	0,0000

λ [nm]	r	g	b	x	y	z
645	0,12905	0,00199	-0,00002	0,7230	0,2770	0,0000
650	0,10167	0,00116	-0,00001	0,7260	0,2740	0,0000
655	0,07857	0,00066	-0,00001	0,7283	0,2717	0,0000
660	0,05932	0,00037	0,00000	0,7300	0,2700	0,0000
665	0,04366	0,00021	0,00000	0,7311	0,2689	0,0000
670	0,03149	0,00011	0,00000	0,7320	0,2680	0,0000
675	0,02294	0,00006	0,00000	0,7327	0,2673	0,0000
680	0,01687	0,00003	0,00000	0,7334	0,2666	0,0000
685	0,01187	0,00001	0,00000	0,7340	0,2660	0,0000
690	0,00819	0,00000	0,00000	0,7344	0,2656	0,0000
695	0,00572	0,00000	0,00000	0,7346	0,2654	0,0000
700	0,00410	0,00000	0,00000	0,7347	0,2653	0,0000
705	0,00291	0,00000	0,00000	0,7347	0,2653	0,0000
710	0,00210	0,00000	0,00000	0,7347	0,2653	0,0000
715	0,00148	0,00000	0,00000	0,7347	0,2653	0,0000
720	0,00105	0,00000	0,00000	0,7347	0,2653	0,0000
725	0,00074	0,00000	0,00000	0,7347	0,2653	0,0000
730	0,00052	0,00000	0,00000	0,7347	0,2653	0,0000
735	0,00036	0,00000	0,00000	0,7347	0,2653	0,0000
740	0,00025	0,00000	0,00000	0,7347	0,2653	0,0000
745	0,00017	0,00000	0,00000	0,7347	0,2653	0,0000
750	0,00012	0,00000	0,00000	0,7347	0,2653	0,0000
755	0,00008	0,00000	0,00000	0,7347	0,2653	0,0000
760	0,00006	0,00000	0,00000	0,7347	0,2653	0,0000
765	0,00004	0,00000	0,00000	0,7347	0,2653	0,0000
770	0,00003	0,00000	0,00000	0,7347	0,2653	0,0000
775	0,00001	0,00000	0,00000	0,7347	0,2653	0,0000
780	0,00000	0,00000	0,00000	0,7347	0,2653	0,0000

Tab.XX: Hodnoty r, g, b a x, y, z spektrálních barev pro jednotlivé vlnové délky λ

13.3.4. Systém YIQ

Systém YIQ byl zaveden pro televizní vysílání v r.1953, a to z důvodů kompatibility televizního příjmu pro černobílé a barevné vysílání. Vzájemné převody mezi RGB a YIQ jsou dány vztahy:

$$\begin{bmatrix} y \\ i \\ q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1.000 & 0.959 & 0.623 \\ 1.000 & -0.272 & -0.648 \\ 1.000 & -1.105 & 1.705 \end{bmatrix} \begin{bmatrix} y \\ i \\ q \end{bmatrix} \quad (13.12)$$

Složka Y s frekvenční šírkou 4 MHz dává informaci o jasu, složka I s frekvenční šírkou 1.5 MHz obsahuje informaci o barvě mezi oranžovou a modrozelenou, složka Q s frekvenční šírkou 0.6 MHz obsahuje informaci o barvě mezi zelenou a purpurovou.

Upozornění

Uvedený vztah definuje převod pro monochromatické zobrazení z barevné předlohy. Jas y je dán výrazem:

$$y = [0.299 \quad 0.587 \quad 0.114][r, g, b]^T$$

13.3.5. Systém Lab – doplnit**13.3.6. Systém AC₁C₂ – doplnit****13.3.7. Systém Opponent – doplnit****13.3.8. Profily ICC – doplnit**

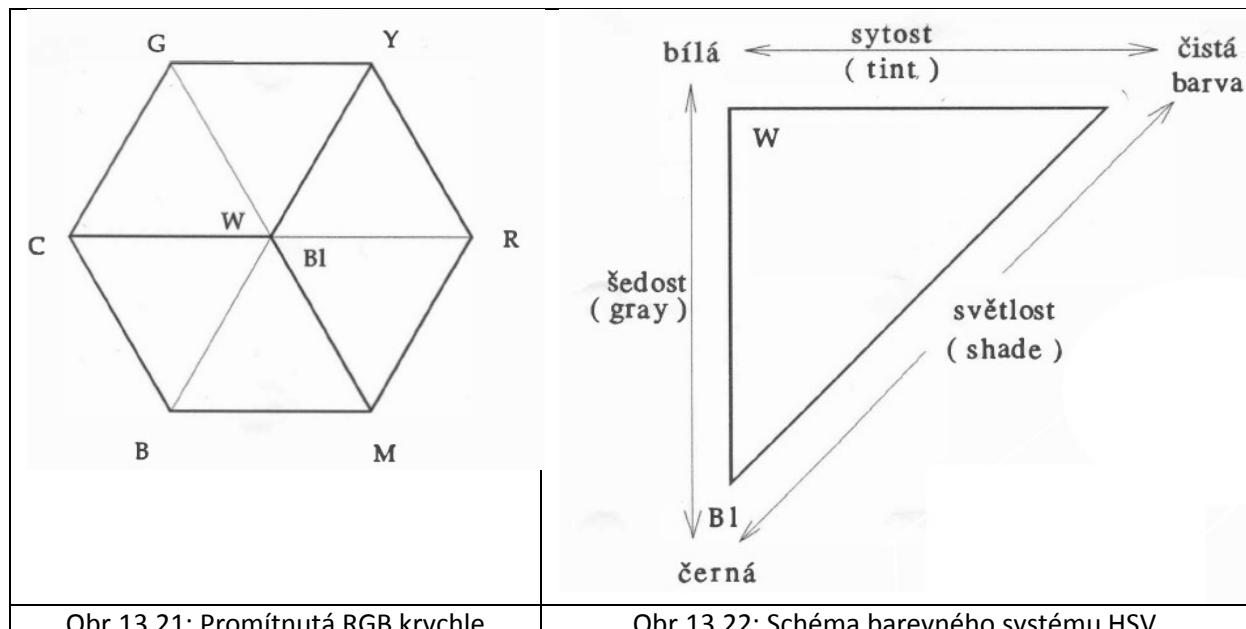
Až dosud jsme se zabývali barevnými systémy orientovanými na technologická zařízení, která nejsou vhodná z hlediska uživatelského. V následujícím textu ukážeme základní uživatelsky orientované systémy HLS, HSV a HSI.

13.3.9. Systém HSV, HLS a HSI

2 Uživatelsky orientované barevné systémy vychází z zásadě z představy práce malíře, který míchá
 3 barvu tak, aby měla požadovaný odstín a světlost, resp. tmavost. Podíváme-li se na RGB krychli po
 4 tělesové úhlopříčce ve směru od barvy bíle k barvě černé, uvidíme po projekci vlastně šestiúhelník,
 5 viz Obr.13.21. Z této představy vychází pak systémy HLS a HSV. Tyto systémy jsou založeny na:

- 6 • tónu barvy – H (Hue)
- 7 • sytosti barvy – S (Saturation)
- 8 • jasu, resp. světlosti/tmavosti – V (Value), resp. L (Lightness)

9 Odtud jsou odvozeny názvy HSV a HLS těchto systémů:



12 Obr.13.22 znázorňuje vztahy na řezu pro konstantní tón barvy, tj. $H = \text{konst.}$, pro systém HSV.

15 Oba systémy, tj. HSV a HLS, se liší v zásadě jen velmi málo, neboť HSL respektuje nejen menší
 16 rozpoznatelnost tmavých barev, ale též i barev hodně světlých.

18 Upozornění

- 19 • převod barev z RGB do HLS a HSV však už není lineární a je nutné použít programovou
 20 sekvenci, viz níže. Analogické sekvence jsou i pro transformaci HLS a transformace zpětné.
- 21 • lineární interpolace barev použitelná v RGB systému *není použitelná* v těchto systémech.

```

begin
    max := MAXIMUM ( r , g , b );
    min := MINIMUM ( r , g , b );
    v := max; { hodnota v }
    if max <> 0 then s := ( max - min ) / max
        else s := 0;
    if s = 0 then h := nedefinováno
    else
        begin q := 1.0 / ( max - min );
            rc := ( max - r ) * q; { vzdálenost od červené }
            gc := ( max - g ) * q; { vzdálenost od zelené }
            bc := ( max - b ) * q; { vzdálenost od modré }
            if r = max then h := bc - gc
                { barva mezi žlutou a purpurovou }
            else if g = max then h := 2 + rc - bc
                { barva mezi modrozelenou a žlutou }
            else if b = max then h := 4 + gc - rc;
                { barva mezi modrozelenou a purpurovou }
            h := h * 60; { konverze do stupňů }
            if h < 0.0 then h := h + 360 { konverze do stupňů }
        end { chromatický případ }
1   end { zRGBdoHSV };
2
3 Zajímavým systémem je systém HSI, kde vzájemné převody RGB $\leftrightarrow$ HSI jsou formulovány jiným
4 způsobem.
5
6 Systém HSI
7 Systém HSI je kompromisem mezi uživatelským přístupem reprezentovaným systémy HSV a HSL
8 a akceptovatelné výpočetní složitosti převodu RGB $\leftrightarrow$ HSI.
9
10 Převody mezi RGB a HSI jsou definovány takto:

```

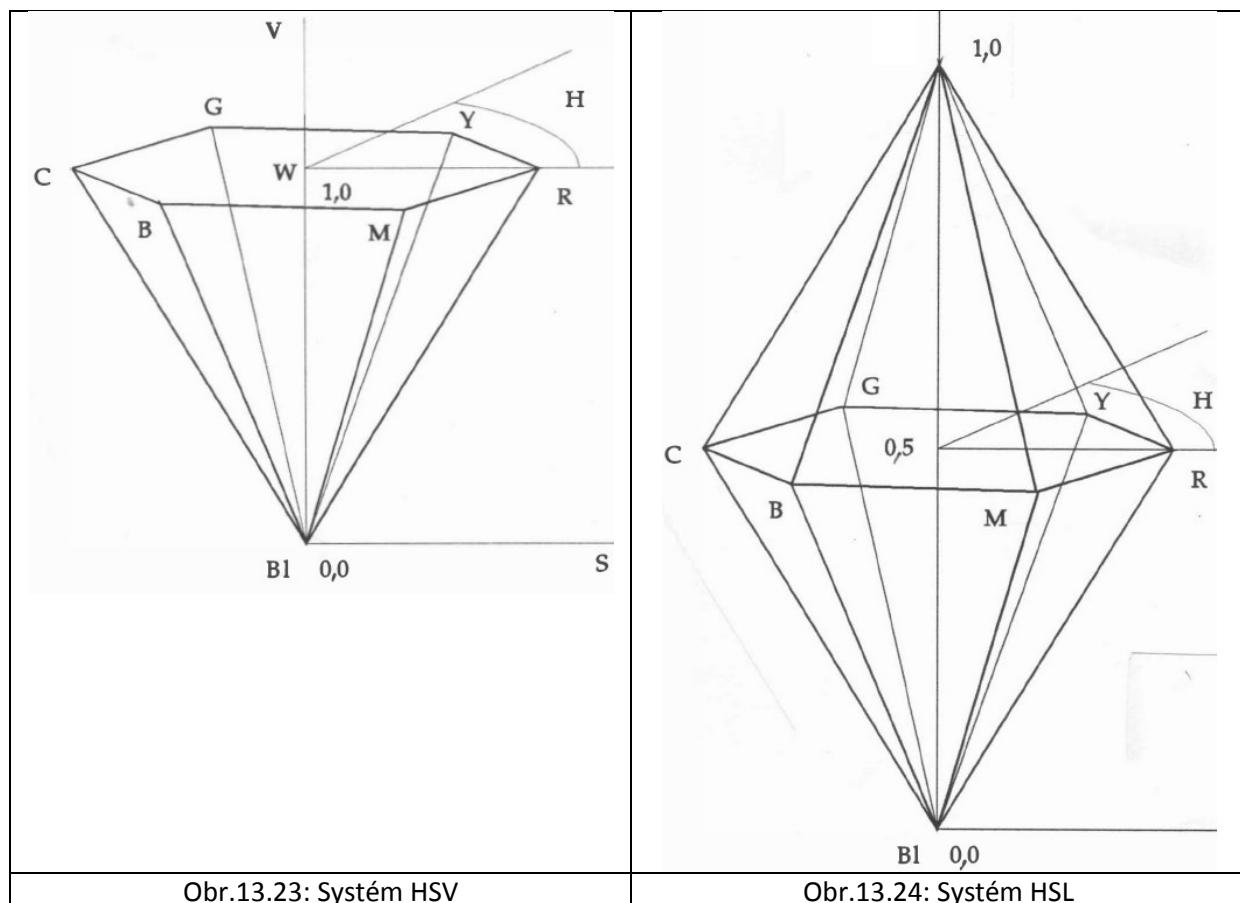
$$\mathbf{Q} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \\ \sqrt{2} & \sqrt{2} & \sqrt{2} \end{bmatrix} \quad (13.13)$$

11

$\begin{bmatrix} M_1 \\ M_2 \\ I_1 \end{bmatrix} = \mathbf{Q} \begin{bmatrix} r \\ g \\ b \end{bmatrix}$ $H = \arctg \left(\frac{M_1}{M_2} \right)$ $S = \frac{\sqrt{3}}{\sqrt{2}} \sqrt{M_1^2 + M_2^2}$ $I = \frac{\sqrt{3}}{3} I_1$	$M_1 = \frac{\sqrt{2}}{\sqrt{3}} S \sin H$ $M_2 = \frac{\sqrt{2}}{3} S \cos H$ $I_1 = \sqrt{3} I$ $\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \mathbf{Q}^{-1} \begin{bmatrix} M_1 \\ M_2 \\ I_1 \end{bmatrix}$
--	---

(13.14)

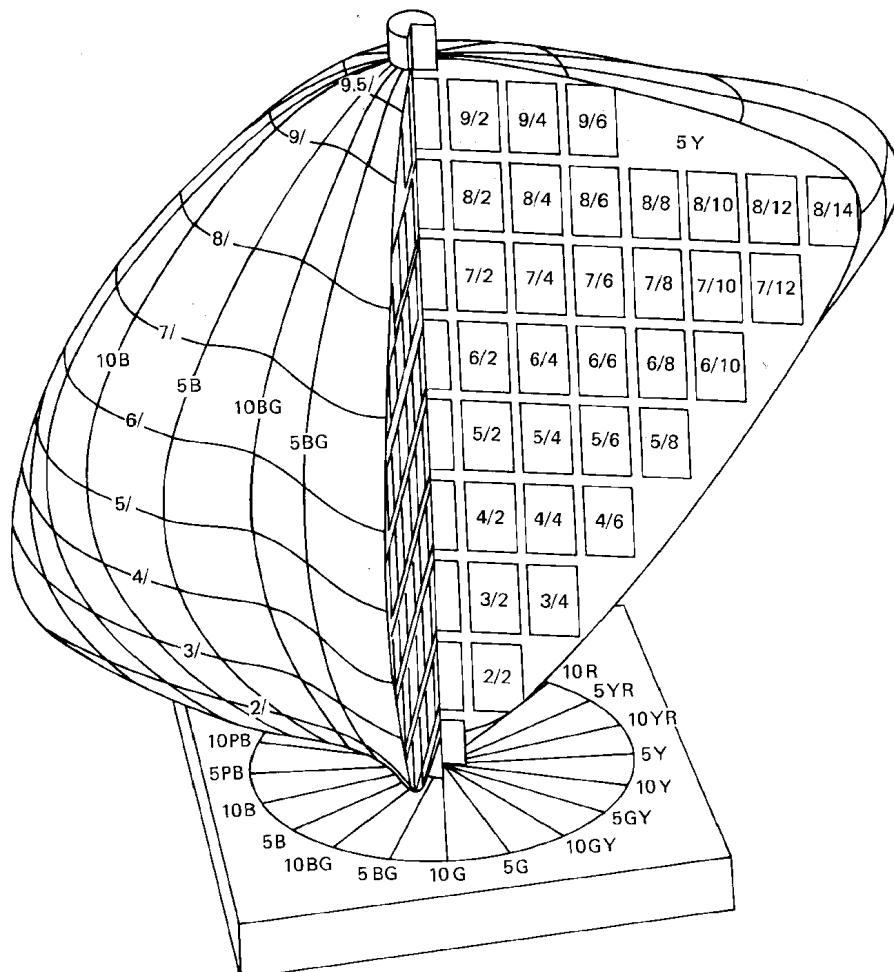
12 Tento systém se často zaměňuje se systémem HLS.



- 1
- 2 Vedle uživatelsky orientovaných barevných systémů se také v praxi používají barevné vzorníky.
- 3

13.3.10. Barevné vzorníky

2 Jedním ze základních barevných vzorníků je Munsellův vzorník, který je vlastně „knihou“ barevných
3 vzorků. Pro každou barvu stránka obsahuje barevná pole o různé sytosti a světlosti. Takže např. barva
4 5Y 7/7 jednoznačně definuje barevnost i světlost. Existují i další vzorníky např. Ostwaldův,
5 Hickethierův a další. Základním požadavkem je, že vzorník musí být dlouhodobě stálobarevný
6 a naprosto přesně barevně reprodukovatelný.



Obr.13.25: Munsellův barevný vzorník

Doporučení

Lze jen doporučit se podívat na <http://www.efg2.com/Lab/Graphics/Colors/>

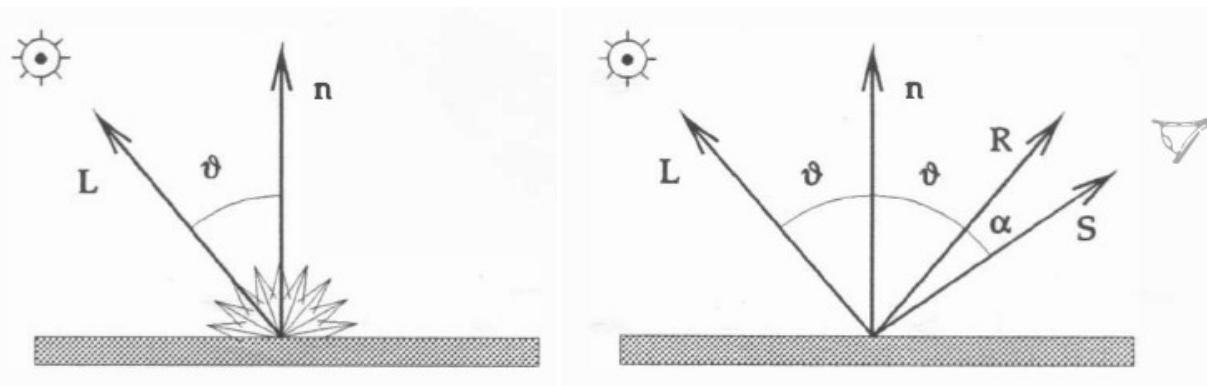
15 Až dosud jsme se zabývali způsoby reprezentace barev a jejich převody. V následujícím textu se
16 budeme zabývat, jak se optické vlastnosti scény včetně barev uplatní při výpočtu zobrazované scény.

14. Modely osvětlení a stínování

Modely osvětlení a stínování hrají podstatnou roli při zobrazování scény. Následně se budeme zabývat především metodami vhodnými pro lokální metody zobrazování povrchu objektů.

14.1. Modely osvětlení

Světlo, které dopadá na povrch fyzikálního tělesa, je částečně pohlceno, odraženo nebo propuštěno. Pokud světlo po dopadu na povrch je prakticky pohlceno, pak se nám povrch jeví jako neviditelný. Pokud těleso odráží všechny vlnové délky stejně, bude se nám povrch jevit jako šedý, v ostatních případech pak jako barevný.



Obr.14.1: Difuzní odraz

Obr.14.2: Zrcadlový odraz

V následujícím textu budeme používat následující značení:

- \hat{x} - normalizovaný vektor x , tj. $\|x\| = 1$
- n – normála plochy
- L – směrový vektor zdroje světla
- R – směrový vektor odraženého paprsku
- S – směrový vektor pozorovatele

V zásadě rozpoznáváme dva „extrémní“ typy odrazu světla od povrchu objektu, a to:

- *rozptylující odraz* (diffuse reflection), viz Obr.14.1, který světlo rozptyluje do všech směrů rovnoměrně a výsledný vjem povrchu je pak matný mající rovnoměrný jas
- *zrcadlový odraz* (specular reflection), viz Obr.14.2, kdy se paprsek odrazí od povrchu a jas je závislý na poloze pozorovatele. Fyzikálně však naprostě přesný odraz není možné realizovat, neboť pozorovatel může vidět zdroj světla, i když je trochu mimo přesný směr odrazu.

Rozptylující odraz

Pro rozptylující odraz platí Lambertův zákon, který určuje intenzitu světla podle úhlu dopadu ϑ na povrch tělesa, ve tvaru:

$$I_d = k_d I_p \cos \vartheta \quad (14.1)$$

kde: k_d je konstanta určující „množství“ odraženého rozptyleného světla $k_d \in (0,1)$, I_p je intenzita bodového zdroje světla, ϑ je úhel dopadu světelného paprsku $\vartheta \in (-\pi/2, \pi/2)$.

Pak lze psát:

$$I_d = k_d I_p \hat{L}^T \hat{n} \quad (14.2)$$

Protože objekty jsou na scéně, která má obecně nějaké světelné pozadí, je nutné ještě respektovat světelnou intenzitu pozadí (ambient light) a tedy:

$$I_d = k_a I_a + k_d I_p \hat{L}^T \hat{n} \quad (14.3)$$

kde: k_a je koeficient určující vliv okolního světla $k_a \in (0,1)$, I_a značí intenzitu okolního světla.

Vliv vzdálenosti od pozorovatele

Pokud by se nyní zobrazovaly dvě stejné rovnoběžné plochy, pak by byly od sebe nerozpoznatelné i při různých vzdálenostech od pozorovatele, neboť není respektován vliv vzdálenosti plochy od zdroje světla. Označíme-li vzdálenost bodu zobrazované plochy od pozorovatele symbolem d , pak můžeme psát:

$$I_d = k_a I_a + (k_d I_p \hat{L}^T \hat{n}) / (k_0 + k_1 d + k_2 d^2) \quad (14.4)$$

kde: k_0, k_1, k_2 jsou konstanty, které určují „útlum“ jasu vlivem vzdálenosti.

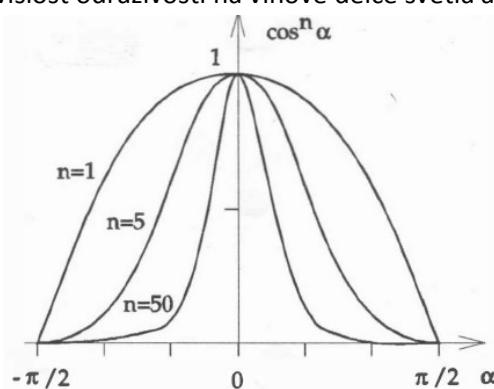
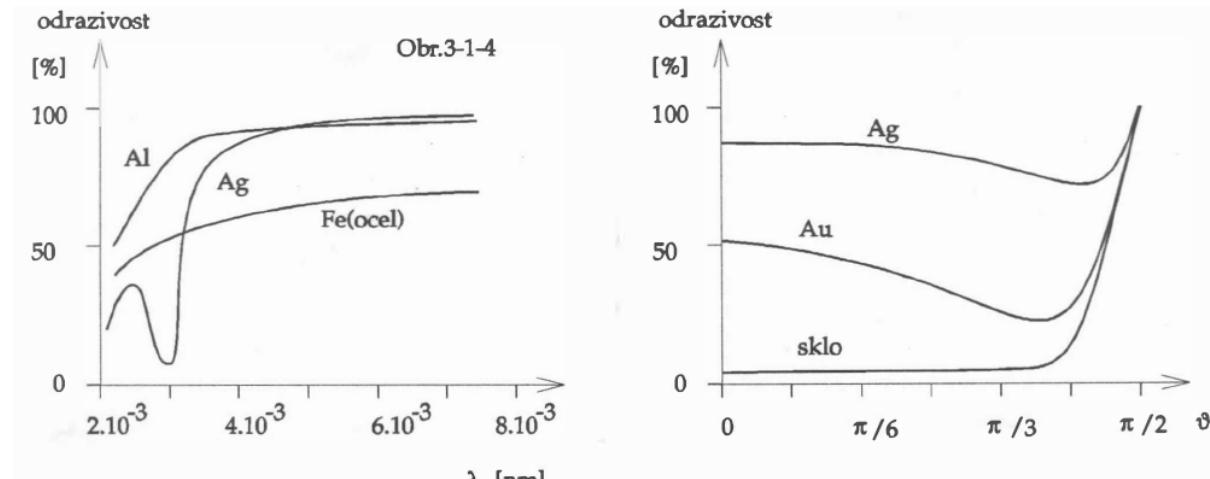
Zrcadlový odraz

Zrcadlová odraz je odrazem přísně směrovým, nicméně pro reálné fyzikální plochy tomu tak není. Proto se pro zrcadlový obraz používá empirický model Bui-Tuong Phonga, který je definován rovnicí:

$$I_s = I_p w(\vartheta, \lambda) \cos^n \alpha \quad (14.5)$$

kde: I_s je intenzita odraženého světla, I_p je intenzita zdroje světla, $w(\vartheta, \lambda)$ je funkce určující množství odraženého světla v závislosti na vlnové délce λ a úhlu dopadu ϑ , α je úhel mezi vektory R a S , n určuje charakteristiku povrchu, tj. jak rychle „mizí“ jev úplného odrazu, $n \in (0, \infty)$.

Vliv vlnové délky světla λ a úhlu dopadu ϑ na odrazivost pro různé materiály je na Obr.14.3.



Obr. 14.4: Vliv koeficientu n

Čím je hodnota n větší, tím se charakter odrazivosti povrchu blíží k ideálnímu zrcadlu. Pro zrcadlové odrazy je obvykle $n \geq 50$.

1

2 Vzhledem ke složitosti funkce $w(\vartheta, \lambda)$ je funkce pro danou vlnovou délku λ nahrazena konstantou k_s
 3 a můžeme tedy psát:

$$I_s = k_s I_p \cos^n \alpha = k_s I_p (\hat{\mathbf{R}}^T \hat{\mathbf{S}})^n \quad (14.6)$$

4 Vliv mocniny n na „rychlosť“ mizení zrcadlového odrazu je na Obr.14.4.

5

6 Takže nyní můžeme pro intenzitu psát:

$$I_s = k_a I_a + I_p (k_d \hat{\mathbf{L}}^T \hat{\mathbf{n}} + k_s (\hat{\mathbf{R}}^T \hat{\mathbf{S}})^n) / (k_0 + k_1 d + k_2 d^2) \quad (14.7)$$

7 Výsledný diagram znázorňující odrazivost je na Obr.14.5 pro různé hodnoty n , přičemž $n_1 > n_2$.

8

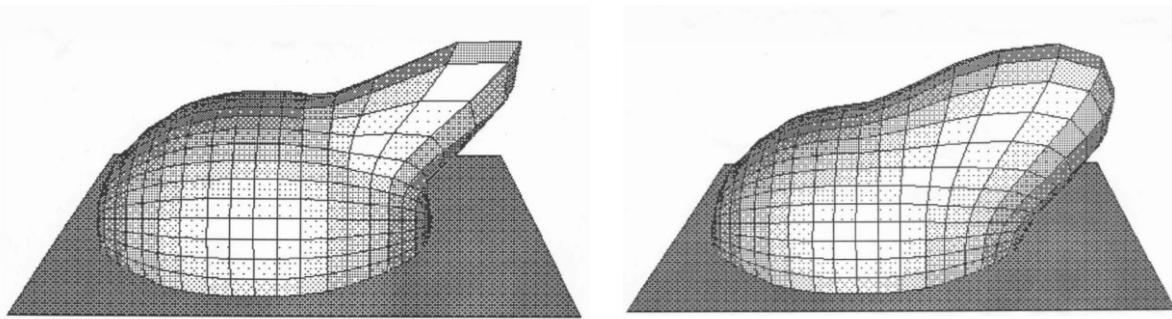


Diagram pro hodnotu n_1

Diagram pro hodnotu n_2

Obr.14.5: Diagram odrazivosti pro různé hodnoty n

9

10 Lom světla je dalším optickým jevem, který je nutné reprezentovat. Lom světla se nevyužívá
 11 v lokálních modelech, ale je vhodné jej zde uvést pro úplnost a jeho využití bude ukázáno později.

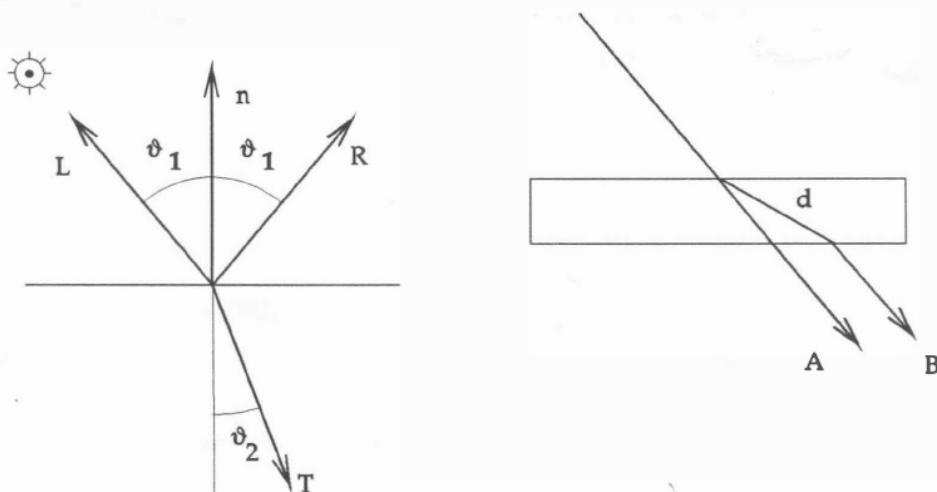
12

1 **Lom světla**

2 Lom světla podle Snellova zákona vzniká na rozhraní dvou matriálů s odlišným indexem lomu. Snellův
3 zákon říká, že platí:

$$\eta_1 \sin \vartheta_1 = \eta_2 \sin \vartheta_2 \quad (14.8)$$

4 kde: η_1 a η_2 jsou indexy lomu dvou optických materiálů a jsou funkcemi vlnové délky λ .



5
6 Obr.14.6: Optický a geometrický paprsek

7
8 V případě reprezentace lomu světla je nutné respektovat, že vedle změny směru paprsku průchodem
9 optického rozhraní, je útlum intenzity závislý exponenciálně na délce dráhy paprsku v daném
10 prostředí, a to:

$$I = I_0 t e^{ad} \quad (14.9)$$

11 kde: a, t jsou konstanty dané optickými vlastnostmi materiálu, d je délka dráhy paprsku v daném
12 prostředí. Hodnota t určuje množství propouštěné intenzity, tj. jak prostředí je průhledné, resp.
13 průsvitné, a hodnota a určuje, jak je světlo rychle pohlcováno.

14
15 V mnoha případech pro zjednodušení výpočtu se optický paprsek nahrazuje paprskem geometrickým,
16 který však nerespektuje lom světla.

17
18 Výsledný iluminační model osvětlení pro zobrazování scén s dobrou věrností je dán pak rovnicí:

$$I = k_a I_a + k_d I_d + k_s I_s + k_t I_t \quad (14.10)$$

19 kde jednotlivé složky určují:

- 20 • $k_a I_a$ vliv okolního světla
- 21 • $k_d I_d$ vliv rozptýleného odrazu
- 22 • $k_s I_s$ vliv odraženého odrazu
- 23 • $k_t I_t$ vliv propuštěného světla

24
25 Až dosud jsme se zabývali modelem osvětlení, u kterého byl požadavek na dobrou reprezentaci
26 optických a fyzikálních vlastností objektů vzhledem k výslednému zobrazování.

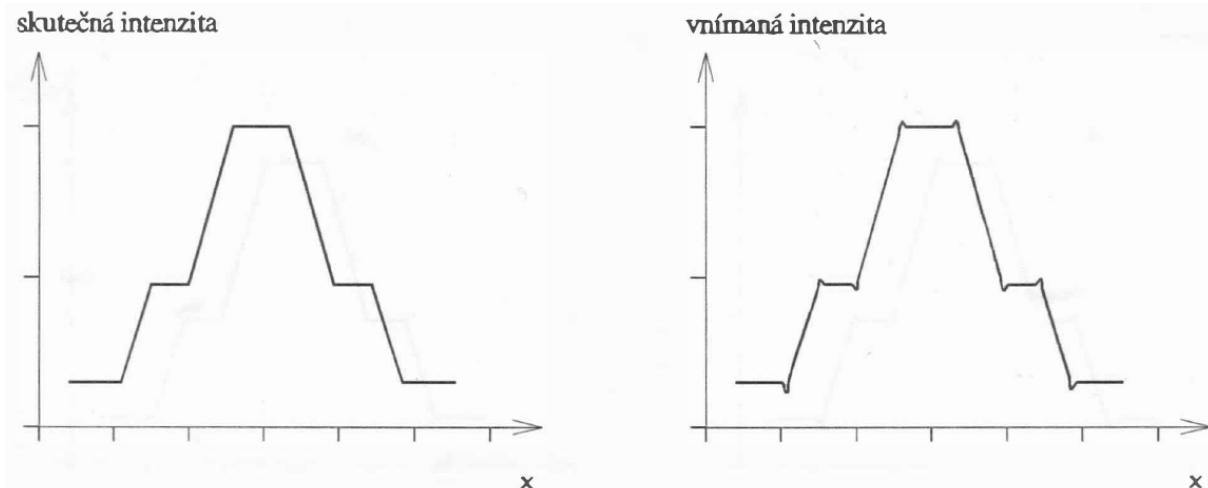
27
28 Povrchy objektů jsou v praxi v převážné většině reprezentovány trojúhelníky a výpočet světelních
29 poměrů je tak prováděn pro lineární plochy, tj. části roviny. Pro tyto případy byly vyvinuty metody
30 lokálního stínování.

14.2. Metody stínování – lokální metody

V převážné většině případů jsou objekty reprezentovány svým povrchem ve formě trojúhelníkové sítě. Představíme-li si např. hladké rotační těleso, pak povrch je reprezentován vlastně mnohostěnem, i když původní těleso bylo hladké. Mnohostěn lze zobrazit tak, že se zobrazí jednotlivé trojúhelníky s použitím dříve uvedených modelů osvětlení. Nicméně pro rotační těleso, které je approximováno mnohostěnem, požadujeme, aby vypadalo hladce. Proto byly vyvinuty lokální metody stínování, které vytvářejí iluzi hladkosti povrchu, i když datová reprezentace povrchu objektu vlastně hladká není.

14.2.1. Konstantní stínování

Konstantní stínování je aplikovatelné na objekty ohraničené lineárními ploškami, např. trojúhelníkovou sítí. Konstantní stínování je velmi často používané pro svoji jednoduchost a rychlosť zobrazování, neboť celý trojúhelník je „vybarven“ stejnou barvou, resp. úrovní šedi. Pro nerovinné povrchy tento postup není příliš vhodný, protože hladká plocha je approximována trojúhelníkovou sítí. Na společné hraně dvou trojúhelníků s různými normálami se intenzita mění skokem a při malých změnách se projevuje Machův efekt, který působí velmi rušivě a pozorovatel vnímá hrany, které na objektu ve skutečnosti nejsou, viz Obr.14.7.



Obr.14.7: Machův efekt – vnímání vlivu změny intenzity

Konstantní stínování vychází ze zjednodušujících předpokladů, a to:

- zdroj světla je v nekonečnu, a tedy $\hat{L}^T \hat{\mathbf{n}}$ je konstantní
- pozorovatel je v nekonečnu, tj. součin $\hat{\mathbf{R}}^T \hat{\mathbf{S}}$ je konstantní

Normála trojúhelníka \mathbf{n} je určena pomocí vektorového součinu, a to:

$$\mathbf{n} = (\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0) \quad (14.11)$$

Poznámka

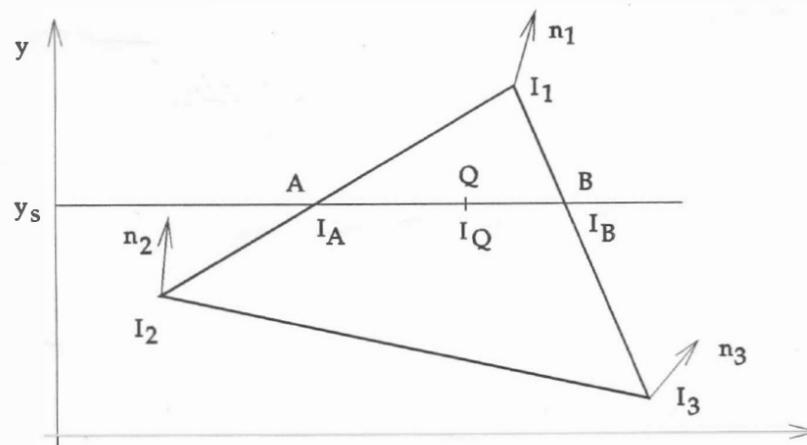
Na GPU je vektorový součin instrukcí, takže výpočet je rychlý.

Vlastní stínování trojúhelníka je realizováno až po promítnutí na průmětně, takže po projekci z $E^3 \rightarrow E^2$. Světelné poměry se však počítají v souřadném systému objektu, tj. ve World Coordinates.

Pro stínování hladkých povrchů je vhodnější použít Gouraud nebo Phong stínování.

1 **14.2.2. Gouraud stínování**

2 Gouraud stínování je založeno na lineární interpolaci intenzit na základě známých intenzit ve
3 vrcholech. Pro jednoduchost si představme situaci na Obr.14.8.

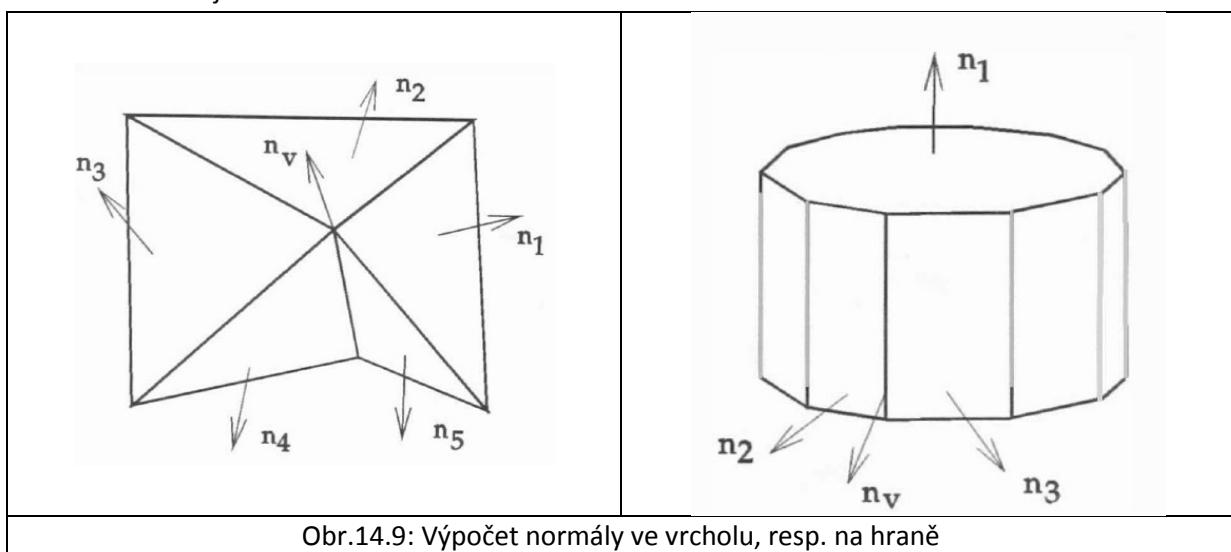


Obr.14.8: Princip stínování Gouraud

4
5
6
7 Ze známých normálových vektorů n_1, n_2, n_3 z modelu osvětlení určíme intenzity světla v jednotlivých
8 vrcholech I_1, I_2, I_3 . Vlastní stínování se realizuje lineární interpolací, kdy se určí intenzita I_A a I_B
9 v bodech A a B , což jsou průsečíky hran s řádkou na obrazovce a pro každý pixel mezi body A a B se
10 opět aplikuje lineární interpolace získání intenzity I_Q v bodě Q . K lineární interpolaci slouží na GPU
11 instrukce *lrep*.

12 Je tedy zřejmé, že výsledný výstup z hlediska intenzity závisí též na geometrických transformacích
13 objektu, které se projeví na obrazovce rotací. Pokud se objekt otočí tak, že po projekci bude rotován
14 např. o úhel $-\pi/2$, budou body A a B obecně na jiných hranách daného trojúhelníka.

15
16 V naprosté většině případů je povrch objektu dán množinou trojúhelníků, resp. trojúhelníkovou sítí
17 bez explicitní znalosti normály ve vrcholech. Je tedy nutné řešit otázku výpočtu normálového vektoru
18 ve vrcholech trojúhelníků.



Obr.14.9: Výpočet normály ve vrcholu, resp. na hraně

1 **Povrch určen trojúhelníkovou sítí**

2 V případě, že je dán povrch trojúhelníkovou sítí, je situace jednodušší, neboť máme informaci, které
 3 trojúhelníky sdílejí daný vrchol. Normálna se pak spočte z normál těchto trojúhelníků. V literatuře lze
 4 nalézt více výpočetních postupů. Jedním z častých postupů je výpočet normály \mathbf{n}_v , ve vrcholu jako
 5 prostého průměru z normál trojúhelníků sdílejících daný vrchol, tj. $q_i = 1$, ať už normalizovaných,
 6 nebo nenormalizovaných normál \mathbf{n}_i , $i = 1, \dots, k$. Pak:

$$\mathbf{n}_v = (q_1 \mathbf{n}_1 + \dots + q_k \mathbf{n}_k) / k \quad (14.12)$$

7 Pokud se v dalším postupu používá normalizovaná normála \mathbf{n}_v , pak lze operaci dělení vynechat
 8 a přímo provést normalizaci. Jiná metoda váží jednotlivé normály podle vrcholového úhlu φ , např.
 9 $q_i = \sin \varphi_i$ pro i -tý trojúhelník atd.

10 Je nutné upozornit, že do výpočtu normály ve vrcholech pro účely stínování se zahrnují jen ty
 11 trojúhelníky, které „*hladce sousedí*“ se stínovanou plochou. Např. v případě válcové plochy
 12 k normálovým vektorům při stínování válcové plochy normála podstavy \mathbf{n}_1 nepřispívá.

14 **Povrch určen jako množina samostatných trojúhelníků**

15 V tomto případě je nezbytné najít pro každý vrchol všechny trojúhelníky, které daný vrchol sdílejí. Je
 16 nutné si uvědomit, že trojúhelníky, které geometricky sdílejí daný vrchol, mají své vrcholy uložené
 17 v paměti odděleně, i když mají stejně souřadnice. Proces, který z množiny samostatných trojúhelníků
 18 vytváří trojúhelníkovou síť, tj. strukturu, kde jsou informace o sousedech, je označován termínem
 19 *rekonstrukce trojúhelníkové sítě*.

21 Algoritmická složitost rekonstrukce trojúhelníkové sítě pro N trojúhelníků závisí na zvoleném postupu
 22 takto:

- 24 • *algoritmus brutální síly* je založen na přímém vzájemném porovnání souřadnic vrcholů všech
 25 dvojic trojúhelníků. Tento postup je evidentně složitosti $O(N^2)$ a je v praxi i pro menší úlohy
 26 nepoužitelný vzhledem k časové náročnosti
- 27 • *aplikace metod řazení* (třídění) je založena na tom, že ze souřadnic x, y, z se udělá klíč a spolu
 28 s identifikátorem trojúhelníka a pořadím vrcholu v trojúhelníku se seřadí a následně se
 29 vytvoří odpovídající struktury sloužící k vytvoření trojúhelníkové sítě. Tento postup je
 30 evidentně složitosti $O(N \lg N)$, komplikovaný a v praxi je i pro střední úlohy nepoužitelný
 31 vzhledem k časové náročnosti kromě jiného
- 32 • *aplikace hash funkce* (geometrický hashing) je založena na hashování souřadnic x, y, z s tím,
 33 že se předpokládá dobré „rozprostření“ v hashovací tabulce. Tento postup při vhodně
 34 zvolené hashovací funkci je složitosti $O(N)$ a je použitelný i pro velké úlohy.
 35 Podrobnější informace viz:
 - 36 ○ Hradek,J., Skala,V.: Hash Function and Triangular Mesh Reconstruction, Vol.29, No.6.,
 37 pp.741-751, Computers&Geosciences, Pergamon Press, ISSN 0098-3004, 2003

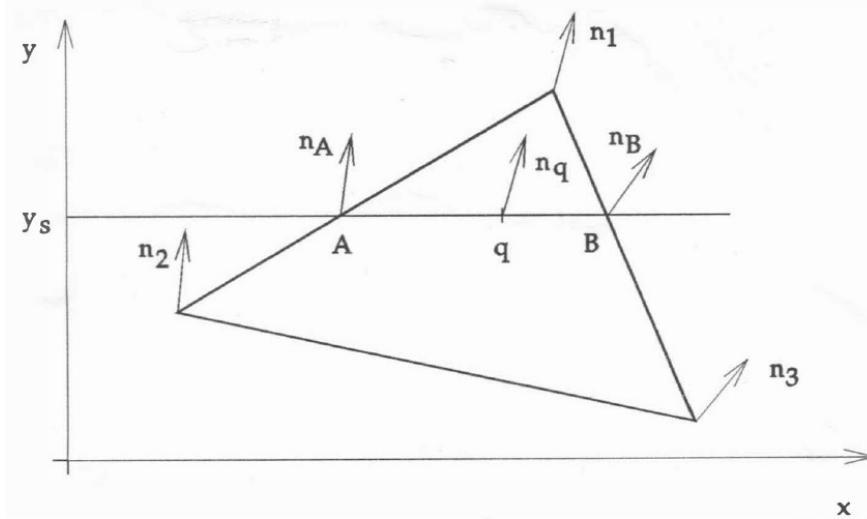
38 **Poznámka**

39 Gouraud stínování je standardně podporováno v grafických akcelerátorech.

41 Lepších výsledků při stínování se dosáhne pomocí stínování Phong, které interpoluje normálové
 42 vektory místo interpolace intenzit.

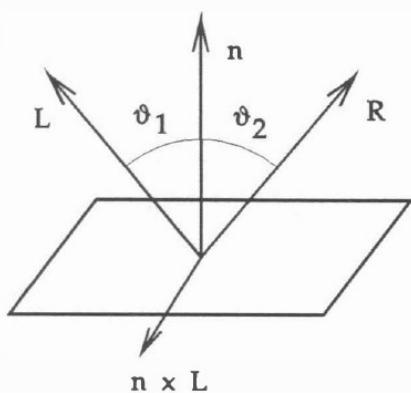
1 **14.2.3. Phong stínování**

2 Výsledky při použití Gouraudovo stínovaní dávají podstatně lepší výsledky než konstantní stínování,
 3 avšak pro hladké oblé povrchy se projevuje to, že se interpolují pouze intenzity světla. Phong navrhl
 4 metodu stínování, která je založena na interpolaci normál, z nichž se následně intenzita určuje, viz
 5 Obr.14.10.

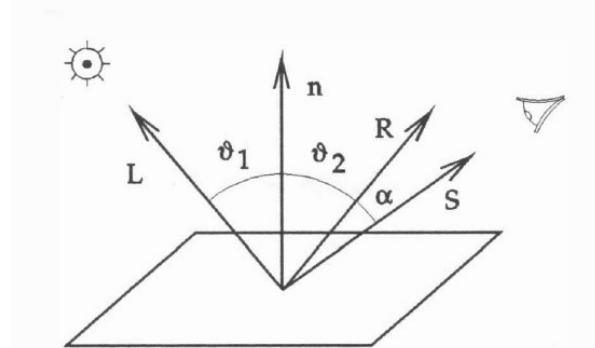


Obr.14.10: Phong stínování – interpolace normových vektorů

9 Tato metoda dává velmi dobré výsledky, avšak je výpočetně náročná, neboť je nutné počítat
 10 normálový vektor pro každý bod plochy a následně také světelné poměry, tj. je nutné respektovat
 11 výpočetní postupy pro osvětlení, viz kap. 14.1 (Modely osvětlení) a reálný zrcadlový odraz Obr.14.12.
 12



Obr.14.11: Ideální zrcadlový odraz



Obr.14.12: Zrcadlový odraz respektující reality

13 **Upozornění**

14 Phong stínování není standardně podporováno hardwarovými akcelerátory.

15

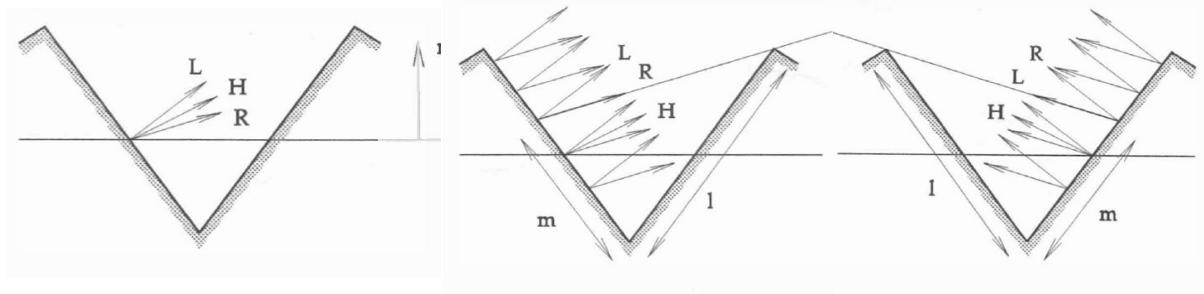
16 **Poznámka**

17 Konkrétní postupy a příklady výpočtu osvětlení lze nalézt:

- 18 • Skala,V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia, 1992

14.2.4. Torrance-Sparrow stínování a Blinn modifikace

Až dosud jsme se zabývali metodami, které ne příliš dobře reprezentovaly fyzikální realitu, neboť vlastně kromě optických vlastností povrchu jsou určující také mechanické vlastnosti povrchu, např. drsnost povrchu atd. V roce 1967 Torrance a Sparrow navrhli model, který velmi dobře reprezentuje experimentálně získané hodnoty, avšak je výpočetně velmi náročný. Model vychází z představy, že povrch se skládá z mikroplošek, které jsou orientovány náhodně a celkový odraz na ploše se určí jako celkový příspěvek mikroplošek, viz Obr.14.13.



a)

b)

c)

Obr.14.13: Speciální případy v modelu Torrance-Sparrow

8

V zásadě mohou nastat následující případy:

- 10 • nenastává interference paprsku s jinou mikroploškou, viz Obr.14.13.a
- 11 • dochází k „maskování“ odraženého světla, viz Obr.14.13.b
- 12 • dochází k zastínění části mikroplošky, viz Obr.14.13.c

13 Pro další výklad použijeme následné značení vektorů a ostatních veličin:

14 **n** - normála plochy bez uvažování mikroplošek

15 **L** – směrový vektor světla

16 **R** – směrový vektor odraženého paprsku

17 **S** – směrový vektor pozorovatele

18 **H** – normála mikroplošky

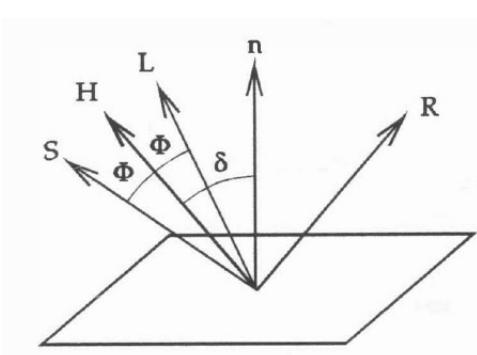
19 **\hat{T}** – normalizovaný vektor **T**

20 **l** – délka hrany mikroplošky – parametr

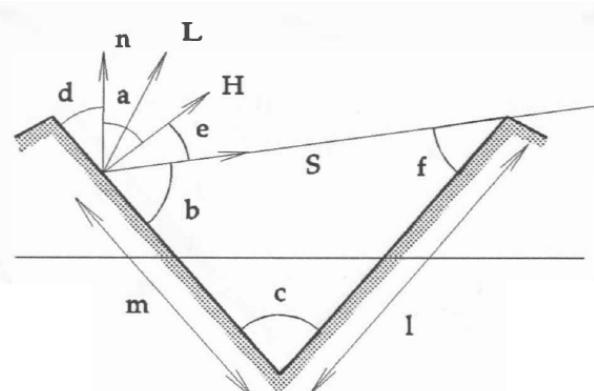
21 **m** – určuje část mikroplošky, která se podílí na osvětlení sousední mikroplošky, viz Obr.14.13.b, resp.

22 zastíněnou část mikroplošky, viz obr. Obr.14.13.c.

23



Obr.14.14: Určení vektoru odrazu



Obr.14.15: Blinnovo zjednodušení modelu

1 Z Obr.14.14 vyplývá, že:

$$\cos \phi = \hat{\mathbf{H}}^T \hat{\mathbf{L}} = \hat{\mathbf{S}}^T \hat{\mathbf{H}} \quad \mathbf{H} = \frac{\mathbf{L} + \mathbf{S}}{|\mathbf{L} + \mathbf{S}|} \quad (14.13)$$

2 To znamená, že pouze mikroploška normálou \mathbf{H} může přispět k celkové intenzitě vnímané
3 pozorovatelem. Torrance a Sparrow navrhli pro výpočet koeficientu odrazu k_s následující vztah:

$$k_s = \frac{DGF}{\hat{\mathbf{n}}^T \hat{\mathbf{S}}} \quad (14.14)$$

4 kde: D je distribuční funkce směrů mikroplošek daného povrchu, F je koeficient odrazivosti
5 respektující Fresnelův zákon odrazu, G je funkce určující zeslabení intenzity vlivem zastínění
6 a maskování.

7 Blinn zjednodušil model pro případ, kdy plošky mají tvar symetrického „V“, viz Obr.14.15, neboť pak
9 platí:

$$\begin{aligned} \frac{m}{l} &= \frac{\sin f}{\sin b} & d + a &= \pi/2 \\ c &= 2d & f + b + c &= \pi \end{aligned} \quad (14.15)$$

10 Pak lze ukázat, že:

$$\frac{m}{l} = \frac{\hat{\mathbf{H}}^T \hat{\mathbf{S}} - 2(\hat{\mathbf{n}}^T \hat{\mathbf{H}})(\hat{\mathbf{n}}^T \hat{\mathbf{S}})}{\hat{\mathbf{H}}^T \hat{\mathbf{S}}} \quad (14.16)$$

11 Označíme-li:

$$G_m = \frac{2(\hat{\mathbf{n}}^T \hat{\mathbf{H}})(\hat{\mathbf{n}}^T \hat{\mathbf{S}})}{\hat{\mathbf{H}}^T \hat{\mathbf{S}}} \quad (14.17)$$

12 pak poměrné zeslabení intenzity vnímané pozorovatelem lze vyjádřit poměrem $(l - m)/l$ a tedy:

$$1 - \frac{m}{l} = G_m$$

13 Pro případ zastínění pak analogicky platí:

$$G_s = \frac{2(\hat{\mathbf{n}}^T \hat{\mathbf{H}})(\hat{\mathbf{n}}^T \hat{\mathbf{L}})}{\hat{\mathbf{H}}^T \hat{\mathbf{S}}} \quad (14.18)$$

14 Pak poměrné zeslabení intenzity vnímané pozorovatelem lze vyjádřit poměrem $(l - m)/l$ a tedy:

$$1 - \frac{m}{l} = G_s \quad (14.19)$$

15 Celkové zeslabení intenzity je pak určeno vztahem:

$$G = \min\{1, G_m, G_s\} \quad (14.20)$$

16 Distribuční funkce D v Phong modelu byla reprezentována funkcí $\cos^n \delta$.

17

18 V Torrance-Sparrow modelu se předpokládá Gaussovo rozložení, tj.:

$$D = c_1 e^{-(\delta c)^2} \quad (14.21)$$

19 kde: D je počet mikroplošek orientovaných ve směru odchýleném o úhel δ od normály plochy \mathbf{n} ,
20 c, c_1 jsou volitelné konstanty určující „drsnost“ povrchu.

21

22 Beckmann ukázal, že hodnotu D lze též určit jako:

$$D = \frac{1}{m^2 \cos^4 \delta} e^{-(\tan \delta/m^2)^2} \quad (14.22)$$

23 přičemž není nutné volit žádnou konstantu.

24

25 Je vhodné upozornit, že existují i jiné distribuční funkce vhodné pro určité materiály, např. pro sklo,
26 viz Trowbridge-Reitz.

1 Pro výpočet konstanty k_s se určí hodnota odrazivosti F podle Fresnelova zákona:

$$F = \frac{1}{2} \left(\frac{\sin^2(\phi - \vartheta)}{\sin^2(\phi + \vartheta)} + \frac{\tan^2(\phi - \vartheta)}{\tan^2(\phi + \vartheta)} \right) \quad (14.23)$$

2 kde: ϕ je incidenční úhel, η je index odrazu a

$$\cos \phi = \hat{\mathbf{L}}^T \hat{\mathbf{H}} \sin \vartheta = \sin \phi / \eta \quad (14.24)$$

3 Vzhledem k tomu, že index odrazu η je závislý na vlnové délce světla λ , tj. $\eta = \eta(\lambda)$, je též F funkcí
4 vlnové délky. Funkci $\eta(\lambda)$ lze určit experimentálně.

5

6 Cook a Torrance navrhli funkci F ve tvaru:

$$F = \frac{1}{2} \left(\frac{c-g}{c+g} \right)^2 \left\{ 1 + \frac{[c(g+c)-1]^2}{[c(g-c)+1]^2} \right\} \quad (14.25)$$

7 kde:

$$c = \hat{\mathbf{L}}^T \hat{\mathbf{H}} \quad g^2 = \eta^2 + c^2 - 1 \quad (14.26)$$

8 Pro $\phi = 0, c = 1, g = \eta$ lze ukázat, že $F_0 = F_{\phi=0}$, že

$$F_0 = \left(\frac{\eta - 1}{\eta + 1} \right)^2 \quad (14.27)$$

9 a je-li funkce F_0 známa, pak:

$$\eta(\lambda) = \frac{1 + \sqrt{F_0(\lambda)}}{1 - \sqrt{F_0(\lambda)}} \quad (14.28)$$

10 Uvedený postup je nutné aplikovat pro každou mikroplošku. Celkový odraz na ploše pak určíme jako
11 celkový příspěvek všech mikroplošek, a to:

$$I = k_a I_a + \sum_{j=1}^q I_{p_j} \left[k_d (\hat{\mathbf{n}}^T \hat{\mathbf{L}}_j) + \frac{D_j G_j F_j}{\hat{\mathbf{n}}^T \hat{\mathbf{S}}} \right] \quad (14.29)$$

12 kde: q je počet zdrojů světla. Intenzita světla je následně zmenšena podle vzdálenosti od
13 pozorovatele.

14

15 Výše uvedené vztahy jsou i přes svoji výpočetní náročnost používány v komerčních systémech.

16

17 Podrobnější informace a nové iluminační modely lze nalézt např. v:

- 18 • Glassner,A.: Principles of Digital Image Synthesis, Vol.I a Vol.II Morgan Kaufmann, 1995
19 ([CLICK off-line](#))

20 Reference

- 21 • Skala,V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia, 1993
22 ([CLICK off-line](#))
- 23 • Beckmann,P., Spizzichino,A.: Scattering of Electromagnetic Waves from Rough Surfaces,
24 MacMillan Press, New York, pp.1-33, 70-98, 1963
- 25 • Blinn,J.F.: Model of Light Reflection for Computer Synthesized Pictures, Computer Graphics,
26 SIGGRAPH 1977, Vol.11, pp.191-198, 1997
- 27 • Cook,R.L., Torrance,K.E.: A Reflectance Model for Computer Graphics, ACM on Graphics,
28 Vol.1, pp.7-24, 1982
- 29 • Trowbridge,T.S., Reitz,K.P.: Average irregularity representation of roughened surfaces for ray
30 reflection, Journal of Optical Society of America, Vol.65, pp.531-536, 1975

31

1
2
3
4
5
6
7
8
9

10 Dosud jsme se zabývali lokálními metodami stínování, které jsou v zásadě jednoduché a výpočetně
11 nenáročné. Phong stínování není běžně realizováno v grafických akcelerátorech z důvodů výpočetní
12 složitosti.

13
14 V následující části se budeme zabývat základními globálními metodami zobrazování scény. Tyto
15 metody jsou podstatně výpočetně náročnější, a to jak z hlediska časové náročnosti, tak i z hlediska
16 náročnosti paměťové.

17

15. Globální metody

2 Globální metody zobrazování scény se snaží o maximální respektování optických poměrů ve scéně a
3 jsou založeny na různých modelech. Lokální metody stínování prezentované v předchozí části jsou
4 vlastně založeny na přímém zobrazování geometrických elementů, většinou trojúhelníků a
5 trojúhelníkových sítí, včetně jejich překryvání, plnění barvou nebo vzorem, řešení viditelnosti atd.

6
7 V následujícím výkladu si uvedeme alespoň dvě základní techniky, a to:

- 8 • **metodu sledování paprsku** (Ray Tracing), která je založena na reprezentaci chování
9 optického paprsku na optickém rozhraní, tj. paprsek se odráží na rozhraní dvou materiálů a
10 propouští se do prostředí druhého, pokud je průhledný nebo průsvitný.

11 Metoda sledování paprsku dobře zpracovává scény se zrcadlovým charakterem nebo
12 s objekty průhlednými a scény s bodovými zdroji světla. *Při změně pozice pozorovatele se
13 však celý výpočetní proces musí opakovat.* Typickým softwarovým představitelem je např.

14 **POV-Ray**: Persistence of Vision Raytracer <http://www.povray.org/>

- 16 • **radiační metodu** (Radiosity), která je založena na energetické bilanci ve scéně, kdy každá
17 elementární ploška přijímá a vydává nějakou energii.

18 Radiační metoda velmi dobře zpracovává scény s difuzním odrazem a plošnými zdroji světla.
19 Protože je založena na energetické bilanci, energetické poměry ve scéně se spočtou pouze
20 jednou, *při změně pozice pozorovatele se pouze jednotlivé plošky odpovídajícím způsobem
21 zobrazují.*

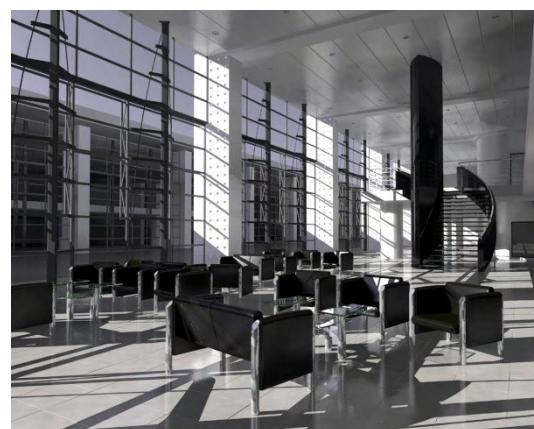
22 Typickým softwarovým představitelem je např.

23 **Radiance** <http://radsite.lbl.gov/radiance/download.html>

24 Obě metody jsou „hraniční“ metody z hlediska optického modelu. Radiační metoda není primárně
25 vhodná pro zrcadlové odrazy, metoda sledování paprsku není vhodná pro difuzní odrazy.



Obr.15.1: Výstup z PovRay – Bonsai Girl



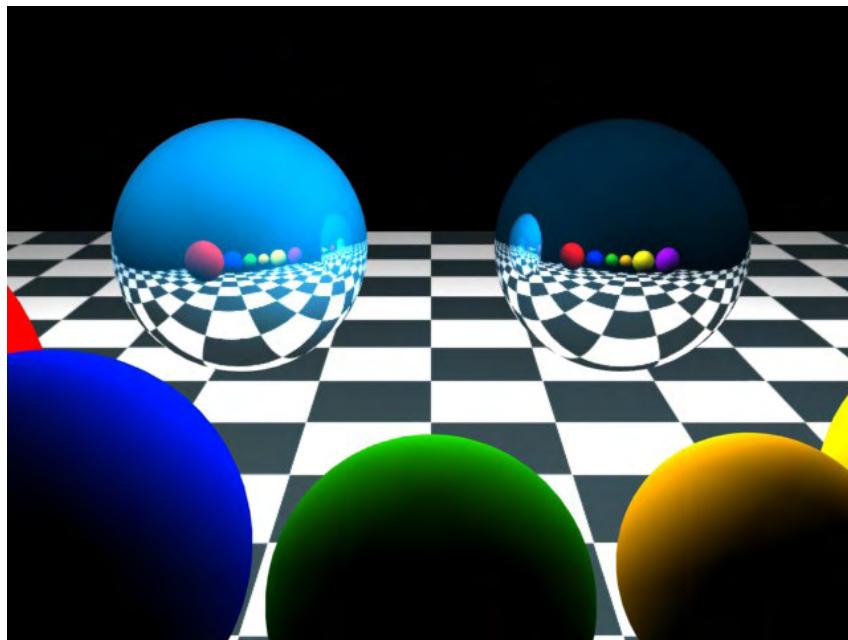
Obr.15.2: Výstup z Radiance - Lobby

27
28 V současné době je možné řešit i velmi složité scény. Je nutné poznamenat a modifikace obou metod
29 dnes umějí řešit jak difuzní, tak i zrcadlový odraz.

30
31 **Upozornění:** Výpočetní náročnost obou globálních metod je nepoměrně vyšší než u metod lokálních.

15.1. Metoda sledování paprsku

Metoda sledování paprsku (Ray tracing) je založena na geometrické představě, že z oka pozorovatele se „vypustí“ paprsek (Ray), který protne danou pozici na obrazovce, tj. daný pixel na pozici (i, j) , a hledá se nejbližší průsečík paprsku se všemi tělesy ve scéně. Po nalezení nejbližšího průsečíku paprsku s povrchem tělesa se paprsek „rozštěpí“ na paprsek odražený od povrchu tělesa podle fyzikálního zákona o odrazu a na paprsek do tělesa propuštěný podle zákona o indexu lomu. Každý z těchto paprsků je dále nezávislý a celý proces se opakuje, pokud není zastaven, např. hloubkou stromu apod.



Obr.15.3: Scéna se šachovnicí demnotrující lom optického světelného paprsku
http://www.neilblevins.com/cg_education/metal_and_refs/metal_and_refs.htm

Vytváří se tak vlastně stromová struktura s průsečíky. Po skončení této fáze výpočtu průsečíků se určí světelné poměry průsečíků paprsku a plochy odpovídajících listům stromu a zpětně se pak počítají světelné poměry průsečíků odpovídajících nadřízeným uzelům stromu, až se určí intenzita daného pixelu.

Z uvedeného je zřejmé, že metoda sledování paprsku:

- umožňuje ve scéně dobře respektovat optické vlastnosti objektů, tj. průhlednost, průsvitnost, index lomu světla, barvu atd., viz Obr.15.3
- je snadno paralelizovatelná, neboť každý paprsek „žije“ nezávislým životem
- dokáže respektovat charakteristiky světelných bodových zdrojů světla
- má velkou výpočetní složitost, a to $O(M N^2 2^k)$, kde: M je počet objektů ve scéně, $N \times N$ je rozlišení výstupu k je počet úrovní výpočetního stromu.

Pro rozlišení $N = 1024 = 2^{10}$, $M = 2^{20}$ (cca 1 mil. objektů) a $k = 2^3$ se bude počítat cca:

$$2^{10} \cdot 2^{10} \cdot 2^{20} \cdot 2^3 = 2^{43} \approx 8.8 \cdot 10^{12}$$

průsečíků paprsku s objekty, resp. obalovými tělesy! Takže, pokud by výpočet jednoho průsečíku trval 0.1 ms, pak by výpočet trval cca $2.4 \cdot 10^5$ hod.!

Je nutné zdůraznit, že výpočetní nároky rostou s kvadrátem rozlišení generovaného obrazu a exponenciálně s hloubkou výpočetního stromu.

1 Algoritmus s detailním výpočtem pro jeden paprsek, viz:

- 2 • Skala,V. Algoritmy počítačové grafiky III, Plzeň, 2011, str.94-107
 3 ([CLICK off-line](#))
 4 • Skala,V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia, 1993
 5 ([CLICK off-line](#))

6 **Poznámka**

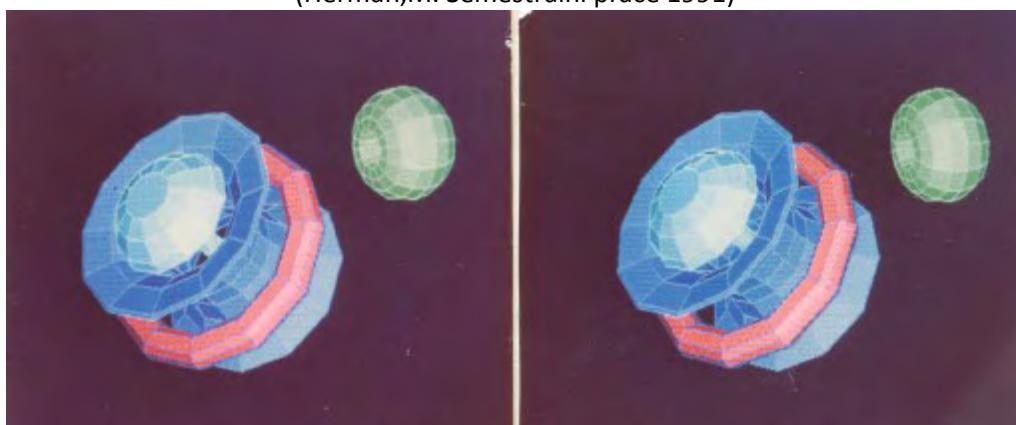
7 První experimenty s metodou sledování paprsku na ZČU (vlastně na jejím předchůdci VŠSE - Vysoké
 8 škole strojní a elektrotechnické v Plzni) byly již v roce 1976 (tehdy se jednalo jen o tzv. primární ray
 9 tracing s použitím CSG stromů a objektů definovaných implicitní funkcí), kdy nebyly k dispozici
 10 odpovídající výpočetní kapacity a ani možnosti grafického výstupu.

11 První reálné grafické výstupy byly realizovány v roce 1991 na počítačích s 64 KB paměti, s připojeným
 12 mikropočítačem s 8 bitovým procesorem I8080 (Intel) bez hardwarové podpory operací v pohyblivé
 13 řádové čárce a výstupem na TV obrazovku 256x256 pixelů se 16 barvami.
 14

15 Experimenty s metodou sledování paprsku realizované jako semestrální práce předmětu
 ekvivalentního ZPG s použitím stereoskopie k získání 3D vjemu (1991).



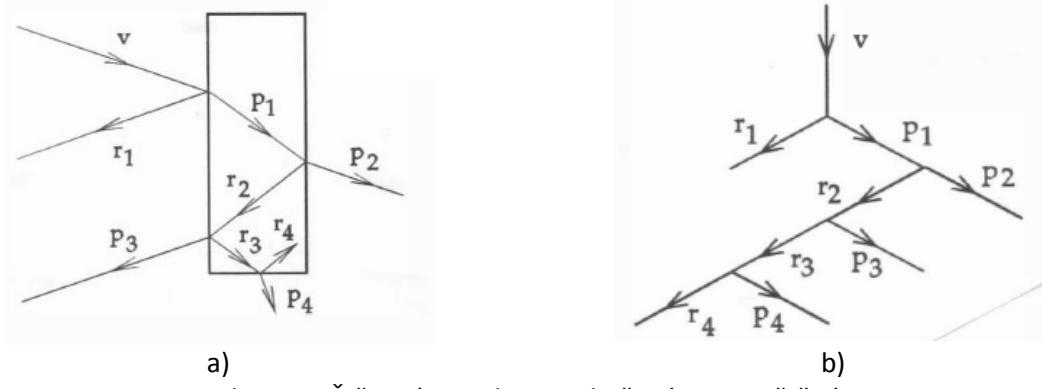
Obr.15.4: Stereoskopický výstup s implicitně definovanými objekty
 (Heřman,M. Semestrální práce 1991)



Obr.15.5: Množinové operace s objekty a se stereoskopickým výstupem

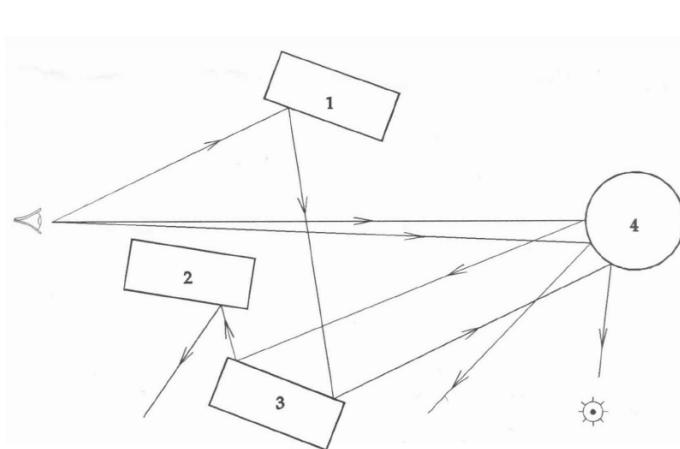
16
 17 Pro realizaci stereoskopického výstupu je nutné vlastně generovat dva výstupní obrazy, tj. pro dvě
 18 různé pozice kamery, resp. pro levé a pravé oko, s odpovídající disparitou, viz kap.5.5 (Stereoskopická
 19 projekce).

1 Představme si pro jednoduchost extrémně jednoduchou scénu, kdy paprsek v , který je dán pozicí
 2 pozorovatele a příslušného pixelu na průmětně, dopadá na kvádr, který je obecně průhledný s jinými
 3 optickými vlastnostmi, tj. s odlišným indexem lomu. Tento paprsek se na povrchu „rozštěpí“ na
 4 paprsek odražený r_1 a propuštěný p_1 , viz Obr.15.6. Paprsky r_1 a p_1 jsou nadále na sobě nezávislé a
 5 zase se odrázejí, resp. pronikají z daného optického prostředí do jiného. Je tedy zřejmé, že se vytváří
 6 vlastně binární výpočetní strom, viz Obr.15.6.b.

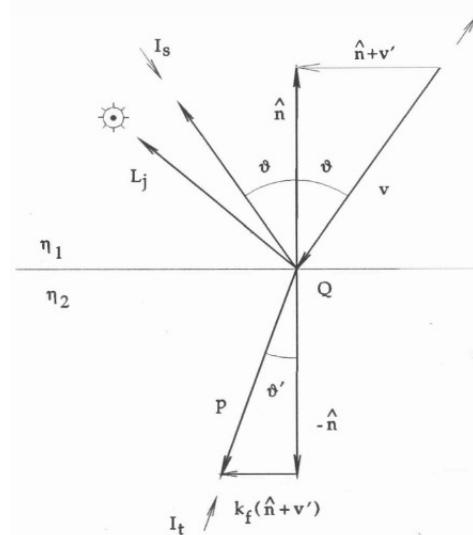


Obr.15.6: Štěpení paprsku na odražený a propuštěný

7 Pro reálnou scénu, která obsahuje více objektů a zdroje světla, je již výpočetní proces
 8 komplikovanější, viz Obr.15.8. Je zřejmé, že pozorovatel uvidí, pokud jsou povrchy tělesa 1 a tělesa 3
 9 zrcadlové, zdroj světla a povrch tělesa 3, které jsou pro pozorovatele zakryté, např. tělesem 2, resp.
 10 neviditelné, neboť jejich plocha je odvrácená od pozorovatele.



Obr.15.7: Schéma scény se světelným zdrojem



Obr.15.8: Geometrie na rozhraní

12 Primární paprsek je určen pozicí pozorovatele a daného pixelu na průmětně, a to:
 13
$$\mathbf{x}(t) = \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A)t = \mathbf{x}_A + st \quad t \in \langle 0, \infty \rangle \quad (15.1)$$

 14 kde: \mathbf{x}_A je pozice pozorovatele a \mathbf{x}_B je pozice pixelu na průmětně v daném souřadném systému.
 15 Nejdříve je nutné najít nejbližší průsečík paprsku s objekty ve scéně. Je zřejmé, že tento proces je
 16 výpočetně poměrně náročný, neboť se provádí pro N^2 pixelů a scéna má M objektů. Navíc po
 17 dopadu paprsku na optické rozhraní se paprsek rozštěpí na paprsky dva, což při k optických rozhraní,
 18 na něž paprsek a jeho potomci narazí, se počítá 2^k průsečíků. Je tedy zřejmé, že je nutné aplikovat
 19 některé akcelerační techniky.

1 Mezi základní akcelerační techniky lze řadit:

- 2 • **obalová tělesa** pro rychlou detekci průsečíku paprsku s objektem. Účelem je rychlá detekce,
3 zda může existovat průsečík paprsku s daným objektem. Obalové těleso, resp. jejich
4 kombinace, by mělo co nejlépe vystihovat tvar daného objektu, aby pravděpodobnost toho,
5 že byl detekován možný průsečík paprsku s objektem a existencí takového průsečíku, byla co
6 nejvyšší. Jako obalová tělesa se používá koule (sphere), která je rotačně invariantní, nebo
7 osově orientovaný kvádr (Axis Aligned Bounding Box – AABB), resp. jejich kombinace, viz
8 kap.7.9 (Algoritmy výpočtu průsečíků a ohraničující tělesa).

9 Je nutné si uvědomit, že může nastat situace, kdy detekce s obalovým tělesem detekuje
10 možný průsečík s tělesem, průsečík opravdu existuje, ale jiné menší těleso má průsečík blíže
11 k počátku paprsku.

- 12 • **dělení prostoru** scény E^3 , kdy každý element prostoru má informaci o objektech, které mají
13 společnou část s daným prostorem. Jde vlastně o formu předzpracování, což v důsledku
14 obvykle vede ke snížení výpočetních nároků. Informace o incidujících objektech je pak
15 uložena v seznamu pro každý prostorový element. Trik spočívá v tom, že se určí, které
16 prostorové elementy paprsek protíná ve smyslu rostoucí hodnoty parametru t a na průsečík
17 se testují pouze ty objekty, které jsou pro daný prostorový element v seznamu uvedeny. Je
18 zřejmé, že pro dělení prostoru můžeme také použít hierarchické datové struktury.

19 Je nutné si uvědomit, že pokud prostor scény rozdělíme na $K \times K \times K$ elementů, pak je
20 nutno pro každý element otestovat, zda s daným prostorovým elementem má nějakou
21 společnou část a pro datovou strukturu je nutno alokovat paměť $O(qK^3)$, kde q je průměrný
22 počet v daném prostorovém elementu. Tento krok má výpočetní složitost $O(MK^3)$ a
23 paměťovou složitost $O(qK^3)$, kde M je počet objektů ve scéně.

24 Také pokud $K = 32 = 2^5$ a $M = 2^{20} \cong 10^6$, pak tento krok má výpočetní složitost
25 $O(MK^3) = c 2^{35}$, kde konstanta $c > 0$.

26 Viz kap.7.4 (Dělení prostoru a Binární masky).

- 27 • **kohherence paprsků**, která je založena na tom, že se průsečíky příliš neliší pro sousední
28 paprsky.

29 V odborné literatuře lze nalézt mnoho dalších urychlovacích technik.

1 15.2. Základní algoritmus Ray-tracing

```

{ xmax, ymax - maximální prostor adresace }

for x := 0 to xmax do
    for y := 0 to ymax do
        begin indikátor průsečíku := false;
            typ paprsku:=v;
            číslo paprsku:=0; číslo generace:=0;
            It:=0; Is:=0; d:=0;
            generuj paprsek v pro daný pixel;
            ulož paprsek v do zásobníku;
            while not zásobník = prázdný do
                begin { Vypočti intenzitu }
                    SOLVE RAY ( I )
                end;
                Display pixel x,y s intenzitou I
            end;
procedure SOLVE RAY ( var I: real);
{ deklarace lokálních proměnných }
begin Vyber paprsek ze zásobníku;
    if indikátor průsečíku = true then
        begin
            Vypočti intenzitu(I);
            if typ paprsku = v
                then EXIT
            else
                begin
                    if typ paprsku = r
                        then nastav Is v zásobníku pro zdrojový paprsek
                        else nastav It v zásobníku pro zdrojový paprsek;
                    Dekrementuj číslo generace;
                    EXIT
                end
        end
    else
        if not průsečík { není průsečík se žádnou plochou } then
            begin
                if typ paprsku = v then
                    begin I:= barva pozadí;
                        Vyber paprsek ze zásobníku
                    end
            end

```

```

    else
        if typ paprsku = r
            then Nastav  $I_s$  v zásobníku pro zdrojový paprsek
            else Nastav  $I_t$  v zásobníku pro zdrojový paprsek;
    EXIT
end
else
    if číslo generace = Stackmax { zásobník je plný ? } then
        begin
             $I_s := 0$ ;  $I_t := 0$ ; d := 1;
            Vypočti intenzitu(I);
            if typ paprsku = r
                then Nastav  $I_s$  v zásobníku pro zdrojový paprsek
                else Nastav  $I_t$  v zásobníku pro zdrojový paprsek;
            EXIT
        end
    else
        begin
            Vypočti vzdálenost d mezi průsečíky paprsku;
            indikátor průsedčíku := true;
            Zvyš číslo generace paprsku;
            Vypočti odražený a propuštěný paprsek;
            { propuštěný paprsek je nutné určit pouze }
            { v případě, že se respektuje lom světla }
            if odražený paprsek existuje
                then
                    begin Zvyš číslo generace paprsku;
                        ulož odražený paprsek do zásobníku
                    end;
            if propuštěný paprsek existuje
                then
                    begin Zvyš číslo generace paprsku;
                        ulož propuštěný paprsek do zásobníku
                    end;
            end;
        EXIT
    end;

```

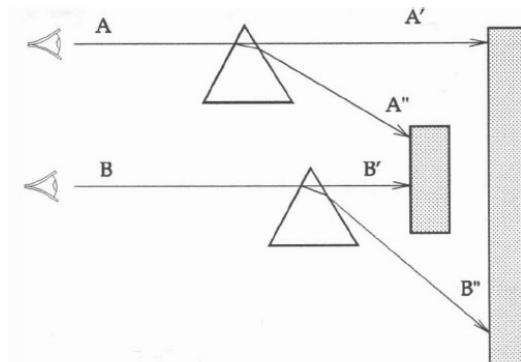
end;

1
2
3
4
5
6

Alg.XX: Základní algoritmus sledování paprsku

1 Respektování indexu lomu světla je důležité pro věrnost zobrazované scény, viz Obr.15.9, kdy
2 pozorovatel vidí při respektování indexu lomu světla naprosto jinou situaci, než v případě
3 nerespektování indexu lomu světla.

4
5



6
7 Obr.15.9: Vliv respektování lomu světla
8

9 Reference

- 10 • Shirley,P., Morley,K.R.: Realistic Ray Tracing, A.K.Peters, 2003
11 • Janda,M.: Multispectral Rendering, diplomová práce (vedoucí Skala,V.), ZČU, Plzeň, 2004

12
13

1 **15.3. Příklad**

Určete intenzitu vnímanou pozorovatelem, je-li dána jednoduchá scéna s plochami kolmými na rovinu zx , viz obr. 3.4.3. Pozorovatel je v nekonečnu ve směru osy z , přičemž $x = 5$. Uvažme dále jeden bodový zdroj světla v poloze $x = 3$, $y = 0$, $z = 10$ a nechť plochy jsou definovány rovnicemi, viz [109]:

$$\begin{array}{ll} 1: & x + z - 12,5 = 0 & 4 \leq x \leq 6 \\ 2: & x - z - 2 = 0 & 4 \leq x \leq 6 \\ 3: & x - 3z + 9 = 0 & 1 \leq x \leq 3 \end{array}$$

Jejich iluminační charakteristiky jsou určeny jednotlivými koeficienty:

$$\begin{array}{lllll} 1: & k_{a_1} = 0,15 & k_{d_1} = 0,15 & k_{s_1} = 0,8 & k_{t_1} = 0,5 & k_{\eta_1} = 1/1,1 \\ 2: & k_{a_2} = 0,15 & k_{d_2} = 0,15 & k_{s_2} = 0,8 & k_{t_2} = 0,5 & k_{\eta_2} = 1,1 \\ 3: & k_{a_3} = 0,15 & k_{d_3} = 0,15 & k_{s_3} = 0,8 & k_{t_3} = 0,0 & k_{\eta_3} = 1,1 \end{array}$$

Intenzita okolního světla $I_a = 1,0$ a intenzita zdroje světla $I_p = 10$, přičemž koeficient n pro Phongovo stínování je pro všechny plochy roven 50.

Paprsek je orientován od pozorovatele směrem ke scéně a je určen rovnicemi

$$x = 5 \quad y = 0$$

přičemž výsledný strom znázorňující dělení paprsku je na obr. 3.4.6.

Průsečík paprsku s plochou 1 je určen řešením soustavy rovnic, která je dána rovnicemi roviny a paprsku

$$\begin{aligned} x + z - 12,5 &= 0 \\ x &= 5 \quad a \quad y = 0 \end{aligned}$$

a tedy

$$z = 7,5$$

Prvním průsečíkem s plochou je bod $x_1 = [5, 0, 7,5]^T$, který tvoří první uzel stromu. Normovaný normálový vektor \hat{n}_1 plochy 1 je:

$$\hat{n}_1 = \frac{1}{\sqrt{2}} [1, 0, 1]^T$$

2

3

Z polohy pozorovatele lze určit pohledový vektor \mathbf{v}_1 :

$$\mathbf{v}_1 = [0, 0, -1]^T$$

a tedy:

$$\mathbf{v}'_1 = \frac{\mathbf{v}_1}{\mathbf{v}_1^T \cdot \hat{\mathbf{n}}_1} = -\sqrt{2} [0, 0, 1]^T$$

Směrový vektor odraženého paprsku je

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{v}'_1 + 2 \hat{\mathbf{n}}_1 = -\sqrt{2} [0, 0, 1]^T + \frac{2}{\sqrt{2}} [1, 0, 1]^T \\ &= \sqrt{2} [1, 0, 0]^T \end{aligned}$$

přičemž:

$$\hat{\mathbf{n}}_1 + \mathbf{v}'_1 = \frac{1}{\sqrt{2}} [1, 0, -1]^T$$

Pak

$$\begin{aligned} k_{f_1} &= \left(k_\eta^2 \cdot \| \mathbf{v}'_1 \|^2 - \| \mathbf{v}'_1 + \hat{\mathbf{n}}_1 \|^2 \right)^{-1/2} = \left[\left(\frac{1}{1,1} \right)^2 \cdot 2 - 1 \right]^{-1/2} \\ &= 1,238 \end{aligned}$$

$$\begin{aligned} \mathbf{p}_1 &= k_{f_1} \cdot (\hat{\mathbf{n}}_1 + \mathbf{v}'_1) - \hat{\mathbf{n}}_1 \\ &= \frac{1}{\sqrt{2}} \cdot \left[1,238 \left([1, 0, -1]^T \right) - [1, 0, 1]^T \right] \\ &= [0, 168, 0, -1, 582]^T \end{aligned}$$

Odražený paprsek \mathbf{r}_1 opouští scénu a nebude proto dále uvažován.
(V uvedeném příkladě toto vyplývá z obrázku, jinak by bylo nutné hledat průsečík s plochami!)

Nyní je možné zapsat rovnice propuštěného paprsku \mathbf{p}_1 v parametrickém tvaru, tj. :

$$x = 5 + 0,168 \cdot t \quad y = 0 \quad z = 7,5 - 1,582 \cdot t$$

Průsečík \mathbf{x}_2 paprsku \mathbf{p}_1 s druhou rovinou je určen dosazením těchto rovnic do rovnice 2. plochy, tj.

$$\begin{aligned} x - z - 2 &= 5 + 0,168 \cdot t - 7,5 + 1,582 \cdot t - 2 \\ &= 1,75 \cdot t - 4,5 = 0 \end{aligned}$$

a tedy

$$t = 2,571$$

Dosazením hodnoty t do parametrických rovnic pro paprsek p_1 .

$$x_2 = 5 + 0,168 \cdot (2,571) = 5,432$$

$$y_2 = 0$$

$$z_2 = 7,5 - 1,582 \cdot (2,571) = 3,433$$

Vzdáenosť mezi průsečíky x_1 , x_2 je dána:

$$d_{12} = |x_2 - x_1| = \sqrt{(5,432 - 5)^2 + (3,433 - 7,5)^2} = 4,09$$

Paprsek p_1 je nyní vzat jako nový paprsek určený:

- směrovým vektorem $v_2 = p_1$
- bodem x_2 .

Pak

$$x_2 = [5,432, 0, 3,433]^T$$

$$v_2 = p_1 = [0,168, 0, -1,582]^T$$

Normalizovaný vektor \hat{n}_2 druhé plochy (pozor na její orientaci!) je určen:

$$\hat{n}_2 = \frac{1}{\sqrt{2}} [-1, 0, 1]^T$$

Pak

$$r_2 = [-1,278, 0, 0,136]^T$$

$$p_2 = [0,215, 0, -1,999]^T$$

Paprsek p_2 opouští scénu a nebude tedy dále uvažován.

Pro paprsek r_2 lze psát:

$$x(t) = x_2 + r_2 \cdot t$$

tj.

$$x = 5,432 - 1,278 \cdot t \quad y = 0 \quad z = 3,433 + 0,136 \cdot t$$

Dosazením do rovnice pro třetí plochu dostáváme :

$$\begin{aligned} x - 3z + 9 &= 5,432 - 1,278 \cdot t - 3(3,433 + 0,136 \cdot t) + 9 \\ &= -1,686 \cdot t + 4,133 = 0 \end{aligned}$$

Řešením dostáváme

$$t = 2,451$$

Dosazením do parametrické rovnice pro paprsek r_2

$$x_3 = 5,432 - 1,278 \cdot 2,451 = 2,299$$

$$y_3 = 0$$

$$z_3 = 3,433 - 0,136 \cdot 2,451 = 3,766$$

Vzdálenost mezi dvěma průsečíky x_2 , x_3 je určena

$$d_{23} = |x_3 - x_2| = \sqrt{(x_3 - x_2)^2 + (z_3 - z_2)^2} = 3,151$$

Nyní je paprsek r_2 vzat opět jako nový paprsek určený

- bodem $x_3 = [2,999, 0, 0,136]^T$

- vektorem $v_3 = r_2 = [-1,278, 0, 0,136]^T$

Nyní lze opět určit

$$\hat{n}_3 = \frac{1}{\sqrt{10}} [1, 0, -3]^T$$

$$p_3 = [-1,713, 0, 0,483]^T$$

$$r_3 = [-1,765, 0, -1,643]^T$$

Vzhledem k tomu, že oba paprsky opouštějí scénu, je výpočet ukončen. Navíc $k_{t_3} = 0$, a tedy se negeneruje žádný paprsek vzniklý průchodem plochou, neboť je neprůsvitná. Z obrázku vyplývá, že světelný zdroj je plochou zakryt, a tedy 3. plocha se nachází ve stínu. Tudíž bod x_3 obdrží pouze jas okoli, tj.

$$I_3 = k_a \cdot I_a = 0,15 \cdot 1 = 0,15$$

Uvedená intenzita je přenášena podél vektoru $-r_2$ na povrch 2. plochy, přičemž jas je zeslaben vlivem vzdálenosti, viz [143], na hodnotu:

$$I_{s_2} = \frac{I_3}{d_{23}} = \frac{0,15}{3,151} = 0,0476$$

Z polohy průsečíku na druhé ploše vyplývá, že nic nezakrývá světelný zdroj (spojnica bodu se světelným zdrojem neprotíná jinou plochu).

Směrový vektor L_2 paprsku přicházejícího od zdroje světla k ploše 2 je určen

$$L_2 = x_p - x_2 = [x_p - x_2, 0, z_p - z_2]^T$$

kde x_p je poloha zdroje světla I_p

x_2 je průsečík paprsku s druhou plochou

Dosazením

$$L_2 = [-2,432, 0, 6,567]^T$$

a normalizaci dostaváme

$$\hat{L}_2 = [-0,347, 0, 0,938]^T$$

Následně lze určit

$$\begin{aligned} \hat{L}_2^T \cdot \hat{n}_2 &= [-0,347, 0, 0,938]^T \cdot \frac{1}{\sqrt{2}} [-1, 0, 1]^T \\ &= 0,909 \end{aligned}$$

Paprsek dopadající na plochu 2 ve směru L_2 se odrazí ve směru R_2 , přičemž:

$$\hat{R}_2 = [-0,938, 0, 0,347]^T$$

Směrový vektor pozorovatele je

$$\hat{s}_2 = -\hat{p}_1$$

a tedy

$$\begin{aligned} \hat{R}_2^T \cdot \hat{s}_2 &= -[-0,938, 0, 0,347]^T \cdot [0,168, 0, -1,582]^T \\ &= -0,707 \end{aligned}$$

Pak

$$\begin{aligned} I_2 &= k_{a_2} \cdot I_a + k_{d_2} \cdot I_p \cdot (\hat{L}_2^T \cdot \hat{n}_2) + k_{s_2} \cdot I_p \cdot (\hat{R}_2^T \cdot \hat{s}_2)^n + k_{s_2} \cdot I_{s_2} + k_{t_2} \cdot I_{t_2} \\ &= 0,15 \cdot 1 + 0,15 \cdot 10 \cdot 0,909 + 0,8 \cdot 10 \cdot 0 + 0,8 \cdot 0,0476 \\ &\quad + 0,5 \cdot 0 = 1,552 \end{aligned}$$

Tato intenzita je přenášena podél vektoru $-p_1$ na první povrch a je vlivem vzdálenosti zmenšena na hodnotu:

1

2

$$I_{t_1} = \frac{I_2}{d_{12}} = \frac{1,552}{4,09} = 0,379$$

Průsečík x_1 není zastíněn žádnou plochou.

Analogicky L_2 a R_2 lze určit L_1 a R_1 :

$$L_1 = x_p - x_1 = [-2, 0, 2,5]^T$$

a po normalizaci

$$\hat{L}_1 = [-0,625, 0, 0,781]^T$$

Následně

$$\hat{L}_1^T \cdot \hat{n}_1 = [-0,625, 0, 0,781]^T \cdot \frac{1}{\sqrt{2}} [1, 0, 1]^T \\ = 0,110$$

Odražený paprsek R_1 je určen vektorem:

$$\hat{R}_1 = [0,781, 0, -0,625]^T$$

Nyní pohledový vektor $\hat{s}_1 = -v_1$, a tedy

$$\hat{R}_1^T \cdot \hat{s}_1 = -0,625$$

Tudíž

$$I_1 = k_{a_1} \cdot I_a + k_{d_1} \cdot I_p \cdot (\hat{L}_1^T \cdot \hat{n}_1) + k_{s_1} \cdot I_p \cdot (\hat{R}_1^T \cdot \hat{s}_1)^n + k_{s_1} \cdot I_{s_1} + k_{t_1} \cdot I_{t_1} \\ = 0,15 \cdot 1 + 0,15 \cdot 10 \cdot 0,11 + 0,8 \cdot 10 \cdot 0 + 0,8 \cdot 0 \\ + 0,5 \cdot 0,379 = 0,505$$

Toto je intenzita, která je vnímána pozorovatelem. Nízká hodnota

je způsobena tím, že první povrch je osvětlen "téměř" tečně.

Vzhledem k velké hodnotě n nejsou odlesky zobrazovány.

V případě barevného světla či barevných ploch je nutné uvedený výpočet provést pro všechny základní barvy (např. pro RGB). Je zřejmé, že je nutné zadat i iluminační charakteristiky pro příslušné základní barvy.

1

2 Výše uvedený algoritmus je převzat z:

- 3 • Skala,V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia, 1993

4

15.4. CSG stromy a sledování paprsku

V předchozí části jsme se zabývali základním algoritmem sledování paprsku a předpokládali jsme víceméně, že scéna se skládá navzájem z disjunktních objektů. Reálné scény tuto podmínu obecně nesplňují, neboť objekty při množinových operacích, tj. při operacích průniku, sjednocení, rozdílu atd., se obecně protínají. Je tedy nutné, aby výpočet průsečíku paprsku s takto definovanými objekty správně identifikoval ten průsečík, který je opravdu průsečíkem s celkovým objektem. Každý výpočet průsečíku paprsku s uzavřeným objektem obecně generuje množinu intervalů

$$\{\langle t_{i_{min}}, t_{i_{max}} \rangle\}_{i=1}^k \quad (15.2)$$

(pouze v případě konvexního objektu se generuje max. jeden interval). S takto vypočtenými intervaly je pak nutné realizovat odpovídající množinové operace, a to podle definice zobrazovaného objektu.

Vzhledem k tomu, že paprsek je parametrizován tak, že:

$$t \in (0, \infty) \quad (15.3)$$

bude výsledný interval hodnot t ještě omezen. Je zřejmé, že stejný postup se použije nejdříve na ohraňující tělesa a pak následně na vlastní průsečíky odpovídajícím způsobem.

V praxi výpočetní proces nad CSG stromem tedy musí pracovat se seznamem intervalů. Navíc uživatel může definovat scénu tak, že dochází k problémům se stabilitou výpočtu, např. realizací duté koule s velmi slabou stěnou, tj. rozdíl dvou koulí s téměř stejným poloměrem.

V každém případě je vhodné se zabývat transformací scény tak, aby operace průniku a rozdílu byly v CSG stromu co nejnižší a naopak operace sjednocení co nejvíce, viz kap.7.7 (CSG stromy).

20

21

16. Radiační metoda

2 Radiační metody jsou založeny na energetické bilanci ve scéně, kdy se scéna uzavře do energetického
 3 obalu a spočítají následně energetický příjem a výdaj jednotlivých plošek reprezentující povrch
 4 objektů ve scéně. Z principu je tedy zřejmé, že radiační metody nejsou schopny jednoduše
 5 respektovat např. lom světla na rozhraní dvou opticky různých materiálů.

6 Celý proces vede na řešení soustav lineárních rovnic velkého rozsahu. Existuje celá řada různých
 7 modifikací základní metody.

8

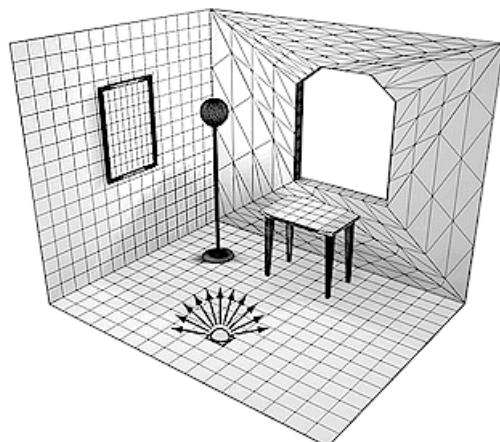
9 Porovnání základních přístupů:

- 10 • Willmott,A., Heckbert,P.S.: An Empirical Comparison of Radiosity Algorithms, Tech.Rep.
 11 Carnegie Mellon Univ., 1997 <http://www.cs.cmu.edu/~radiosity/emprad-tr.html> ([CLICK off-line](#))

16.1. Princip radiační metody

14 Princip radiační metody vychází z principu energetické rovnováhy, kdy daná scéna je uzavřena do
 15 „energetického“ obalu a scéna obsahuje dva typy plošek, a to: plošky difúzně absorbuje světlo, tj.
 16 s difuzním odrazem, a plošky difúzně produkují světlo. Pro jednoduchost uvažme jednoduchou
 17 scénu, viz Obr.16.1, kdy jednotlivé plochy jsou reprezentovány malými ploškami. Výpočet je založen
 18 na energetické bilanci.

19



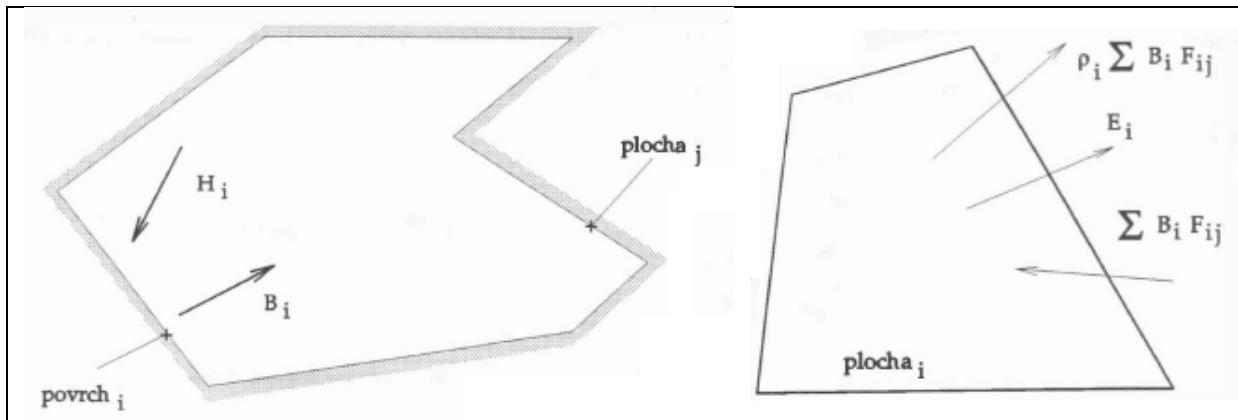
20
21 Obr.16.1: Ukázka jednoduché scény

22
23 V následujícím budeme používat značení:

- 24 • i – index i -té plochy
- 25 • ρ_i – odrazivost i -té plochy určující, jaká část přijaté energie bude emitována zpět, tj.
26 odražena
- 27 • B_i – radiace (vyzařování) i -té plochy
- 28 • E_i – vlastní emise (vyzařování) i -té plochy
- 29 • H_i – radiace přijatá i -tou plochou od ostatních ploch
- 30 • A_i – je velikost i -té plochy

31 Výpočet je založen na energetické bilanci, takže radiace musí být násobena plochou dané plošky. Pro
 32 jednoduchost použijeme diskrétní popis místo integrálního popisu.

1



2 Obr.16.2: Energetické poměry na ploše

3
4 Celková radiace (energie na jednotku plochy [W/m^2]) i -té plochy je určena vztahem:

$$5 \quad B_i = E_i + \rho_i H_i \quad (16.1)$$

6 Pro energetickou bilanci pak můžeme psát:

$$7 \quad H_i A_i = \sum_{j=1, i \neq j}^N B_j A_j F_{ji} \quad i = 1, \dots, N \quad (16.2)$$

8 kde: F_{ji} je konfigurační faktor (form factor), který reprezentuje vliv vzájemné polohy ploch i a j a také
9 případné jejich zakrytí.

10 Úpravou obdržíme:

$$11 \quad H_i = \frac{1}{A_i} \sum_{j=1, i \neq j}^N B_j A_j F_{ji} \quad i = 1, \dots, N \quad (16.3)$$

12 Je zřejmé, že musí platit vztah:

$$13 \quad A_j F_{ji} = A_i F_{ij} \quad i, j = 1, \dots, N \quad (16.4)$$

14 Pak lze psát:

$$15 \quad H_i = \frac{1}{A_i} \sum_{j=1, i \neq j}^N B_j A_j F_{ji} = \sum_{j=1, i \neq j}^N B_j F_{ij} \quad i = 1, \dots, N \quad (16.5)$$

16 Emitovaná radiace i -té plochy je pak dán:

$$17 \quad B_i = E_i + \rho_i \sum_{j=1, i \neq j}^N B_j F_{ij} \quad i = 1, \dots, N \quad (16.6)$$

18 a tedy:

$$19 \quad B_i - \rho_i \sum_{j=1, i \neq j}^N B_j F_{ij} = E_i \quad i = 1, \dots, N \quad (16.7)$$

1 Uvedené vztahy v maticové formě pak mají tvar:

$$\begin{bmatrix} 1 - \rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1 - \rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1 - \rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix} \quad (16.8)$$

2 přičemž vektor neznámých je $\mathbf{B} = [B_1, \dots, B_N]^T$.

3

4 Je nutné si uvědomit, že uvedená soustava rovnic je parametrická, kde E_i , F_{ij} a ρ_i jsou vlastně
5 parametry, které musejí být určeny pro každou plošku. Hodnoty E_i jsou nenulové pro plošky
6 přispívající k radiaci, tj. tvořící zdroj světla. Navíc hodnoty E_i a ρ_i jsou funkčemi vlnové délky λ .

7

8 K řešení soustavy rovnic lze použít s výhodou iterační metody, neboť matice je diagonálně
9 dominantní a pozitivně definitní.

10

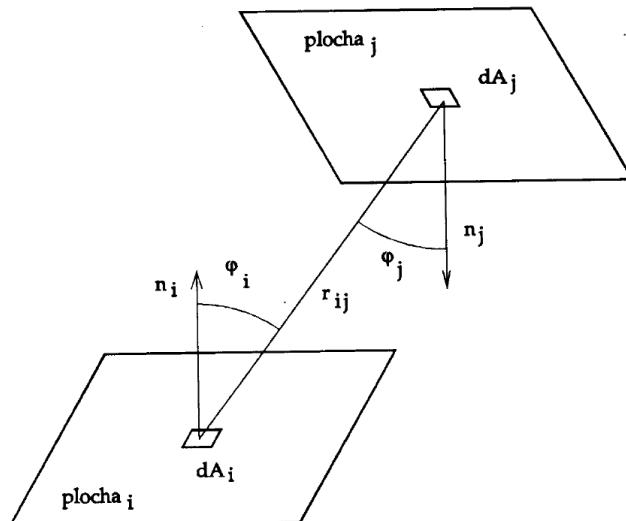
11 Konfigurační faktory F_{ij} jsou pro danou scénu konstantní. Pokud se tedy scéna nemění, pak se
12 výpočet provede pouze jednou a výsledný obraz, který závisí na pozici pozorovatele, je pak dán jen
13 vlastně zobrazením plošek s respektováním viditelnosti, neboť jsou k dispozici všechny hodnoty
14 potřebné pro stínování zobrazované plochy.

15

16 Nyní zbývá určit konfigurační faktory F_{ij} .

17 16.2. Výpočet konfiguračních faktorů

18 Hodnota konfiguračního faktoru (form factor) je dána vzájemnou polohou plošek a také jejich
19 vzájemnou viditelností, viz Obr.16.3.



Obr.16.3: Konfigurační faktory

20

21

22

23 Pro plochy ničím nezakryté lze v diferenciálním tvaru psát:

$$F_{dA_i dA_j} = \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} \quad (16.9)$$

24 kde: r_{ij} je vzdálenost elementárních plošek dA_i a dA_j , přičemž $r_{ij} = r_{ji}$.

- 1 Celkový energetický příspěvek elementární plošky dA_i k energii celé plochy A_j dostáváme integrací
2 výrazu přes j -tou plochu:

$$F_{A_i dA_j} = \int_{A_j} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} dA_j \quad (16.10)$$

- 3 Celková energie získaná plochou j -té plochy od i -té plochy je dána

$$F_{A_i A_j} = \int_{A_i} \int_{A_j} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} dA_j dA_i \quad (16.11)$$

- 4 Průměrný radiační příspěvek i -té plochy od j -té plochy lze určit jako:

$$F_{ij} = \frac{1}{A_i} F_{A_i A_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} dA_j dA_i \quad (16.12)$$

5

- 6 Ze symetrie pak vyplývá

$$F_{ji} = \frac{1}{A_j} F_{A_j A_i} = \frac{1}{A_j} \int_{A_j} \int_{A_i} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} dA_j dA_i \quad (16.13)$$

7

8

- 9 Ze zákona o zachování energie dostáváme podmínky:

$$\sum_{j=1}^N F_{ij} = 1 \quad i = 1, \dots, N \quad (16.14)$$

- 10 přičemž $F_{ii} = 0$ neboť plocha nemůže „vidět“ sama sebe.

11

- 12 V reálné situaci jsou plochy zcela nebo částečně zakryty jinými plochami. Pak konfigurační faktor je
13 dán:

$$F_{ij} = \frac{1}{A_i} F_{A_i A_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} [\mathbf{HID}_{ij}] dA_j dA_i \quad (16.15)$$

- 14 kde: \mathbf{HID}_{ij} je funkce, která nabývá hodnot 0 nebo 1 podle toho, zda elementární ploška dA_j je
15 viditelná z elementární plošky dA_i .

16

- 17 Z uvedeného je zřejmé, že analytický výpočet konfiguračních faktorů F_{ij} je téměř nemožný.

- 18 V případě, že vzdálenost je ve srovnání s velikostmi ploch velká, lze psát:

$$F_{ij} \approx \frac{1}{A_i} \int_{A_i} K dA_i \quad (16.16)$$

- 19 kde:

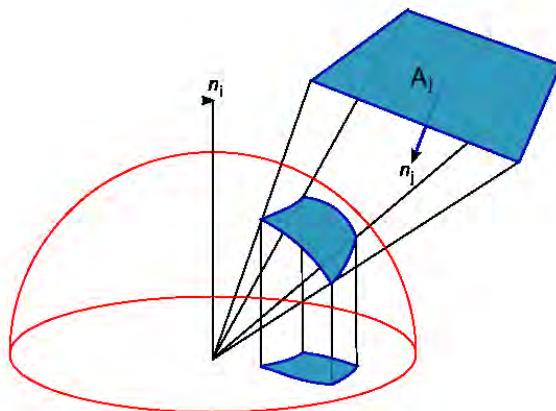
$$K \approx \int_{A_j} \frac{\cos \varphi_i \cos \varphi_j}{\pi r_{ij}^2} [\mathbf{HID}_{ij}] dA_j \quad (16.17)$$

- 20 neboť hodnota K se prakticky nemění. Nusselt ukázal, že uvedený postup je ekvivalentní promítnutí
21 plochy A_j na jednotkovou polokouli (hemisphere) s následnou projekcí na jednotkový kruh základny a
22 poměr promítnuté plochy a plochy kruhu pak určuje hodnotu konfiguračního faktoru, viz Obr.16.4.
23 Nicméně analytická projekce na polokouli je stále příliš složitá. Chen a Greenberg navrhli řešení
24 mapováním plochy na horní část krychle, viz Obr.16.5.

25

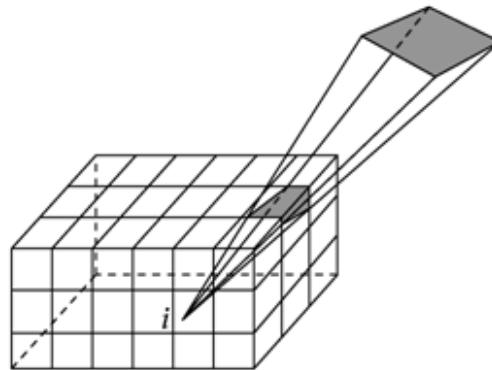
26

27



Obr.16.4: Mapování plochy na polokouli

Idea je celkem jednoduchá, neboť plocha se promítne na horní část polokrychle (hemicube), jejíž povrch se rozdělí na $m \times m$ plošek, kde $m \in \{50, \dots, 10^3\}$. Daná plocha A_j se promítne na polokrychli, resp. promítnutím těch částí, které nejsou zastíněny jinou plochou.



Obr.16.5: Výpočet konfiguračního faktoru pomocí polokrychle

Je zřejmé, že je nutné udělat korekce vůči promítání na polokouli. Jsou dvě zásadní situace, a to promítnutí na horní stěnu a promítnutí na boční stěnu. Každá ploška polokrychle se podílí *delta faktorem* na celkovém konfiguračním faktoru:

$$\Delta F_j = \frac{\cos \theta_i \cos \theta_j}{\pi r^2} \Delta A \quad (16.18)$$

kde ΔA je plocha buňky, r je vzdálenost plochy A_j a plošky dA_i .

Předpokládejme, že polokrychle má svůj vlastní souřadný systém (x, y, z) s počátkem ve středu. Pak pro horní stěnu, viz Obr.16.6.a, polokrychle lze psát:

$$r = \sqrt{x_p^2 + y_p^2 + 1} \quad \cos \theta_i = \cos \theta_j = \frac{1}{r} \quad (16.19)$$

Pak delta faktor je určen:

$$\Delta F_j = \frac{\cos \theta_i \cos \theta_j}{\pi(x_p^2 + y_p^2 + 1)^2} \Delta A \quad (16.20)$$

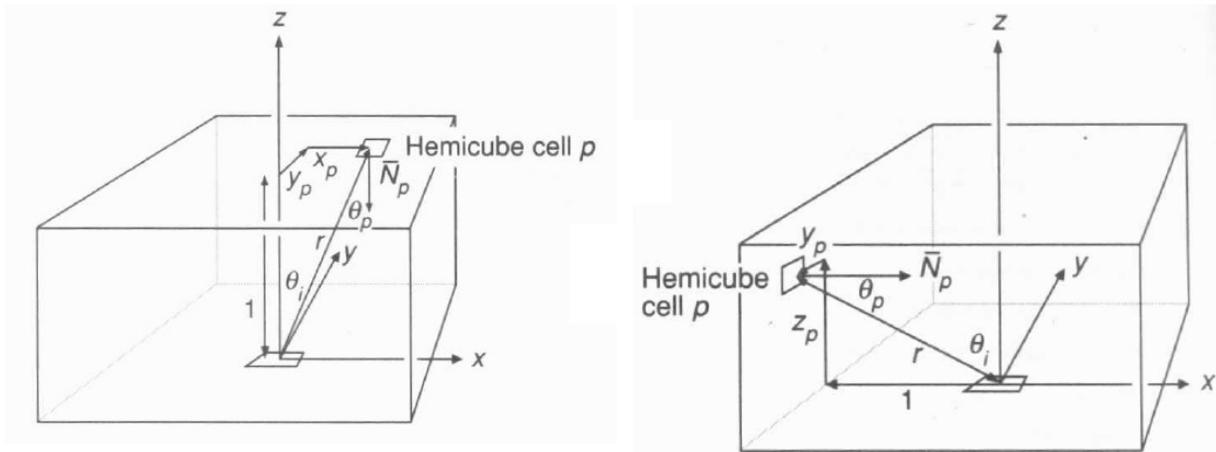
Pro boční stěnu, viz Obr.16.6.b, lze analogicky psát:

$$r = \sqrt{x_p^2 + z_p^2 + 1} \quad \cos \theta_i = \frac{z_p^2}{r} \quad \cos \theta_j = \frac{1}{r} \quad (16.21)$$

1 Pak delta faktor je určen:

$$\Delta F_j = \frac{\cos \theta_i \cos \theta_j}{\pi(x_p^2 + z_p^2 + 1)^2} \Delta A \quad (16.22)$$

2 Konfigurační faktor je pak dán součtem delta faktorů. Díky symetrii pak lze některé výpočty ušetřit.



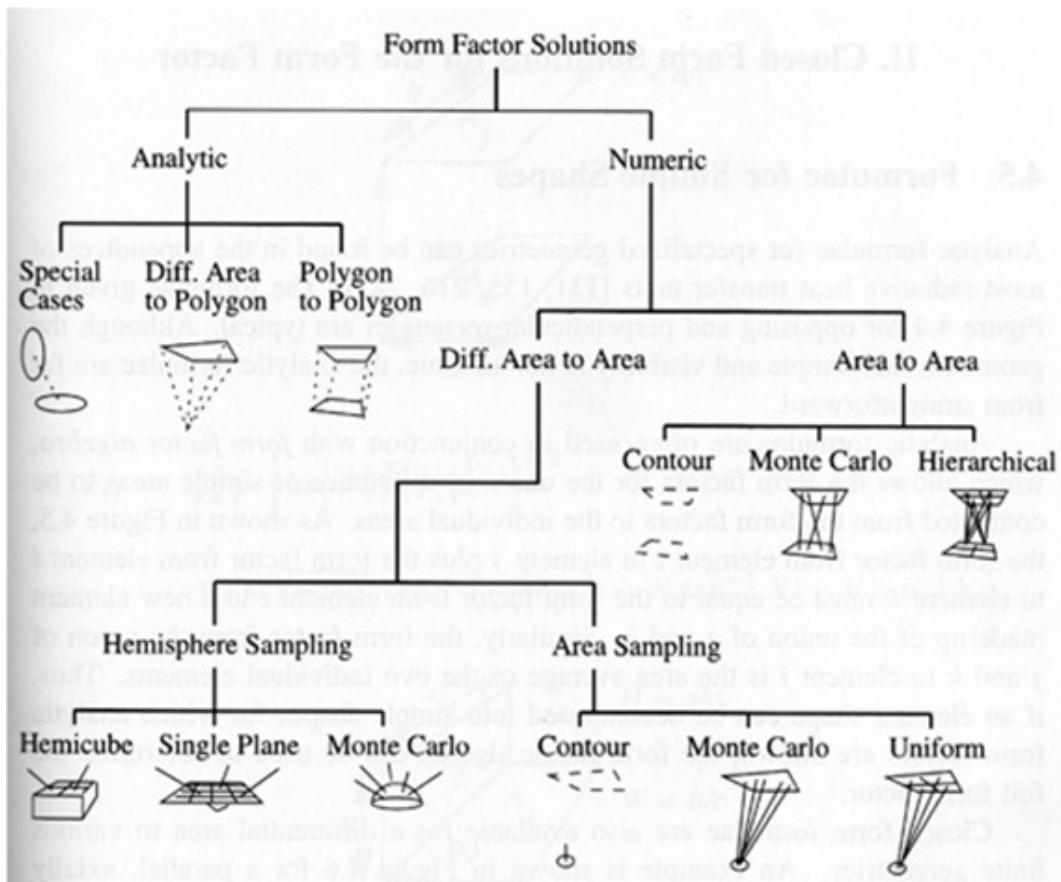
a) Delta faktor pro horní stěnu

b) Delta faktor pro boční stěnu

Obr.16.6: Určení delta faktoru pro stěny polokrychle

3

4 Pro výpočet konfiguračních faktorů se používá celá řada metod, viz Obr.16.7.



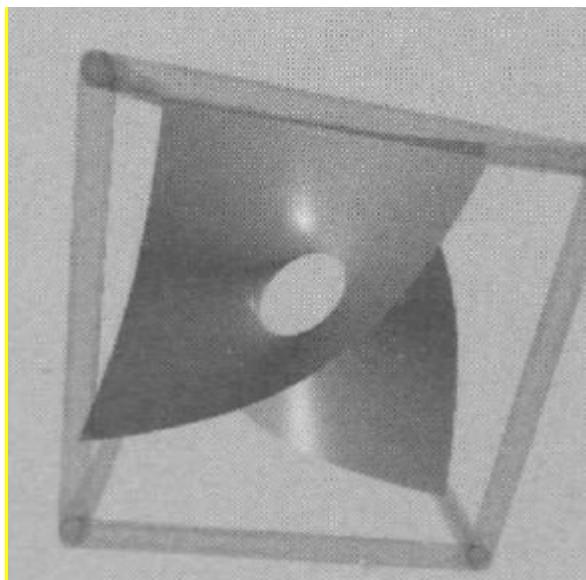
Obr.16.7: Základní metody výpočtu konfiguračních faktorů

7 Podrobněji viz např.:

- Foley,J.D., van Dam,A., Feiner,S.K., Hughes,J.F.: Computer Graphics: Principles and Practice, Addison-Wesley, 1995

Upozornění

Zde je vhodné upozornit, že lokální metody obecně mohou poskytovat vlivem interpolací falešné výsledky. Na Obr.16.8 je ukázána plocha vzniklá trilineární interpolací a zobrazená pomocí sledování paprsku. Plocha má uprostřed díru, která se při použití trojúhelníků obvykle nezobrazí.



Obr.16.8: Trilineární interpolace a skutečný povrch

- Shirley,P.: Fundamentals of Computer Graphics, A.K.Peters, 2005

17. Mapovací techniky – texturování – vložit text

2 V předchozím textu byly předloženy základní techniky zobrazování geometrických objektů.
 3 V převážné většině případů se v praxi pracuje s planárními objekty, většinou s trojúhelníky, které
 4 umíme vystínovat, resp. vybarvit, tak, že výsledný vjem je blízký realitě. Nicméně v mnoha případech
 5 potřebujeme daný povrch dále „upravit“, např. na povrchu má být nějaký vzor nebo má být
 6 zvrásněný apod. Techniky, které toto umožňují, se nazývají mapovací techniky.

7
 8 Mapovací techniky lze rozdělit na následující základní typy:

- 9 • texturovací techniky (texture mapping) – na daný povrch se „natáhne“ vzor, resp. textura
- 10 • techniky zvrásnění (bump mapping) – povrch objektu se zvrásní hrbolek a vytváří se dojem
 11 „hrbolatého“ povrchu
- 12 • mapování okolí (environmental mapping) – tato technika umožňuje na povrchu objektů
 13 zobrazit odraz okolního prostředí. Tato technika je vlastně založena na zrcadlení, tj. na
 14 povrchu objektu vidíme odraz okolního světa

15 Někdy se do mapovacích technik ještě zahrnují techniky generování stínů, které pomáhají k lepšímu
 16 prostorovému vjemu.

17.1. Texturovací techniky

18 Texturovací techniky jsou technikami, kdy na roviný nebo hladký povrch „nanášíme“ nějaký vzor,
 19 který se nazývá textura. Textura může být určena vzorem, kterým se daný povrch, např. n-úhelník,
 20 vyplní, nebo obrazovou předlohou nebo může být generována procedurálně. Texturováním se
 21 vlastně vyvolává vjem detailního zpracování povrchu geometrického objektu.

22
 23 Textury mohou být jedno, dvou, tří nebo čtyř rozměrné. Jednorozměrná textura umožňuje na křivku
 24 namapovat daný vzor, třírozměrná textura může popisovat „blok“ materiálu, z kterého daný objekt
 25 můžeme vytiskat atd. Nicméně protože počítačová grafika se zabývá především zobrazováním
 26 povrchů, budeme se v dalším textu zabývat dvourozměrnými texturami.

27
 28 Existuje mnoho metod, jak lze texturování realizovat, nicméně všechny techniky obsahují mapování
 29 mezi třemi nebo čtyřmi souřadnými systémy. Metody texturování jsou obecně dány typem povrchu,
 30 který zobrazujeme, a architekturou grafického systému.

31 Předpokládejme, že textura je dána obrazem. Každý element textury $T(s, t)$ se nazývá *texel* (Texture
 32 Element), obdobně jako u obrazu se obrazový element nazývá *pixel* (Picture Element). Proměnné s, t
 33 se nazývají texturové souřadnice (Texture Coordinates) a jsou obvykle mapovány na interval
 34 $s, t \in \langle 0,1 \rangle$.

35 Texturovací mapa vlastně jednoznačně spojuje každý bod T s daným bodem geometrického objektu,
 36 který je sám mapován do obrazových souřadnic po projekci na průmětně. Pokud je tedy objekt
 37 reprezentován prostorovými souřadnicemi (x, y, z) , resp. (x, y, z, w) , pak tyto mapovací funkce jsou:

$x = x(s, t)$	$y = y(s, t)$
$z = z(s, t)$	$w = w(s, t)$

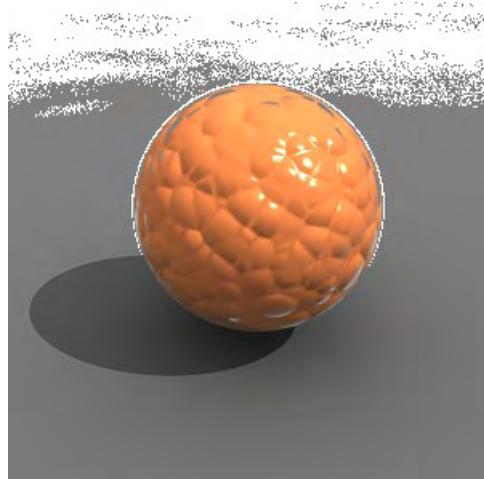
38 I když uvedené mapování existuje, nemusí být vždy jednoduché nebo možné toto mapování určit.
 39 Navíc potřebujeme vlastně inverzní mapování, tj. pro daný bod o souřadnicích (x, y, z) , resp.
 40 (x, y, z, w) najít odpovídající texturovací souřadnice s, t , tj. potřebujeme inverzní funkce:

$s = s(x, y, z, w)$	$t = t(x, y, z, w)$
---------------------	---------------------

1
2
3

4 **17.2. Techniky zvrásnění**

5
6



7
8 Obr.17.1:
9

10 **17.3. Mapování okolí**

11
12

13 **17.4. Procedurální texturování**

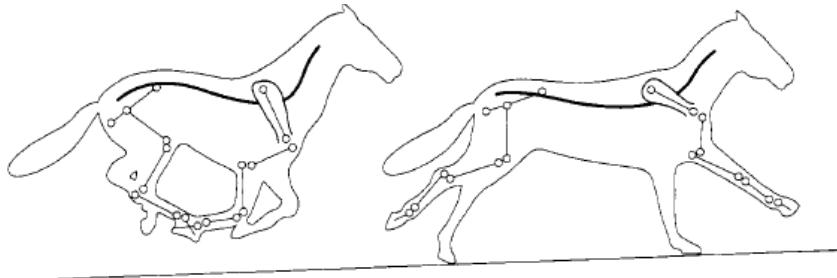
14 dodělat
15
16
17

1 **18. Animace, principy a inverzní kinematika – vložit text**

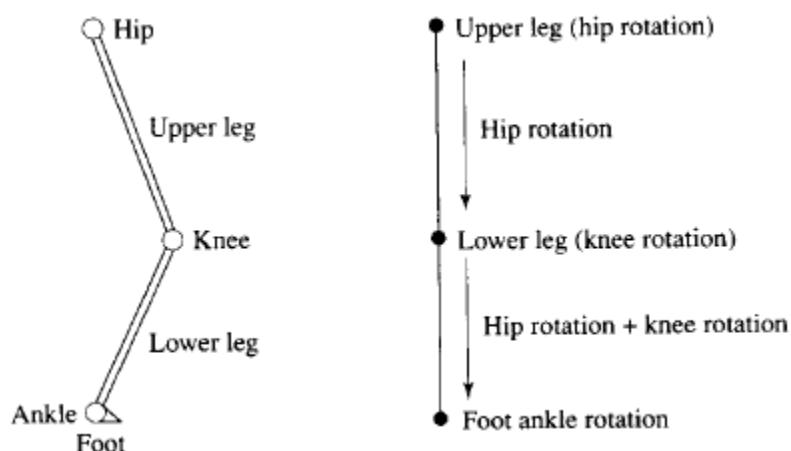
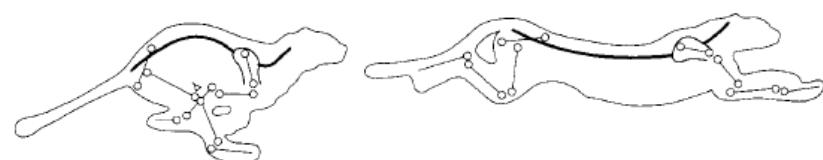
2 Animační techniky – viz reference

3

4 Samostatné studium



5
6



7
8

9 **18.1. Kinematika, Inverzní kinematika – vložit text**

10 **18.2. Dynamika – vložit text**

11 **18.3. Particles – vložit text**

12 **18.4. Ryv a jeho dusledky pro animaci a taktilní I/O – vložit text**

13

14 **Reference:**

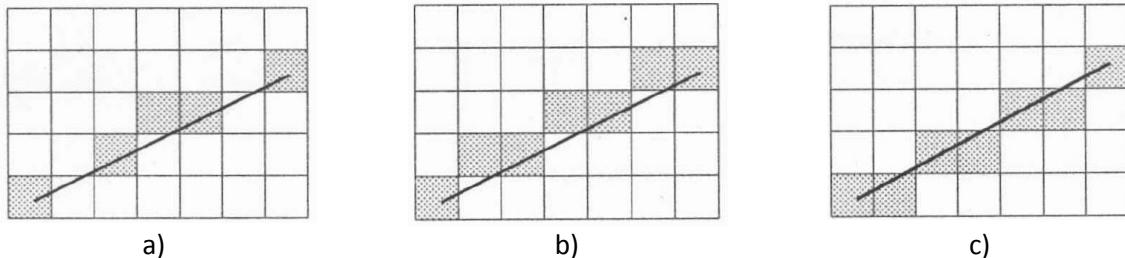
- Watt,A., Polycarpo,F.: 3D Games: Animation and Advanced Real-time Rendering, Addison Wesley, 2003
- Watt,A.: 3D Computer Graphics, Pearson Addison Wesley, 2000
- Frank,M.: Spider project,

19. Rastrová grafika – základní algoritmy

2 V předcházejícím textu byly vysvětleny základní metody a datové struktury používané v počítačové
 3 grafice včetně příslušného matematického aparátu. Vlastní zobrazení je však realizováno většinou
 4 v diskrétním prostředí na konkrétním zařízení na tzv. rastru, který je většinou pravoúhlý případně
 5 hexagonální. Nyní ukážeme základní algoritmy pro kreslení úsečky a kružnice v pravoúhlém rastru.

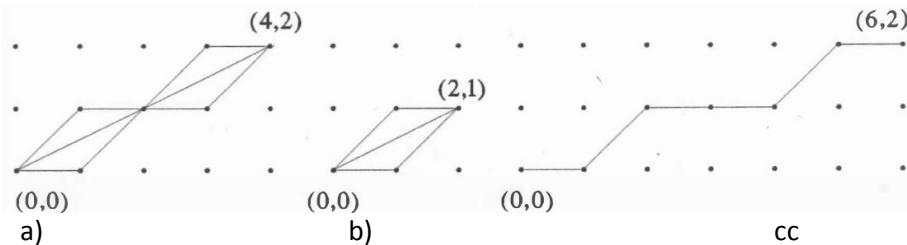
19.1. Bresenhamův algoritmus – úsečka

7 Nakreslení úsečky se zdá být jednoduchým problémem, nicméně v diskrétním prostředí je úloha
 8 trochu komplikovanější, viz Obr.19.1. Navíc, vzhledem k velkému počtu generovaných úseček, musí
 9 být algoritmus velmi jednoduchý.



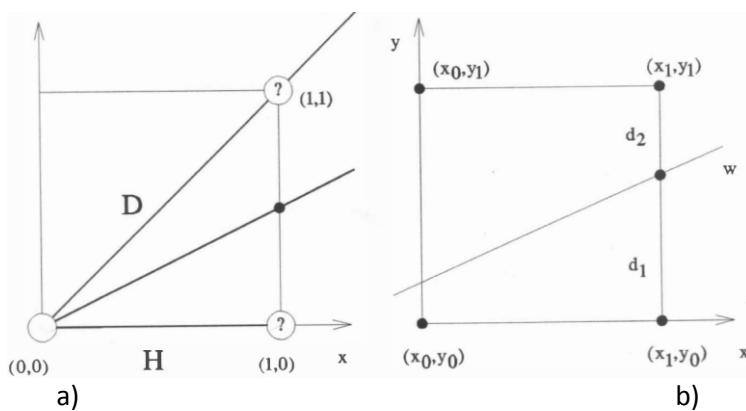
Obr.19.1: Kreslení úsečky v rastrovém prostředí

10 Představíme-li si jednoduchou situaci na Obr.19.2, kdy se kreslí úsečka z bodu (0,0) do bodu (4,2),
 11 pak je zřejmé, že v diskrétním prostředí nelze nakreslit celou úsečku jako část přímky, neboť kreslená
 12 čára se skládá z jednotlivých segmentů s diferencemi ($\pm 1, \pm 1$). Navíc je několik možností, jak lomenou
 13 čáru z bodu (0,0) do bodu (4,2) realizovat.



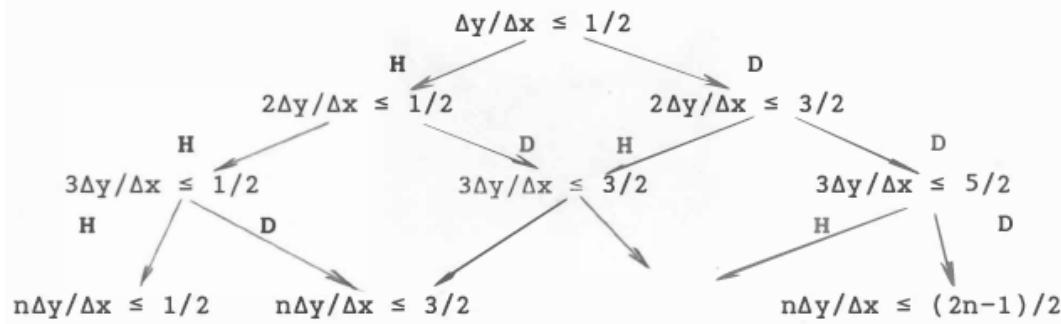
Obr.19.2: Úsečka v rastrovém prostředí

14 Na Obr.19.2.b je situace, kdy nelze algoritmicky rozhodnout, kterou lomenou čáru generovat, neboť
 15 obě možnosti jsou rovnocenné. V případě úsečky z bodu (0,0) do bodu (6,2) je pak generovaná
 16 sekvence znázorněna na Obr.19.2.c. Omezme se na první oktant v následujících úvahách a tedy
 17 generovaný krok bude buď ve směru horizontálním, který označíme H , nebo diagonálním, který
 18 označíme D , Obr.19.3.a.



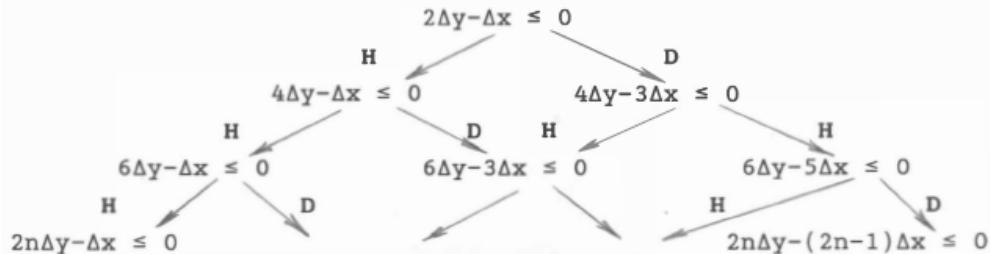
Obr.19.3: Princip rozhodování Bresenhamova algoritmu

1 Pokud úsečka bude procházet pro $x = 1$ pod $1/2$ pak se bude generovat horizontální krok H ,
 2 v opačném případě pak krok diagonální D . V dalších krocích už počáteční bod nebude v pozici $(0,0)$,
 3 Obr.19.3.b, nicméně rozhodovací kritérium se nemění. Lze pak ukázat, že generování lomené čáry
 4 reprezentující požadovanou úsečku se bude řídit výsledky podmínek, viz Obr.19.4.



Obr.19.4: Rozhodovací postup Bresenhamova algoritmu

8 Je zřejmé, že jde o poměrně jednoduchý algoritmus, nicméně zdánlivě vyžadující výpočet v pohyblivé
 9 řádové čárce. Vzhledem k tomu, že pro 1. oktant platí $Δx > 0$ lze všechny podmínky vynásobit
 10 hodnotou $Δx$ a dostaváme modifikované rozhodovací podmínky pro generování jednotlivých kroků,
 11 viz Obr.19.5.



Obr.19.5: Modifikovaný rozhodovací postup Bresenhamova algoritmu

15 Tento algoritmus formuloval Jack Bresenham v roce 1965. Z popisu je zřejmé, že není zapotřebí
 16 operací v pohyblivé řádové čárce a ani operace násobení v celočíselné aritmetice, neboť vše je možné
 17 realizovat posuvem a sčítáním, resp. odečítáním.

19 Podrobněji viz:

- Besenham,J.E.: Algorithm for computer control of a digital plotter. IBM Systems Journal, Vol.4, No.1, pp.25–30, 1965

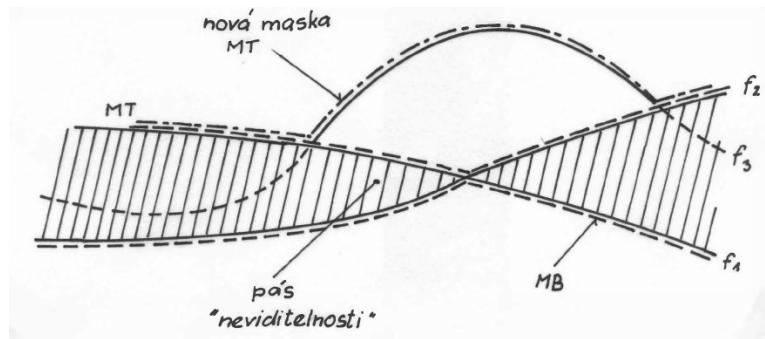
22 Následující algoritmus schematicky popisuje algoritmus generování úsečky pro 1. oktant, rozšíření na
 23 všechny oktanty je jednoduché.

```
procedure BRESENHAM ( x1, y1, x2, y2: integer );
var i, x, y, a, b, d: integer;
begin x := x1;
y := y1;
dx := x2 - x1;
dy := y2 - y1;
d := 2 * dy - dx;
a := 2 * dy;
b := 2 * ( dy - dx );
for i := 0 to dx do
  if d <= 0 then
    begin d := d + a; x := x + 1;
    KROK( "H" )
    end
  else
    begin d := d + b; x := x + 1; y := y + 1;
    KROK ( "D" )
    end
end;
```

Obr.19.6: Bresenhamův algoritmus generování úsečky

19.2. Bresenhamův algoritmus a algoritmus plovoucího horizontu

Řešení neviditelnosti algoritmem plovoucího horizontu uvedené v kap. 12.2 (Řešení viditelnosti pro plochy $2\frac{1}{2}$ D – $z=f(x,y)$) lze efektivně implementovat uvnitř Bresenhamova algoritmu pro kreslení úsečky. Algoritmus plovoucího horizontu je založen na tom, že se kreslí postupně řezy od pozorovatele směrem dozadu. Představme si, že povrch je dán jako funkce $z = f(x, y)$. Označme nejbližší řez f_1 a další vzdálenější řez f_2 . Pak tyto křivky vlastně definují plochu, kterou pozorovatel vidí z podhledu, nebo nadhledu, označme jako *pás neviditelnosti*. Při kreslení dalšího vzdálenějšího řezu f_3 , se pak kreslí jen ty části této křivky, které jsou nad nebo pod pásem neviditelnosti. Předpokládejme, že horní horizont je reprezentován funkcí M_T (top) a dolní horizont pak funkcí M_B (bottom), viz Obr.19.7.



Obr.19.7: Algoritmus plovoucího horizontu

Pak jednotlivé řezy jsou dány křivkami:

$$\begin{aligned} z &= f(x, y_i) & z &= f(x_j, y) \\ i &= 1, \dots, n, x \in \langle a_x, b_x \rangle & j &= 1, \dots, m, z \in \langle a_y, b_y \rangle \\ a_y = y_1 < y_2 < \dots < y_{n-1} < y_n = b_y & a_x = x_1 < x_2 < \dots < x_{m-1} < x_m = b_x \end{aligned} \quad (19.1)$$

Algoritmus plovoucího horizontu lze pak formulovat pro jednotlivé řezy pro osu z takto:

```

MT := -∞; MB := ∞;
for k:=1 to počet řezů do
begin Kresli funkci  $f(x, z_k)$  s respektováním
    viditelnosti  $\forall x \in \langle a_x, b_x \rangle$ ;
    MT := max { MT ,  $f(x, z_k)$  }  $\forall x \in \langle a_x, b_x \rangle$ 
    MB := min { MB ,  $f(x, z_k)$  }  $\forall x \in \langle a_x, b_x \rangle$ 
    { max, min jsou funkce, jejichž argumenty jsou }
    { funkce a výsledkem je opět funkce }
end

```

Vzhledem k tomu, že křivka $f(x, y_k)$ bude kreslena jako posloupnost lineárních úseků vzhledem k proměnné x , lze s výhodou použít Bresenhamův algoritmus pro kreslení úsečky. Viditelnost, resp. neviditelnost je pak realizována modifikovanou instrukcí pro kreslení elementárního kroku

```

1  DRAW_STEP
2      procedure DRAW_STEP;
3          { if yp  $\geq$  MT[xp] then
4              { MT[xp] := yp; PLOT (xp,yp) }
5          if yp  $\leq$  MB[xp] then
6              { MB[xp] := yp; PLOT (xp,yp) }
7      }

```

Alg.XXX

Pro reprezentaci horního horizontu M_T (top) se používá pole MT a pro dolní horizont M_B pak pole MB. Poloha (xp, yp) je polohou kresleného bodu už přímo ve fyzických souřadnicích rastru. Reálná implementace však ještě musí respektovat některé specifické rysy diskretizace. Podrobněji viz:

- Skala,V: Algoritmy počítačové grafiky II, pp.119-125, 2011 ([CLICK off-line](#))

14

15 Dodatek

Vzhledem k tomu, že v diskrétním prostředí může docházet ke kreslení svislých elementárních kroků, je nutné aktualizovat horizont až po nakreslení celého řezu. Proto je nutné v paměti udržovat aktuální horizonty, masky M_B a M_T , a nově vytvářené horizonty M_{B1} a M_{T1} . Po nakreslení celého řezu se pak masky M_B a M_T aktualizují z masek M_{B1} a M_{T1} , tj. $M_B := M_{B1}$ a $M_T := M_{T1}$.

20

```

procedure INIT MASK;
var k: integer;
begin for k:=0 to rstx do
    begin MT1[k]:= -maxint; MB1[k]:= maxint end
end { INIT MASK };
procedure SET MASK;
var k: integer;
begin MT:=MT1; MB:=MB1
end { SET MASK };

procedure DRAW TO ( x,y: integer );
var u,v,ix,iy: integer;
procedure DRAW STEP;
begin if ( yp < MB[xp] ) or ( yp > MT[xp] ) then
    begin plot (xp,yp,1);
        if yp > MT1[xp] then MT1[xp]:=yp;
        if yp < MB1[xp] then MB1[xp]:=yp
        { nastavení masek }
    end
end { DRAW STEP };

```

21

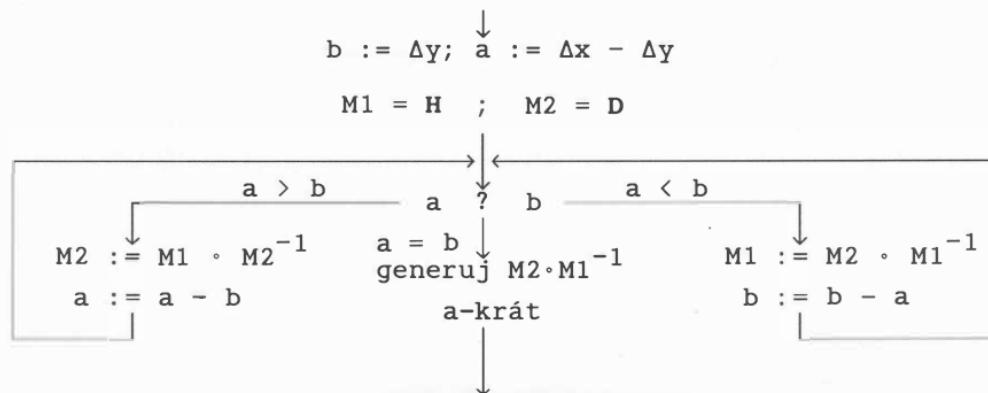
22

23 Po nakreslení celého řezu se musí vyvolat SET MASK aktualizující horní a dolní horizont.

24

19.3. Castle-Pitteway algoritmus generování úsečky

Zajímavý algoritmus Castle-Pitteway vychází z jednoduchého požadavku a to, aby generovaná sekvence z obou stran byla symetrická. To znamená, že generovaná sekvence L , skládající se z kroků H a D , by měla být $L = L_1 \circ L_1^{-1}$, kde \circ značí operaci zřetězení, L^{-1} značí inverzi posloupnosti a L_1 je posloupnost kroků generovaných pro 1/2 úsečky. Tato myšlenka je „lákavá“, ale není platná v diskrétním prostředí. Castle a Pitteway ukázali, že problém je ekvivalentní problému nejmenšího společného dělitele a navrhli následující algoritmus:



Obr.19.8: Castle-Pitteway algoritmus pro generování úsečky

11 Uveďme si na příkladu, postup generování pro úsečku z bodu $(0,0)$ do bodu $(51,11)$.

a	b	M1	M2
40	11	H	HD
29	11	H	HDH
18	11	H	HHDH
7	11	HHDHH	HHDH
7	4	HHDH	HHDHHDHH
3	4	HHDHHHDHHHHDH	HHDHHHDHH
3	1	HHDHHHDHHHHDH	HHDHHHDHHHHDHDDHHHDHH
2	1	HHDHHHDHHHHDH	HHDHHHDHHHHDHDDHHHDHH
1	1	generuje výslednou posloupnost kroků	HHDHHHDHHHHDHDDHHHDHHHDHHHDHHHDHHHDHHHDHHHDHHHDHHHDHHHDHH

Obr.19.9: Castle-Pitteway algoritmus – výsledek generace

15 Celou generovanou posloupnost lze zkráceně zapsat:

$$H^2DH^3DH^4DH^4DH^3DH^4DH^4DH^3DH^4DH^4DH^3DH^2$$

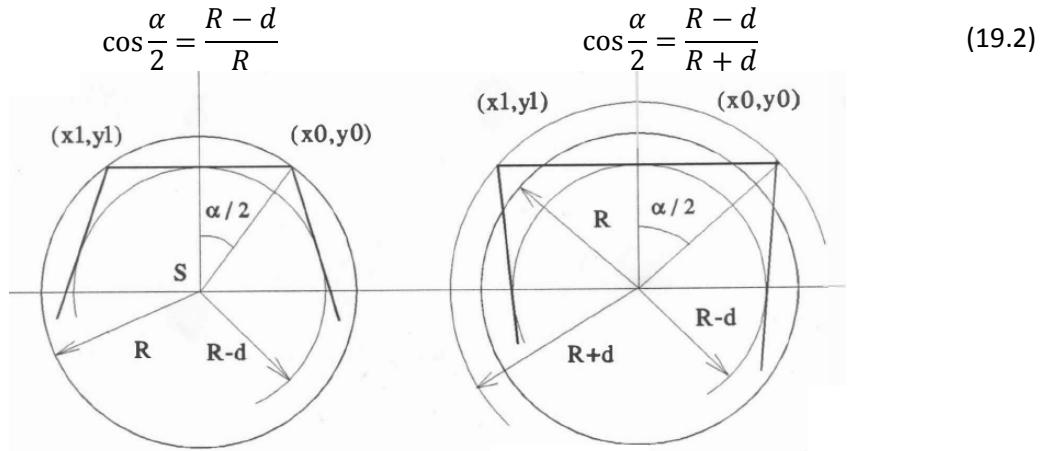
16 Algoritmus je tedy složitosti $O(lgn)$, kde n je počet kroků na ose x . Nicméně je vhodné poznamenat,
17 že operace zřetězení a inverze řetězce je poměrně časově náročná. Inverze řetězce se dá eliminovat
18 samostatnou datovou reprezentací pro inverzní řetězec.

19 Podrobnější informace viz:

- Castle,C.M.A., Pitteway,M.L.V.: An Application of Euclid's Algorithms to Drawing Straight Lines, in Fundamental Algorithms for Computer Graphics (Ed.Earnshaw,R.A.) NATO Advanced Study Institute, Series F, Vol.17, Springer Verlag, pp.135-139, 1985

19.4. Bresenhamův algoritmus pro kružnici

Generování kružnice je trochu komplikovanější, neboť generovaný konvexní n-úhelník může být kružnicí opsán, nebo vepsán. Toto je důležitý faktor např. u numericky řízených obráběcích strojů, kdy je nutné rozlišit, zda jde o „díru“ nebo „hřídel“. Pro podobné aplikace je nutné dodržet odpovídající hodnoty úhlů α pro danou toleranci d .



Obr.19.10: Generování opsané a vepsané kružnice

Obvykle se generuje kružnice tak, že body n-úhelníka jsou na kružnici s poloměrem R . Nejjednodušším způsobem je aplikace polárních souřadnic, a to:

$$x = R \cos \varphi \quad y = R \sin \varphi \quad (19.3)$$

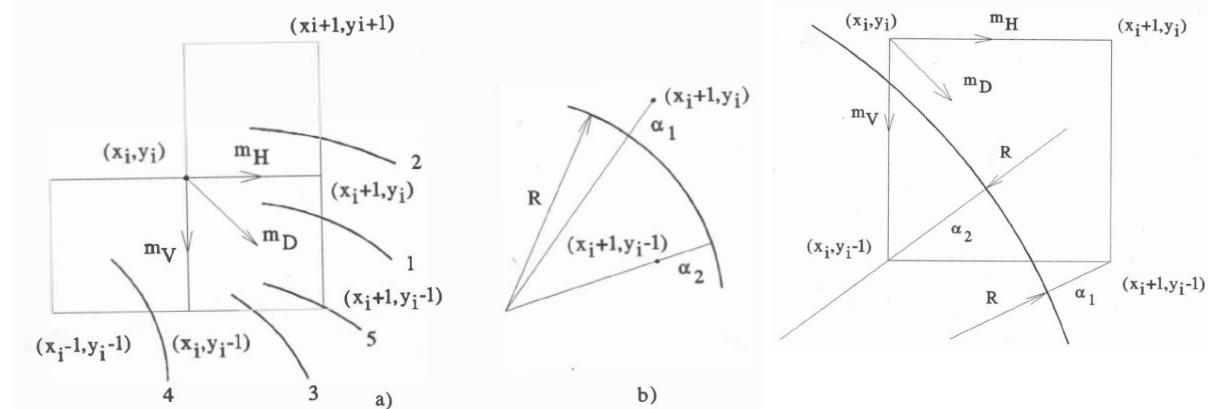
kde: $\varphi \in (0, 2\pi)$ a souřadnice (x, y) jsou počítány s malým krokem $\Delta\varphi$. Tento způsob vede na opětovný výpočet goniometrických funkcí, tj. $\cos \varphi$ a $\sin \varphi$. Body na kružnici však můžeme generovat také tak, že vhodně aplikujeme geometrické transformace, neboť

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos \Delta\varphi & -\sin \Delta\varphi \\ \sin \Delta\varphi & \cos \Delta\varphi \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (19.4)$$

např. pro počáteční pozici rotovaného bodu $x_0 = [R, 0]^T$.

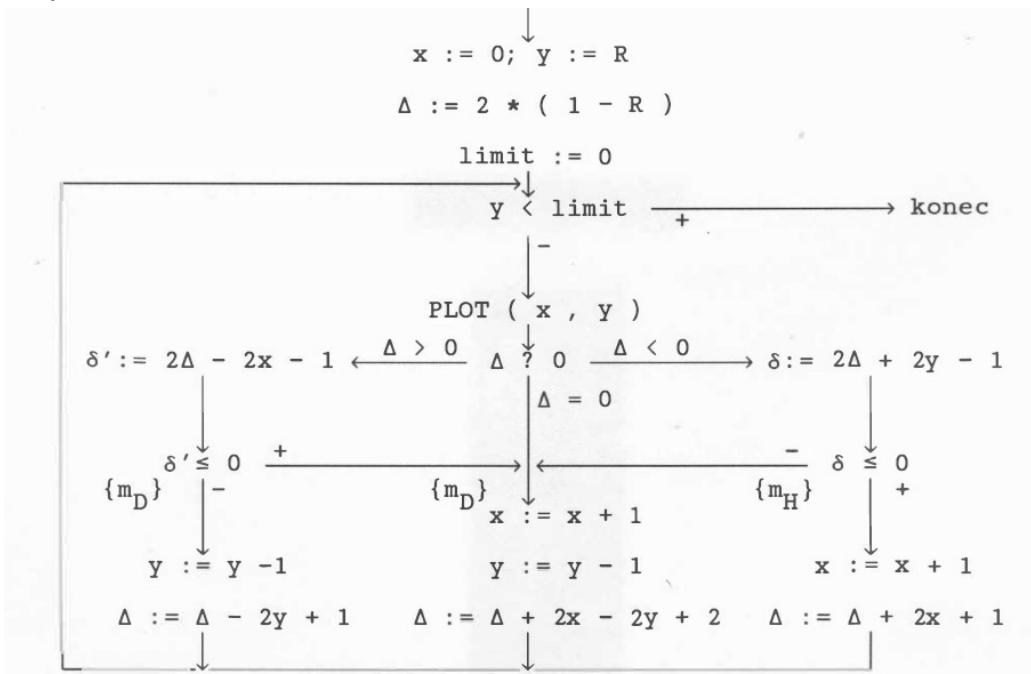
Pokud je kružnice generována v rastru, vzniká otázka, jak kružnici v rastrovém prostředí efektivně generovat? Efektivní algoritmus odvodil Jack Bresenham.

Předpokládejme, že jsme v pozici (x_i, y_i) a generujeme kružnici v prvním kvadrantu. Pak mohou nastat pouze situace, kdy generujeme krok buď horizontální m_H , nebo diagonální m_D nebo krok vertikální m_V . Obr.19.11 znázorňuje případy, které je nutné rozlišit. Algoritmus je založen na porovnávání vzdáleností α_1 a α_2 .



Obr.19.11: Specifikace případů Bresenhamova algoritmu pro kružnici

1 Důslednou analýzou problému se ukázalo, že algoritmus kreslení kružnice v rastrovém prostředí je
 2 možné opět realizovat bez použití operací v pohyblivé řádové čárce a bez operace násobení
 3 v celočíselné aritmetice, neboť vše je možné realizovat posuvem a sčítáním, resp. odečítáním. Vlastní
 4 algoritmus je uveden na Obr.19.12.



Obr.19.12: Bresenhamův algoritmus pro kreslení kružnice v rastrovém prostředí

5

6

Podrobné odvození algoritmu lze nalézt v:

- Skala,V.: Algoritmy počítačové grafiky I, 2011 ([Click off-line](#))

Z výše uvedeného je zřejmé, že algoritmus je velmi jednoduchý a snadno realizovatelný. Pro nakreslení celé kružnice je algoritmus nutné modifikovat, neboť kružnice je generována pouze v 1. kvadrantu. S výhodou lze použít symetrii.

13

14

15

Pokud se omezíme při kreslení pouze na 2. oktant, pak lze algoritmus ještě více zjednodušit a s využitím symetrie lze pak odvodit Michnerův algoritmus.

16

17

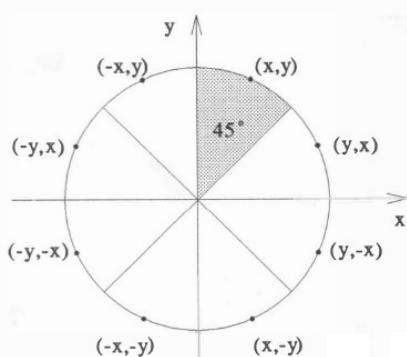
18

19

19.5. Michnerův algoritmus

2 Michnerův algoritmus je založen na symetrii a předpokládá, že obrazové pixely lze aktivovat
 3 v libovolném pořadí. Nicméně algoritmus se dá snadno upravit tak, aby kružnice byla vykreslována
 4 postupně, pokud je požadováno kreslení kružnice např. pomocí pera, nebo pro vykreslování
 5 kruhového oblouku.

6



7
8 Obr.19.13: Aplikace symetrie v Michnerově algoritmu
9

```

procedure MICHNER(R:integer);
var x, y, d: integer;
procedure PLOT8 ( x, y: integer );
begin PLOT ( x, y );      PLOT ( -x, -y );
      PLOT ( x,-y );     PLOT ( -x, y );
      PLOT ( y, x );      PLOT ( -y, -x );
      PLOT ( y,-x );     PLOT ( -y, x );
      { PLOT ( x, y ) aktivuje bod o souřadnicích (x,y) }
end;
begin x := 0;    y := R;    d := 3 - 2 * R;
      while x < y do
      begin PLOT8(x,y);
          if d < 0 then d := d + 4 * x + 6
          else begin d := d + 4 * ( x - y ) + 10;
                  y := y - 1
          end;
          x := x + 1
      end;
      PLOT8(x,y)
end;

```

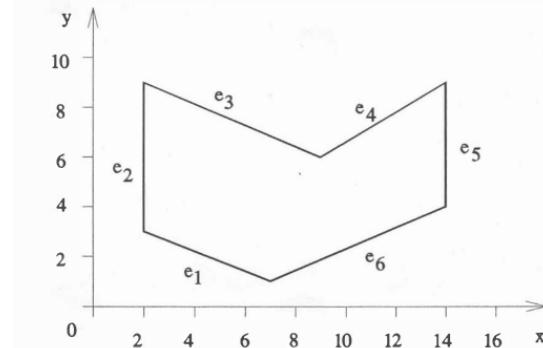
10
11 Obr.19.14: Michnerův algoritmus kreslení kružnice
12

13 Operace násobení se opět převedou na operace součtu a posuvu.

14

19.6. Řádková konverze

2 Řádková konverze (scan line) je technika, která je založena na jiné reprezentaci n-úhelníka. V zásadě
 3 jsou jednotlivé hrany n-úhelníka reprezentovány počátečním bodem, koncovou hodnotou y a
 4 směrnicí dané úsečky. Uvažme pro jednoduchost n-úhelník na Obr.19.15.

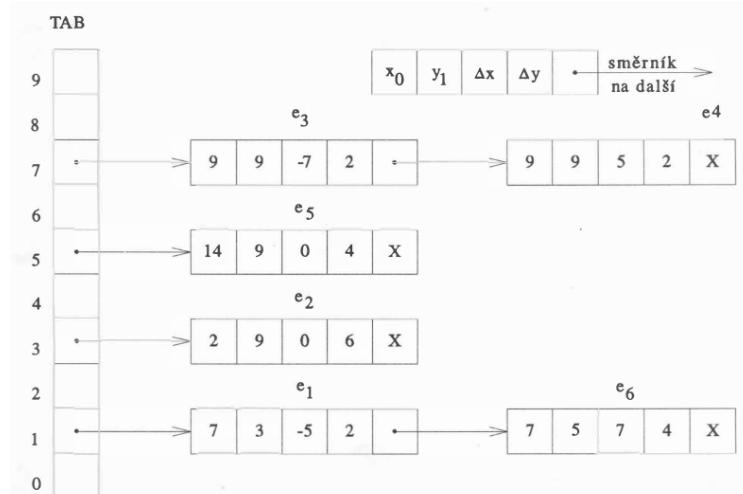


Obr.19.15: Daný n-úhelník

hrana	[y_0 , x_0 , y_1 , Δx , Δy]
e ₁	[1 , 7 , 3 , -5 , 2]
e ₂	[3 , 2 , 9 , 0 , 6]
e ₃	[7 , 9 , 9 , -7 , 2]
e ₄	[7 , 9 , 9 , 5 , 2]
e ₅	[5 , 14 , 9 , 0 , 4]
e ₆	[1 , 7 , 5 , 7 , 4]

Obr.19.16: Reprezentace hran

5
 6 Reprezentace založená na řádkové konverzi je vhodná zejména např. pro případ průsečíku přímky
 7 s nekonvexním n-úhelníkem, kdy se přímka a n-úhelník otočí tak, aby daná přímka byla rovnoběžná
 8 s osou x a vypočtené průsečíky se pak transformují zpět do správné polohy. V tomto případě se
 9 vlastně počítají hodnoty x pro jednotlivé průsečíky, neboť hodnota y je dána a je konstantní.



Obr.19.17: Generovaná datová struktura

10
 11 Tabulka TAB obsahuje pro jednotlivé hodnoty y ukazatele na ty hranы, které na dané
 12 hodnotě y „startují“. Kromě uvedené struktury se ještě udržuje uspořádaný seznam hran, které jsou
 13 pro danou řádku aktivní, tj. s nimi se pak např. počítá průsečík s danou přímkou. Tento seznam se
 14 označuje zkratkou AEL (Active Edge List). Seznam AEL se mění, postupně jak se realizuje vykreslování
 15 pro jednotlivé řádky. Řádková konverze se s výhodou používá např. pro operaci šrafování n-úhelníka,
 16 řešení viditelnosti parametrických ploch atd.

17
 18 Pro operace plnění a šrafování v diskrétním prostředí se používá její modifikace – rastrová řádková
 19 konverze, viz např.:
 20

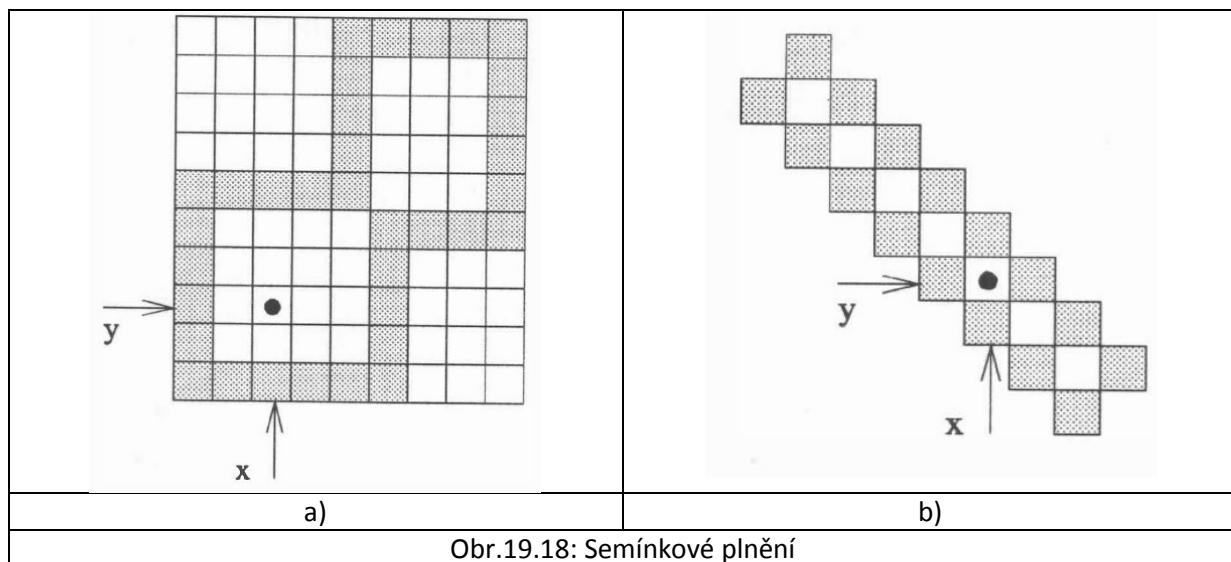
- 21 • Skala,V.: Algoritmy počítačové grafiky I, pp.81, 2001 ([CLICK off-line](#))

19.7. Algoritmy plnění a šrafování

Operace plnění, šrafování, resp. plnění vzorem, jsou poměrně časté operace, zejména pak v oblasti strojírenství a stavebnictví. Plnění barvou je běžnou operací ve většině grafických výstupů. V současné době jsou asi nejčastěji používané metody určené pro rastrové prostředí, které navíc umožňují efektivní řešení v hardwaru.

19.7.1. Semínkové plnění

Semínkové plnění je nejjednodušším algoritmem pro plnění, který je založen na „obarvení“ sousedních bodů k danému bodu (x, y) , pokud již nejsou body hranice. Algoritmus semínkového plnění je založen na rekurzi a jeho přímá programová implementace je velmi neefektivní. Algoritmus semínkového plnění v některých případech nevyplní celý n-úhelník, viz Obr.19.18.a, kdy se vyplní pouze spodní část n-úhelníka a do druhé části se algoritmus „nepřelije“. Také pro velmi štíhlé objekty, viz Obr.19.18.b, dochází k neúplnému vyplnění.



Obr.19.18: Semínkové plnění

Vlastní algoritmus lze pak zapsat sekvencí:

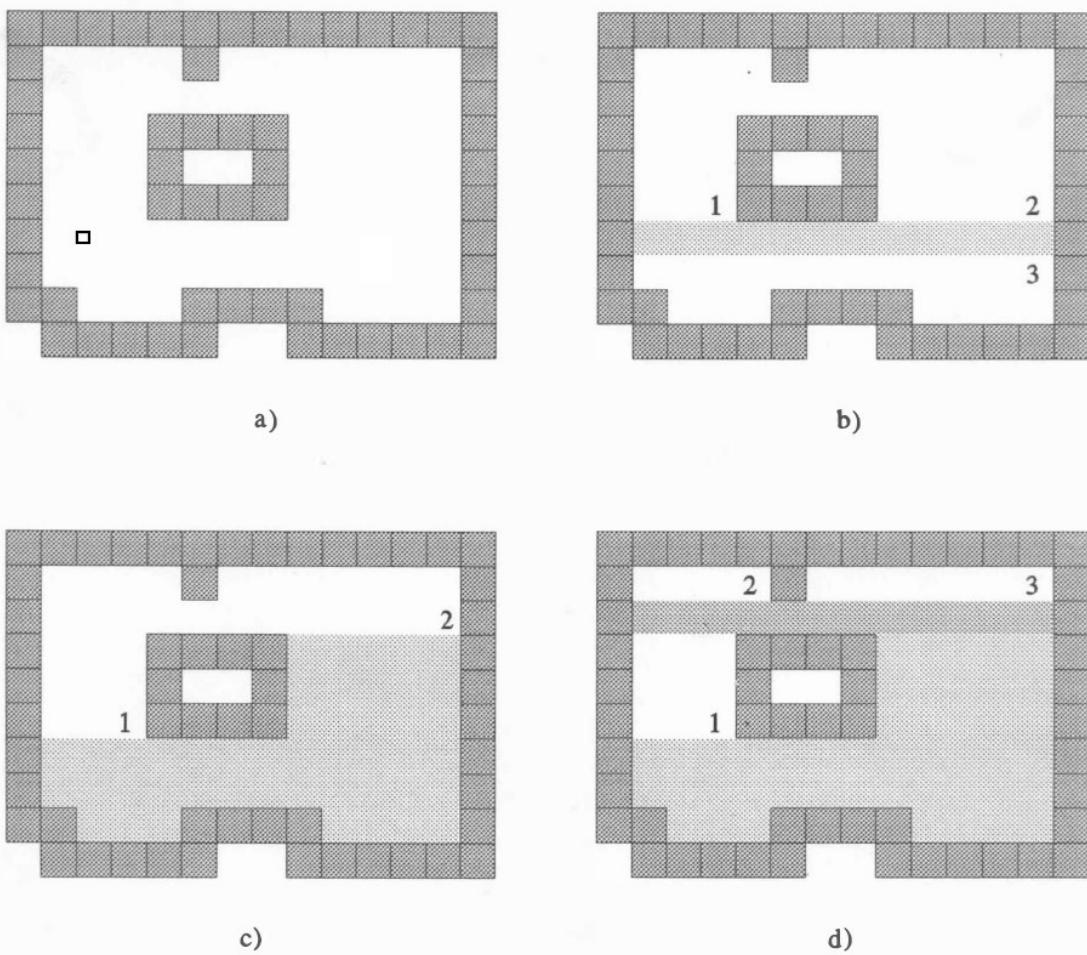
```

procedure SEED FILL ( x,y, { startovací bod }
                      bc, { barva hranice }
                      fc: integer { barva plnění } );
var i: integer;
begin i:= Get Pixel Color ( x,y ); { vrací barvu pixelu (x,y) }
  if (i <> bc) and (i <> fc) then
    begin Set Pixel Color ( x , y , fc );
       { nastaví barvu pixelu (x,y) }
       SEED FILL ( x+1 , y , bc , fc );
       SEED FILL ( x-1 , y , bc , fc );
       SEED FILL ( x , y+1 , bc , fc );
       SEED FILL ( x , y-1 , bc , fc );
    end
  end { SEED FILL };

```

1 **19.7.2. Řádkové semínkové plnění**

2 Pro efektivní implementaci se používá nerekurzivní modifikace založená na řádkové konverzi. Pro
3 názornost uvažme n-úhelník na Obr.19.19.



Obr.19.19: Algoritmus řádkového plnění

7 Nerekurzivní verze semínkového plnění je popsána níže uvedeným algoritmem.

```

procedure SCAN SEED FILL ( x,y, { startovací bod }
                           bc, { barva hranice }
                           fc: integer { barva plnění } );
procedure Check and Set ( xl, xr, { rozsah řádku }
                         y: integer { kontrolovaná řádka } );
var x,x0: integer;      flag: boolean;
begin
  x:=xl;
  while x < xr do
  begin
    flag:=false;
    while ( Get Pixel Color (x,y) <> bc )
           and ( Get Pixel Color (x,y) <> fc )
           and ( x < xr ) do
    begin flag:=true; x := x + 1 end;
  
```

```

    { ulož do zásobníku pixel, který je nejvíce vpravo }
if flag then
begin if ( x = xr ) and ( Get Pixel Color (x,y) <> bc )
        and ( Get Pixel Color (x,y) <> fc )
        then PUSH (x,y,s) else PUSH (x-1,y,s);
flag:= false
end;
x0 := x;
while (( Get Pixel Color (x,y) = bc )
       or ( Get Pixel Color (x,y) = fc ))
       and ( x < xr ) do
       x := x + 1;
if x = x0 then x := x + 1
end { while }
end { Check and Set };
begin
PUSH (x,y,s);           { ulož souřadnice bodu do zásobníku s }
while not EMPTY (s) do      { pokud není zásobník prázdný }
begin
POP (s,x,y);           { vyber ze zásobníku souřadnice bodu }
Set Pixel Color (x,y,fc);
{ vyplň řádku nalevo }
x0 := x; x := x + 1 ;
while Get Pixel Color (x,y) <> bc do
begin Set Pixel Color (x,y,fc); x := x + 1 end;
xr := x - 1; x := x0;      { obnovení původní souřadnice x }
x := x - 1;
while Get Pixel Color(x,y)<>bc do
begin Set Pixel Color(x,y,fc); x := x - 1 end;
xl := x + 1;           { <xl,xr> je nyní interval plnění }
{ Nyní je nezbytné určit, zda všechny pixely nad nebo pod }
{ zpracovávanou řádkou jsou body hranice nebo již byly   }
{ aktivovány. Pokud tomu tak není, pak je nezbytné      }
{ pokračovat v plnění }
Check and Set (xl,xr,y+1);
Check and Set (xl,xr,y-1);
end { while }
end { SCAN SEED FILL };

```

Algoritmus řádkového plnění

Zkusme nyní krátce vysvětlit princip algoritmu. Po zadání startovacího bodu se vyplní 1. řádka. Při vyplňování této řádky se do zásobníku uloží ty body, které jsou těsně před hranicí na řádce nad a pod danou řádkou. Tyto body pak slouží jako nové startovací body. Lze nalézt mnoho algoritmů a jejich modifikací, které využívají např. přímo frame-bufferu apod., nebo jsou založeny přímo na aplikaci řádkové konverze, např. viz

- Skala,V.: Algoritmy počítačové grafiky I, pp.94, 2001 ([CLICK off-line](#))

20. Vizualizace informací a dat

Vizualizace dat v obecném slova smyslu je vlastně prostředníkem pro předání cílově orientované informace uživateli, který tuto informaci potřebuje ke své další činnosti. Z tohoto vyplývá jeden podstatný faktor - a to, že vizualizace je poskytovatelem informace a je oborově různorodá, přičemž některé reprezentace dat a některé metody mohou být totožné, resp. velmi podobné.

Vizualizace informací a dat se dělí na dvě hlavní části, a to:

- vizualizaci informací - jde převážně o textová, historická data, data získaná z různých databází zkoumající především vztahy mezi daty, data statistická apod. Uveděme např. vizualizace textů a dokumentů, vizualizace grafů a sítí, vizualizace časových řad, vizualizace softwaru, vizuální systémy pro spolupráci více uživatelů atd.
- vizualizaci dat – jde zejména o technická data získaná simulací fyzikálních jevů nebo měřením, např. elektromagnetické pole, teplotní pole apod., data lékařská, např. CT a MRI data, data z ultrazvuku, data chemická, např. vizualizace proteinu a chemických struktur, data biologická, např. DNA, RNA apod., data astronomická, např. vizualizace jevů ve vesmíru, mlhovin apod.

Podívejme se nejdříve na vizualizaci informací a dat z pohledu historie.

20.1. Vizualizace informací

Vizualizace hraje důležitou úlohu v lidské činnosti již od časných dějin lidstva. Pravděpodobně první zmínky pocházejí z období raného hinduismu (slovo *Hindu* je původem z arabského výrazu *al-Hind*, které označuje území okolo řeky Indus) (<http://en.wikipedia.org/wiki/Hinduism>) v pozdním Neolitu (http://cs.wikipedia.org/wiki/P%C5%99eddynastick%C3%A1_doba), tj. cca 5.-2.století př.n.l., přes období Egypta, tj. cca 5.-3.století př.n.l. a období Babylonu, tj. cca 1750-130 let př.n.l., města ležícího na březích řeky Eufratu na území dnešního Iráku a Iránu.



Obr.20.1: Babylon

Pieter Breugel, 1526/1530–1569

<http://cs.wikipedia.org/wiki/Babyl%C3%B3n>



Obr.20.2: Mapa Babylonu

WiKi & Radomił Binek

<http://cs.wikipedia.org/wiki/Babyl%C3%B3n>

Nicméně archeologické výzkumy ukazují, že i na území Evropy lze nalézt první známky o vizualizaci dat v podobě map. První zmínky lze datovat dokonce do období 12 tis. let př.n.l.

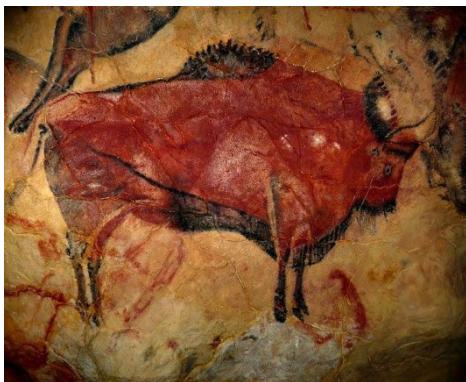


Obr.20.3: Nejstarší mapa (Abauntz, Navarra, Spain) - Courtesy of „The Telegraph“
<http://www.telegraph.co.uk/news/worldnews/europe/spain/5978900/Worlds-oldest-map-Spanish-cave-has-landscape-from-14000-years-ago.html>

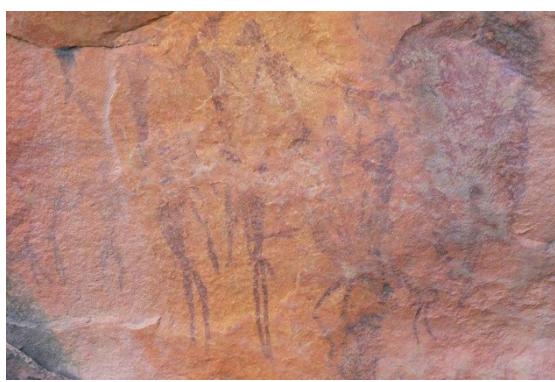


Obr.20.4: Pravděpodobně nejstarší kresba El Castillo, Francie, doba Neandrtálská - Courtesy of „National Geographic“
<http://news.nationalgeographic.com/news/2012/06/120614-neanderthal-cave-paintings-spain-science-pike/>

- 1 Pravděpodobně nejznámější jsou pak jeskynní kresby.



Obr.20.5: Altamira, Spain – oblast Santander
http://en.wikipedia.org/wiki/Cave_painting

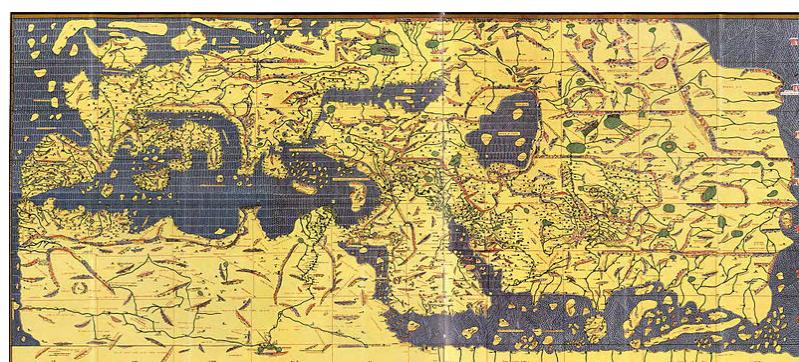


Obr.20.6: „Bush men“, Jižní Afrika
http://en.wikipedia.org/wiki/Cave_painting#Africa

- 2 Další historické malby: <http://en.wikipedia.org/wiki/Lascaux>
 3 Je tedy zřejmé, že již v raném období lidstva dochází k pokusům o zaznamenání nějaké situace, ať už geografické, např. mapy apod., nebo výjevů z lovu, resp. z denního života apod. za účelem předání informace ostatním. S rozvojem obchodu v raných dobách bylo nutné nějakým způsobem zaznamenat obchodní trasy, místa kam obchodníci došli, atd. Pravděpodobně nejstarší dochovanou mapou je Babylonská mapa světa z 9. stol. př.n.l.

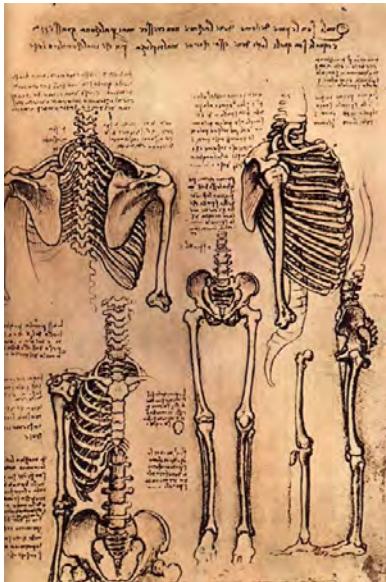


Obr.20.7: Babylonská mapa světa - WiKi

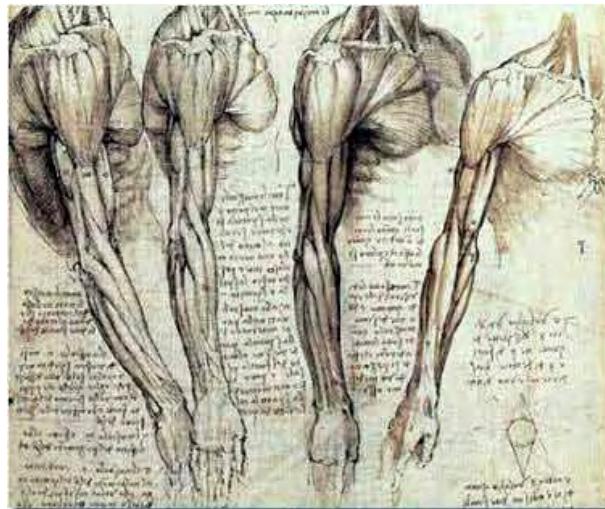


Obr.20.8: Tabula Rogeriana zakreslené Muhammem al-Idrisi v roce 1154 pro Rogera II Sicilského - WiKi

- 1 Z mapy Tabula Rogeriana (<http://en.wikipedia.org/wiki/Cartography>) je vidět, že zřejmě v té době již obchod čile kvetl a tvorba mapy byla toho důsledkem.
- 2 Pravděpodobně jedním z největších „vizualizérů“ byl Leonardo da Vinci, jehož kresby se zabývaly jak anatomií, tak i lidským tělem.

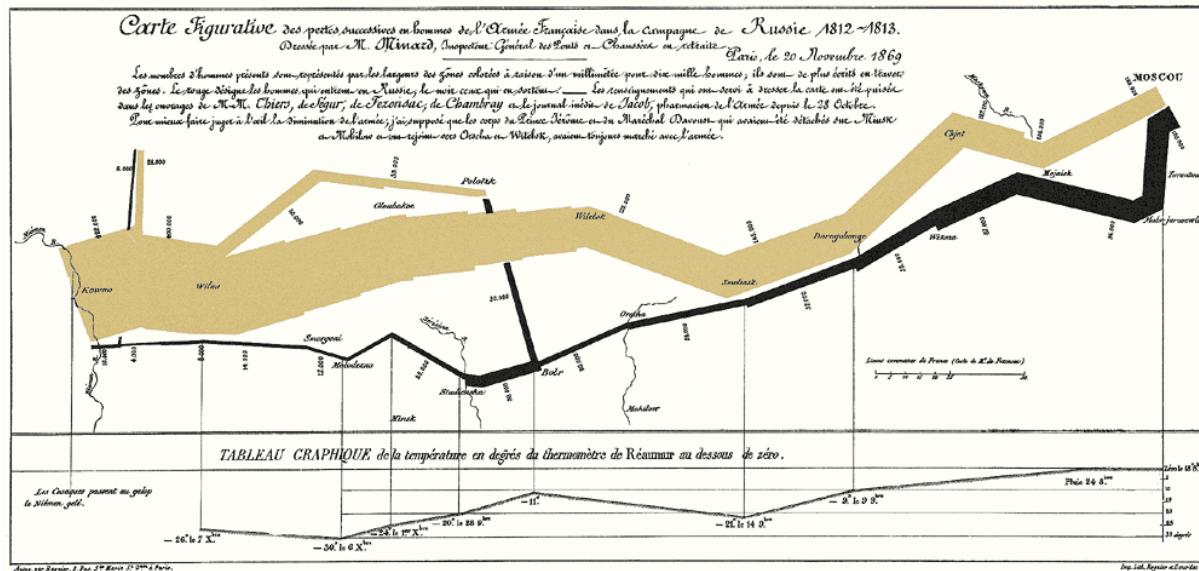


Obr.20.9: Leonardo-Skeleton-Wiki



Obr.20.10: Leonardo-Hand-Wiki

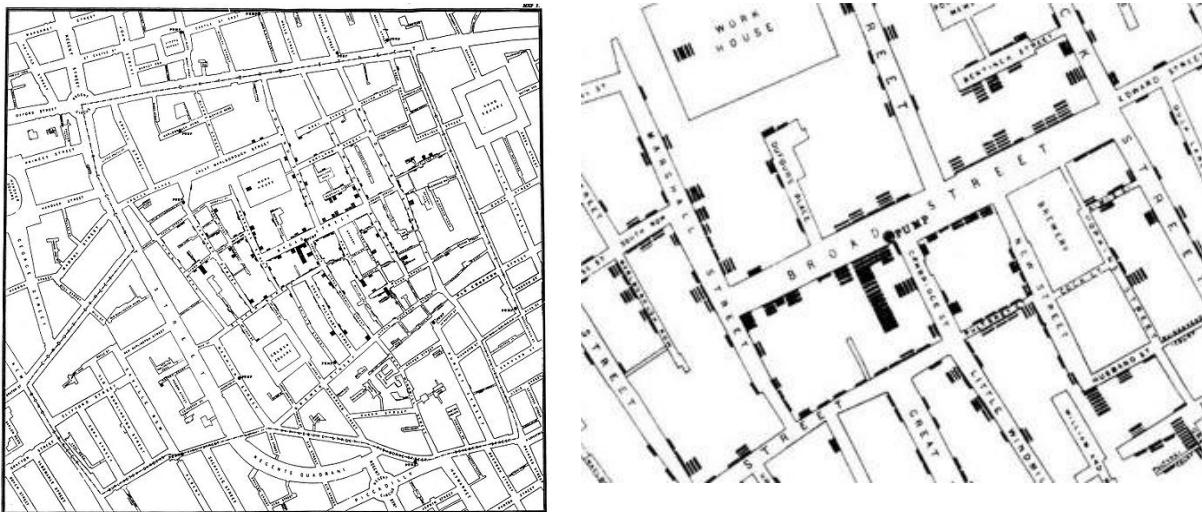
- 5
- 6 U studií lidského těla bychom to dnes označili jako „non-realistic rendering“, tj. nerealistické
- 7 zobrazování.
- 8
- 9 Je zcela pochopitelné, že způsoby historických zobrazení jsou dány jednak vývojovou úrovní lidí a
- 10 také dostupnými prostředky. V současné době technologie nabízí nepřeberné technologické a
- 11 komunikační prostředky a rovněž nepředstavitelné možnosti pro vizualizaci dat a informací
- 12 v obecnějším slova smyslu.
- 13
- 14 Dnes již klasickou ukázkou vizualizace je znázornění Napoleonova tažení na Moskvu v letech
- 15 1812-1813 (http://en.wikipedia.org/wiki/French_invasion_of_Russia). Charles Joseph Minard
- 16 znázornil, jak postupně francouzská armáda, která měla téměř 500 tisíc vojáků, postupně ztrácela na
- 17 síle. Bitvu u Borodina blízko Moskvy, kde zemřelo, či bylo zraněno přes 70 tisíc vojáků, Napoleon sice
- 18 vyhrál, ale dostal se jen do hořící Moskvy. Zima a nedostatek potravin pak způsobila další decimaci
- 19 francouzské armády při návratu zpět. Francouzská armáda ztratila přes 380 tisíc vojáků. Pouze
- 20 120 tisíc vojáků se vrátilo z tažení zpět. Mapa dobré znázorňuje velikost armády a skutečnost, jak se
- 21 její velikost v průběhu času měnila, dále geografické pozice pochodu armády, směr jejího pohybu na
- 22 cestě k Moskvě a zpět, povětrnostní podmínky podél zpáteční cesty atd. Minardova mapa se pokládá
- 23 za jednu z nejlepších kreslených animací.



Obr.20.11: Minardova mapa Napoleoova težení na Moskvu

Další velmi známou kreslenou vizualizací je mapa výskytu cholery v Londýně v Soho v oblasti Broad Street v roce 1853-1854, kterou vytvořil lékař Dr.John Snow. Zakreslením více než 500 obětí, které zemřely v prvních 10 dnech v září 1854 a analýzou jednoduché vizualizace vytvořené z poměrně obsáhlé datové množiny, dospěl k překvapivému odhalení spojitosti mezi výskytem cholery a používaného vodního zdroje. Dr.Snow odstranil páku na pumpování a dle historických záznamů tak zachránil přes 500 osob.

Můžeme jen spekulovat, jak by postupoval Dr.Snow při použití dnešní technologie a znalostí.



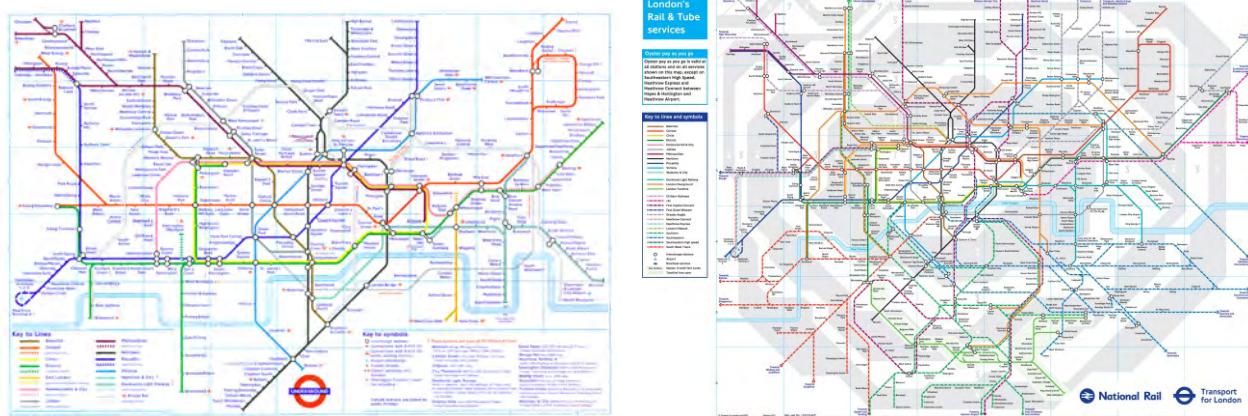
Obr.20.12: Snowova mapa výskytu cholery a její detail

Mapy jsou pravděpodobně jednou z nejstarších a nejčastějších vizualizací, neboť vycházely z reálných potřeb lidí. V současné době, kdy lidé žijí ve velkých aglomeracích, městech a megapolích, kde počet obyvatel přesahuje 10 mil. (např. Shanghai 18 mil., Istanbul 14 mil., Karachi 13 mil atd.) je kromě jiného, potřeba mít přehled o linkách městské dopravy, tj. autobusů, metra atd. Na mapách metra v Londýně si ukažme reprezentativní vizualizace. V zásadě jde o dva rozdílné přístupy, a to:

- „abstraktní“ – zobrazující přehledně jednotlivé linky, přestupní stanice bez ohledu na geografické uspořádání. Tento typ schématu je vhodný pro rychlou orientaci, pokud je známo z které výchozí a do které cílové stanice chceme jet. Nicméně nám neříká nic o skutečné poloze stanic, resp. kudy trasa linky metra vlastně vede. Pro rychlou a základní orientaci je

1 výhodná abstraktní schematická mapa. Její původní návrh udělal Harry Beck v roce 1931 a její
 2 princip se používá dodnes. Obr.20.13.a zobrazuje standardní schéma metra v Londýně.
 3 Obr.20.13.b využívá ještě podbarvení k zachycení jednotlivých cenových zón

- 4 • „geograficky“ orientované – trasy linek jsou zakresleny do mapového podkladu tak, aby
 5 uživatel měl představu o poloze stanic. Tento typ je vhodný zejména v případě, že hledáme
 6 stanici metra v dané geografické pozici a chceme najít vhodné spojení do cílové stanice.

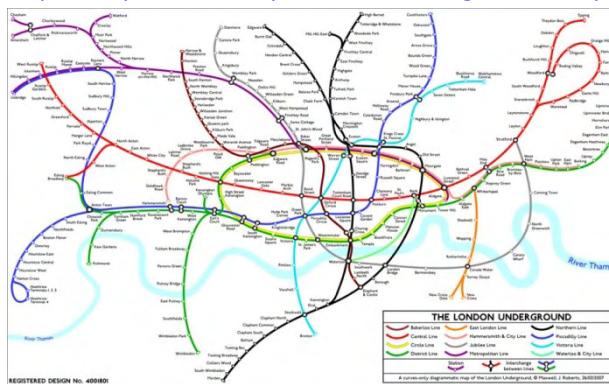


a) Podle původního návrhu Harry Beck 1931

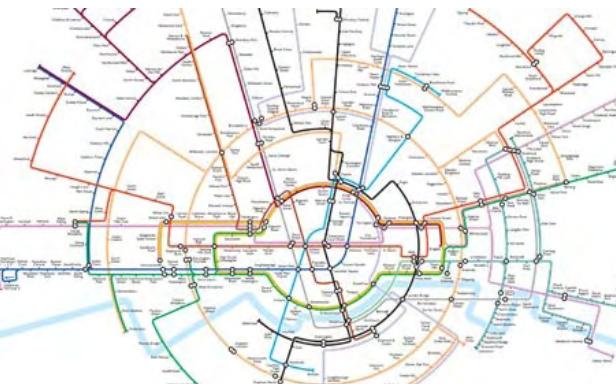
Obr.20.13: Mapa metra v Londýně

b) Současně používaná mapa

7 <http://mapsof.net/map/london-underground-map>



Obr.20.14: Maxwell Roberts



Obr.20.15: Dr.Max Roberts, Guardian, 2013

<http://www.guardian.co.uk/news/datablog/2013/jan/31/circle-line-tube-map-visualised#>

8 O tom, že Beckův návrh byl geniální, svědčí i to, že tento přístup používá např. Tokyo a Singapore.



Obr.20.16: Mapa metra a vlaků v Tokyu



Obr.20.17: Mapa metra v Singapuru

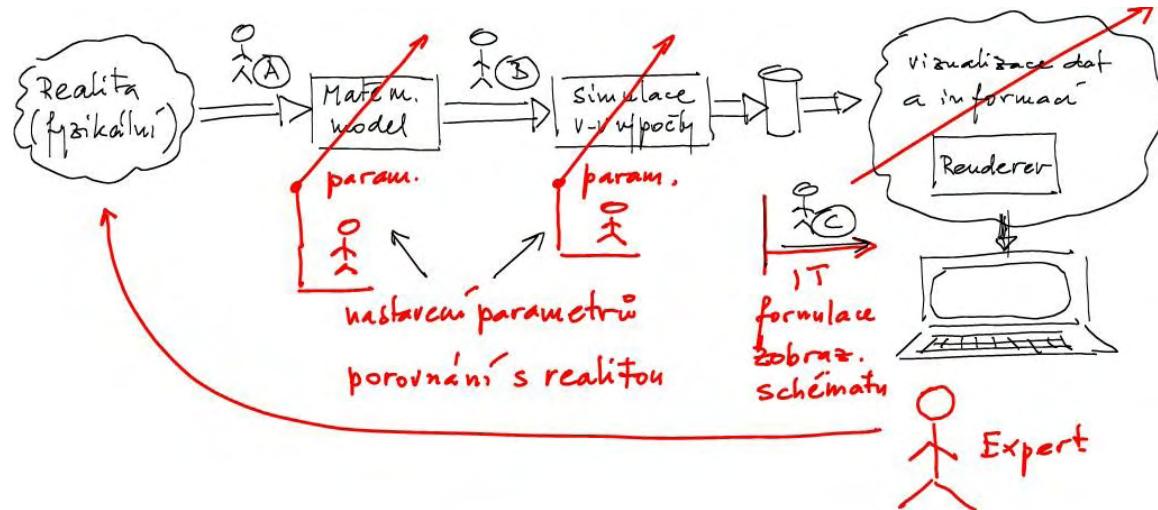
9 Pochopitelně je možné obdobný přístup aplikovat i např. na silniční síť atd.

20.2. Vizualizace dat

Vizualizace dat je většinou vázána na děje v okolním světě. Většinou se jedná o zpracování naměřených dat nebo dat získaných simulací, ať již fyzikální nebo biologické podstaty. Při zpracování se většinou jedná o zpracování statických nebo dynamických dat, která mohou být skalární, vektorová či vícerozměrná.

20.2.1. Modelování a simulace reality

Existující realita, nejen fyzikální, je obvykle zkoumána a modelována, aby ji bylo možné pochopit a následně pak získané znalosti, např. zákonitosti, využít pro další zpracování apod. Tento krok je v mnoha případech realizován návrhem matematicko-fyzikálního modelu.



Obr.20.18: Schematické znázornění kontextu vizualizace dat

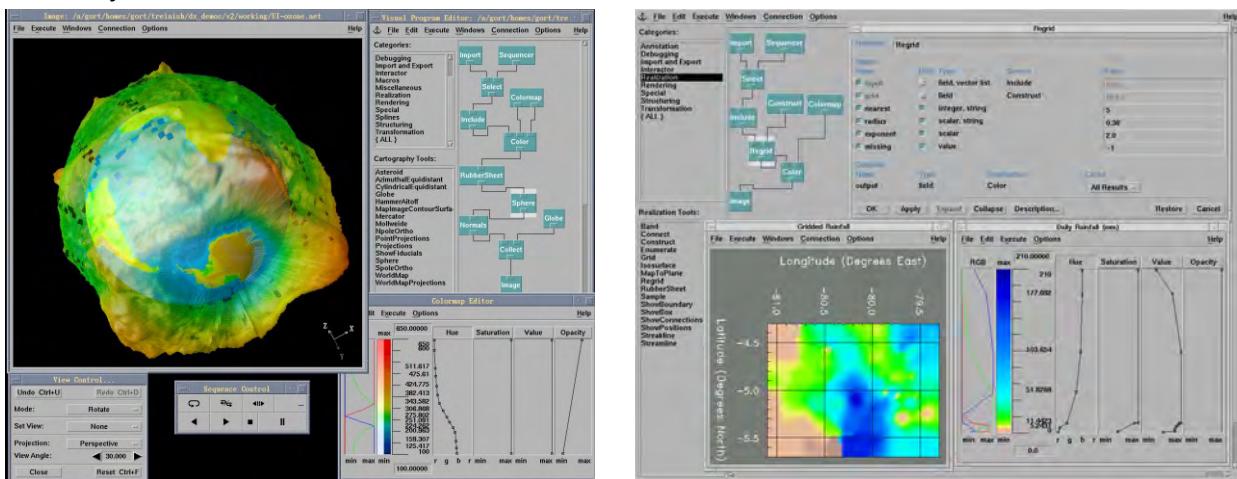
Matematicko-fyzikální model zřejmě také bude mít ve většině případů nějaké parametry, které bude nutné nastavit tak, aby co nejlépe vystihovaly modelovanou realitu. Zde je nutné zdůraznit, že každý matematicko-fyzikální model bude zajisté reprezentovat hlavní modelované vlastnosti, nicméně vzhledem k bohatosti mnoha aspektů okolního světa model zřejmě:

- bude dobře reprezentovat, po vhodném nastavení parametrů, modelovanou realitu
- nebude schopen postihnout určité aspekty, pro jejichž modelování je nutné vytvořit specializovaný model
- bude umožňovat „reprezentovat“ situace, které nejsou v reálném světě možné a je nutné při interpretaci výsledků tyto případy detektovat

Po vytvoření matematicko-fyzikálního modelu zřejmě půjde o výpočetní realizaci, resp. simulaci chování navrženého výpočetního modelu, dále o ověření vlastností modelu v různých situacích a opět zřejmě bude nutné nastavovat různé parametry vlastního výpočtu, resp. simulace. K vlastnímu výpočtu mohou být použity různé výpočetní prostředky, včetně paralelních a distribuovaných „high performance“ systémů. Data získaná výpočtem jsou u velkých úloh většinou ukládaná na paměťová media k „off-line“ vyhodnocování a vizualizaci, nebo je vyhodnocování a vizualizace realizována „online“.

Úloha vizualizace dat je vlastně úlohou, ve které se použijí standardní techniky pro vizualizaci dat, nebo se nové metody navrhnu, realizují a následně se provádí vlastní explorační výpočtených dat s použitím zvolené, či nově realizované zobrazovací techniky. Návrh vlastního vizualizačního postupu

není nijak triviální záležitostí. Ani explorace dat při různých parametrech a při volbě různých metod také není jednoduchou záležitostí.



Obr.20.19: IBM Data Explorer - Typická situace při vizualizaci dat

Z výše uvedeného je zřejmé, že v netriviálních úlohách:

- jde o aktivity skupinové, do kterých jsou zapojeni experti z různých oblastí
- pracovník, který má na starosti vizualizační část, musí mít základní znalosti jak faktické, tak i terminologické z dané oblasti, aby byl schopen se domluvit s ostatními a pochopit, co se od něj očekává.

Vizualizace dat je tedy interdisciplinární ve své podstatě, a proto odborné aktivity musejí vycházet z potřeb dané oblasti.

11

20.2.2. Typy dat

Při zpracování dat obvykle nepracujeme pouze se samostatnými údaji, neboť zpracováváme poměrně velké datové množiny. V současné době je objem „malých“ datových množin obvykle v řádu do několika [GB], zatímco velká data dosahují [TB] nebo [PB]. V každém případě je vhodné data nějakým způsobem shlukovat, strukturovat atd., abychom se v nich mohli nějakým způsobem úsporně „pohybovat“.

$$\{\mathbf{x}, \mathbf{h}\} = \{[\mathbf{x}, \mathbf{y}, \mathbf{z}]^T, \mathbf{h}\} \quad (20.1)$$

Ve fyzikálně-matematických orientovaných aplikacích data obvykle mají svoje vlastní uspořádání, které můžeme reprezentovat uspořádanou dvojicí:

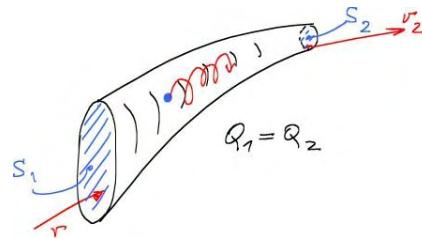
$$\{\mathbf{x}, \mathbf{h}\} = \{[\mathbf{x}, \mathbf{y}, \mathbf{z}]^T, [\mathbf{h}_1, \dots, \mathbf{h}_m]^T\} \quad (20.2)$$

tj. pro E^3 dostáváme vlastně uspořádanou dvojici hodnot, kde \mathbf{x} např. reprezentuje pozici částice v prostoru, tj. x, y, z . S každou částicí je spojena celá řada „asociovaných“ hodnot \mathbf{h}_i , $i = 1, \dots, m$. Některé specifikují v daném bodě skalární veličiny, např. teplotu, tlak, vlhkost atd., jiné pak vektorové, např. rychlosť, zrychlení apod., a případně hodnoty tenzorové, např. namáhání pro výpočty v oblasti pevnosti a pružnosti.

Pro obecný d -rozměrný případ pak dostáváme:

$$\{\mathbf{x}, \mathbf{h}\} = \{[\mathbf{x}_1, \dots, \mathbf{x}_d]^T, [\mathbf{h}_1, \dots, \mathbf{h}_m]^T\} \quad (20.3)$$

Uvažme pro názornost velmi jednoduchý případ zahrádkní trysky:



a) Zahradní tryska

b) Fyzikální náhled na proudění v trysce

Obr.20.20: Proudění v zahradní trysce

1 Pro nestlačitelné kapaliny z Bernoulliho rovnice vyplývá:

2

$$Q_1 = Q_2 \quad S_1 v_1 = S_2 v_2 \quad (20.4)$$

3 Tedy, že množství kapaliny, které „vteče“, se musí rovnat množství kapaliny, která vyteče. Pokud
 4 $S_1 \neq S_2$, pak také $v_1 \neq v_2$ a tedy uvnitř trysky se musí měnit také rychlosť, proto akcelerace částečky
 5 je nenulová. To znamená, že pro každou částečku musíme reprezentovat její aktuální rychlosť
 6 $\mathbf{v} = [v_x, v_y, v_z]^T$ a její aktuální zrychlení $\mathbf{a} = [a_x, a_y, a_z]^T$. Každá částečka může ještě uvnitř rotovat,
 7 takže máme další vektor $\mathbf{rot} = [rot_x, rot_y, rot_z]^T$.

8 Také i na tomto jednoduchém případě vidíme, že kromě pozice částečky \mathbf{x} , musíme ukládat hodnoty
 9 \mathbf{h}_i , což je v našem případě vlastně více než 10 asociovaných hodnot pro každou částečku v trysce.

10 Abychom mohli spočítat a následně zobrazit situaci v trysce, pak musíme zpracovat n částeček, kde
 11 $n \gg 10^9$. Pak zpracovávaná množina $\Omega = \{\mathbf{x}, \mathbf{h}\}_1^n$, pokud budeme uvažovat jednoduchou přesnost,
 12 bude $(3 * 4 + 11 * 4) * 10^9 = 56$ [GB]. Nicméně pro reálný výpočet bychom potřebovali
 13 pravděpodobně podstatně více „částeček“.

14 Až dosud jsme však uvažovali velmi specifický případ, kdy data jsou statická, což není obecný případ
 15 při řešení fyzikálně-matematických problémů. Z uvedeného je zřejmé, že zpracovávaná data můžeme
 16 rozdělit do několika skupin, pro něž zřejmě budeme mít nejen odlišné výpočetní a zobrazovací
 17 metody, ale i jiné reprezentace dat.

Asociované hodnoty \mathbf{h}			
	Statické	Dynamické (t-variantní)	
Souřadnice \mathbf{x}	Statické	$\Omega = \{\mathbf{x}, \mathbf{h}\}_1^n$	$\Omega = \{\mathbf{x}, \mathbf{h}(t)\}_1^n$
	Dynamické (t-variantní)	$\Omega = \{\mathbf{x}(t), \mathbf{h}\}_1^n$	$\Omega = \{\mathbf{x}(t), \mathbf{h}(t)\}_1^n$

18

Tabulka xxxxx

19 V případě dynamických dat je ještě jeden velmi důležitý aspekt, jak data jsou v časové ose získávána.

20 Proto lze dynamická data dále dělit podle toho, jak jsou získávána, a to:

- 21 • synchronně, tj. zda v daný okamžik jsou získána všechna odpovídající $\{\mathbf{x}(t), \mathbf{h}(t)\}$ hodnoty.
 22 Jde o velmi častý případ v praxi, který se také někdy označuje termínem „frame-by-frame“.
 23 Výhodou tohoto režimu je, že lze použít standardní interpolační metody

- 1 • asynchronně – typickým případem mohou být plovoucí sondy v moři, které vyšlou data,
 2 pokud např. vybočí z očekávaného intervalu hodnot. V tomto případě nelze použít obvyklé
 3 interpolaci metody, protože nejsou k dispozici všechny hodnoty pro jednotlivé časové
 4 úseky, tj. pro jednotlivé časové okamžiky t_i .

5 Z výše uvedeného dělení vidíme, že data sama o sobě mají poměrně pestrou paletu možností, které
 6 nespecializované vizualizační systémy musí umožňovat.

7 S časově proměnnými daty je spojena ještě jedna podstatná komplikace, a to výpočet vzdálenosti.
 8 Vzdálenost je v Eukleidovském prostoru definována jako:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad (20.5)$$

9 V mnoha systémech je v případě t-variantních dat používán výraz:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2 + (\Delta t)^2} \quad (20.6)$$

10 což je pochopitelně špatně, neboť sčítáme „hrušky“ a „jablka“ dohromady. Je nutné si uvědomit, že
 11 Δz má rozměr [m], zatímco Δt má rozměr [s]. V astronomii je výpočet vzdálenosti s určen jako
 12 interval Minkowského prostor-čas:

$$s = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2 - c^2(\Delta t)^2} \quad (20.7)$$

13 kde c je rychlosť světla. Zdá se, že pro fyzikální úlohy by vzdálenost měla být tedy vzdálenost
 14 definována jako:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2 - \gamma^2(\Delta t)^2} \quad (20.8)$$

15 kde γ je rychlosť šíření daného fenoménu (ne nezbytně fyzikálního) v daném prostředí. Na rozdíl od
 16 rychlosti světla je zřejmé, že hodnota $\gamma = \gamma(*)$ je závislá na dalších parametrech, např. na teplotě,
 17 tlaku, atd.

20.2.3. Datové struktury

20 Data lze z pohledu uspořádanosti, např. z hlediska jejich prostorového uspořádání, rozdělit takto:

- 21 • *neuspořádaná* - tato data nemají žádnou implicitně ani explicitně danou strukturu; jde tedy o
 22 jakousi „hromadu“ dat bez vnitřního uspořádání. Zpracovávají se pouze data obsažená
 23 v datovém setu Ω .
 - 24 ○ *roztroušená (scattered)* – obecně nejsou známy žádné specifické charakteristiky,
 25 občas jsou informace o rozložení dat, např. rovnoměrné, normální (Gaussovo) resp.
 26 Poissonovo atd.
 - 27 ○ *shluková (clustered)* – data vytvářejí v datovém prostoru shluky, které mohou mít
 28 specifické vlastnosti
- 29 • *uspořádaná* – tato data mají nějakou dodatečnou informaci o jejich vnitřním uspořádání,
 30 resp. strukturalizaci.
 - 31 ○ *nestrukturovaná* – typickou ukázkou jsou trojúhelníkové a čtyřstěnné sítě, kdy vrchol
 32 sítě nemá pevný počet trojúhelníků, nebo čtyřstěnů, které daný vrchol sdílejí.
 33 v případě trojúhelníkové sítě trojúhelníky mohou mít různý tvar, tj. různý poměr
 34 délek hran, přičemž se preferují většinou tvary blízké rovnostrannému trojúhelníku.
 35 V některých aplikacích, zejména v GIS (Geographical Information Systems)
 36 systémech, se používají rovnostranné pravoúhlé trojúhelníkové sítě, které se ale
 37 vlastně považují za strukturované a u kterých se dobře dělá adaptivní dělení na
 38 jemnější síť.
 - 39 ○ *strukturovaná* – mají pevnou strukturu, tj. vrchol je sdílen konstantním počtem
 40 buněk, např. trojúhelníků apod. Obecně struktura může být pravoúhlá nebo

1 nepravoúhlá. V daném datovém setu je jednoznačná mapovací funkce mezi buňkou a
 2 vrcholy tuto buňku sdílející

- 3 • *pravidelná* – pozice vrcholů je možné určit přímo z indexů, obvyklá reprezentace pro
 4 pravoúhlou strukturu je matice hodnot.

5 To znamená, že pro datový set je nutné si pamatovat:

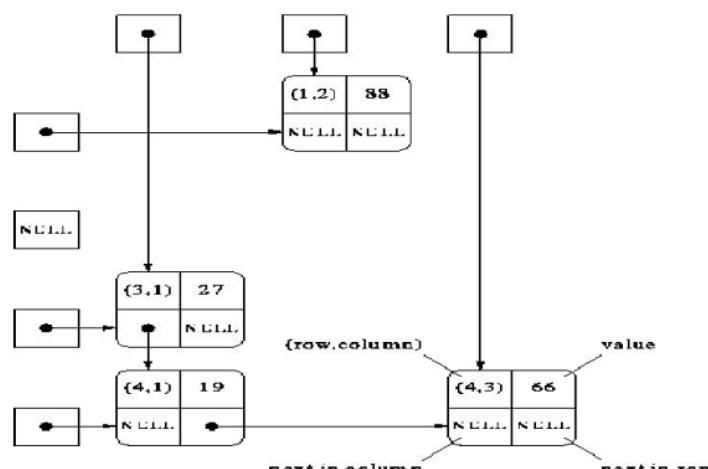
- 6 ■ typ souřadného systému a dimenze souřadnic a hodnot \mathbf{h}_i
- 7 ■ počet dělení na jednotlivých osách, tj. v případě E^2 , N_x a N_y
- 8 ■ minimální a maximální hodnoty, tj. x_{min} a x_{max}
- 9 ■ matici hodnot $\mathbf{A} = \{a_{ij}\}$
- 10 • *nepravidelná* – pozice vrcholů je určena nepřímo přes samostatné reindexační vektory pro
 11 souřadnice x , y , resp. z , obr.x.b. Nepravidelná strukturovaná data nám umožňují zjemnit
 12 velikost jednotlivých buněk v případě potřeby, např. pro přesnější výpočty.

13 To znamená, že pro datový set je nutné si pamatovat:

- 14 • typ souřadného systému a dimenze souřadnic a hodnot \mathbf{h}_i
- 15 • počet dělení na jednotlivých osách, tj. v případě E^2 , N_x a N_y
- 16 • hodnoty pro dělení na osách x a y , tj. hodnoty x_j a y_i ; indexy i a j odpovídají indexům prvků
 17 v matici \mathbf{A} .
- 18 • minimální a maximální hodnoty nejsou zapotřebí, tj. x_{min} a x_{max} nejsou zapotřebí.
- 19 • matici hodnot \mathbf{A}

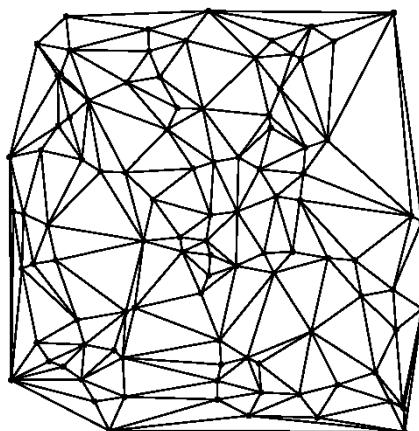
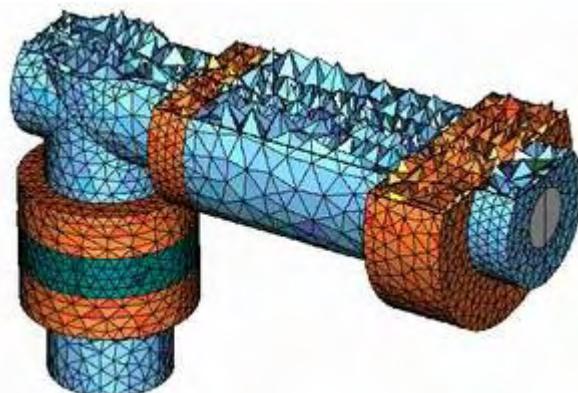
20 V případě použití nepravidelné struktury, která není jednoznačně určena použitým souřadným
 21 systémem (např. strukturovaná nepravidelná síť složená ze čtyřúhelníků), se ukládají matice X a Y
 22 obsahující souřadnice x_j a y_i odpovídající prvku a_{ij} matice \mathbf{A} . Obdobně se postupuje v některých
 23 případech strukturované trojúhelníkové, resp. tetrahedronové sítě, kdy je celá síť topologicky
 24 pravidelná a je znám jednoznačný vztah mezi indexem vrcholu a jeho souřadnicí.

25
 26 Ve specializovaných výpočetních a vizualizačních systémech se používají specializované datové
 27 struktury. Například v oblasti výpočetních systémů s konečnými elementy se používají k reprezentaci
 28 dat řídké matice („sparse matrix“).



Obr.20.21: Datová struktura pro řídké matice

1 Tato datová struktura je vhodná pro reprezentaci neuspořádaných roztroušených dat, pokud
 2 hodnoty x a y souřadnic jsou celočíselné. Datovou strukturu je možné rozšířit i pro aplikace v E^3 .
 3 Pokud je rozsah souřadnic velký pro malý počet výskytů hodnot, je možné s výhodou využít datovou
 4 strukturu založenou na hashovací funkci.
 5 Strukturovaná data mají obecně menší paměťovou náročnost, hůře reprezentují oblé tvary v případě
 6 pravoúhlých sítí. Naproti tomu nestrukturovaná data mají větší paměťovou náročnost, složitější
 7 algoritmy, ale daří se jimi lépe reprezentovat objekty s nerovinnými plochami a hranami.
 8 Někdy se mluví o „gridu“ a „grid generation“, zejména u fyzikálních výpočtů a vizualizací, např. u
 9 pružnosti, pevnosti, teplotních polí apod. Také se lze setkat s pojmy buňka („cell“).
 10 Uspořádaná data z dat neuspořádaných obvykle získáme triangulačními technikami, tj. triangulaci
 11 v E^2 nebo tetrahedronizací v E^3 . Nicméně výpočetní složitost pro vyšší dimenze a implementační
 12 komplikovanost je neúnosná. Navíc v případě dynamických dat, kdy se mění pozice jednotlivých
 13 bodů, tj. našich částeček, by se vždy musel proces triangulace opakovat, což by vedlo k neúnosným
 14 datovým a časovým nárokům. Proto se v současné době začínají upřednostňovat tzv. „meshless“
 15 techniky, tj. techniky, které nevyžadují uspořádání dat.

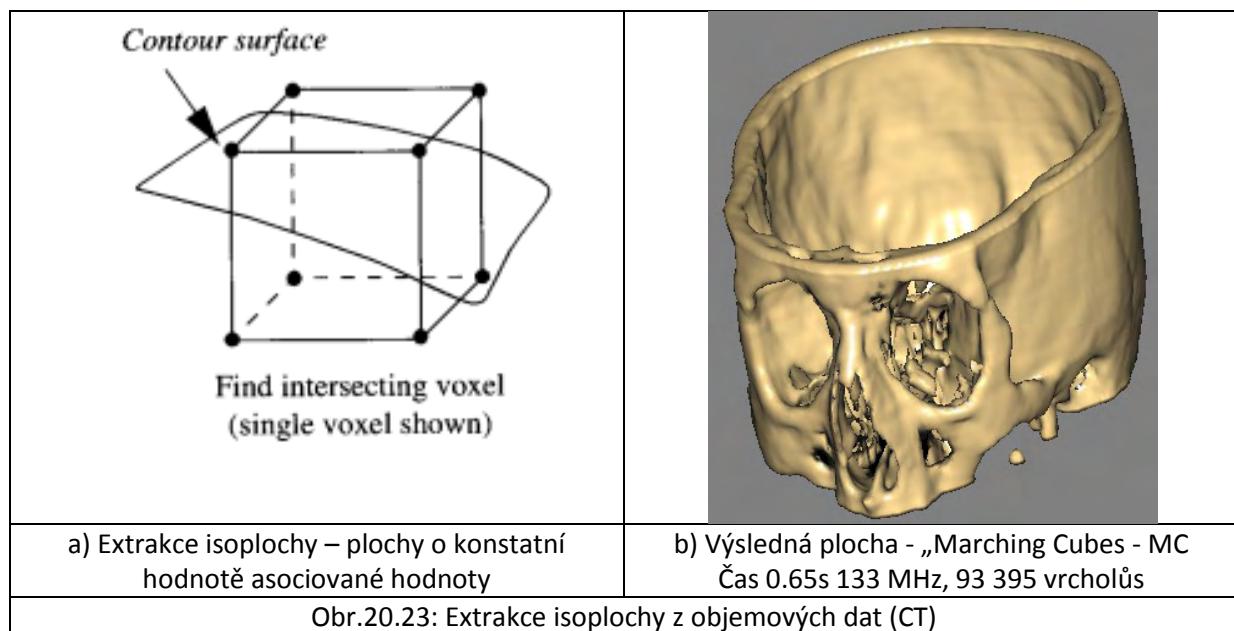
Výsledek triangulace v E^2 Výsledek triangulace v E^3

Obr.20.22: Ukázka výsledku triangulace a tetrahedronizace

16
 17 Pokud tedy data jsou uspořádaná, je jejich součástí i informace o struktuře dat.
 18 Z důvodů velkého objemu dat a složitosti manipulace s nimi se upřednostňují strukturovaná data.
 19
 20

21 20.2.4. Objemová (volumetrická) data

22 Velmi častou datovou strukturu jsou tzv. objemová data, která se používají zejména pro
 23 reprezentaci skalárních polí, např. CT (Computer Tomography), nebo MRI (Magnetic Resonance
 24 Image) dat. Jde vlastně o analogii rastrového obrazu, kdy je reprezentován objem ve tvaru kvádru,
 25 resp. krychle, a element se nazývá voxel (Volume element).

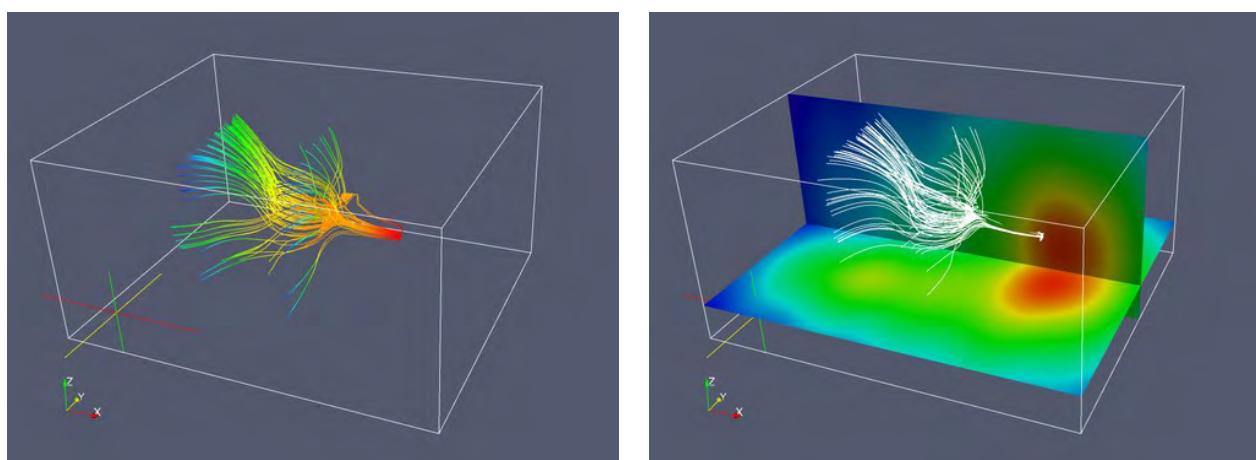


Obr.20.23: Extrakce isoplochy z objemových dat (CT)

1
 2 V převážné většině, zejména pak při zpracování CT a MRI dat, se používá duální reprezentace, kdy
 3 hodnoty jsou asociovány do vrcholů. Při hledání iso-plochy se pak hodnoty vrcholů ohodnotí, zda
 4 hodnota je vyšší nebo nižší než hodnota hledané iso-plochy.
 5 Následně se určí průsečíky iso-plochy s hranou krychle a určí se, jak iso-plocha uvnitř krychle vypadá.
 6 Některé techniky krychli „rozbijí“ na čtyřstěny a pak se počítají průsečíky hran čtyřstěnů s iso-
 7 plochou. Tento přístup má výhodu především v tom, že vznikají buď trojúhelník, nebo neplanární
 8 čtyřúhelník, který se rozdělí na dva trojúhelníky.
 9

10 **20.2.5. Vizualizace dynamických dat**

11 Zvláštní oblastí je vizualizace dynamických dat, kdy nejen asociované hodnoty, ale i geometrický tvar
 12 se v čase mění. Typickou ukázkou je pak např. vizualizace teploty a proudění směsi ve válci
 13 spalovacího motoru, kdy se v čase mění i geometrie, neboť píst motoru se pohybuje tam a zpět.
 14 Obr.20.24.a znázorňuje vizualizaci proudění, Obr.20.24.b pak ještě zobrazuje teplotní, nebo tlakové
 15 poměry v různých řezech.
 16



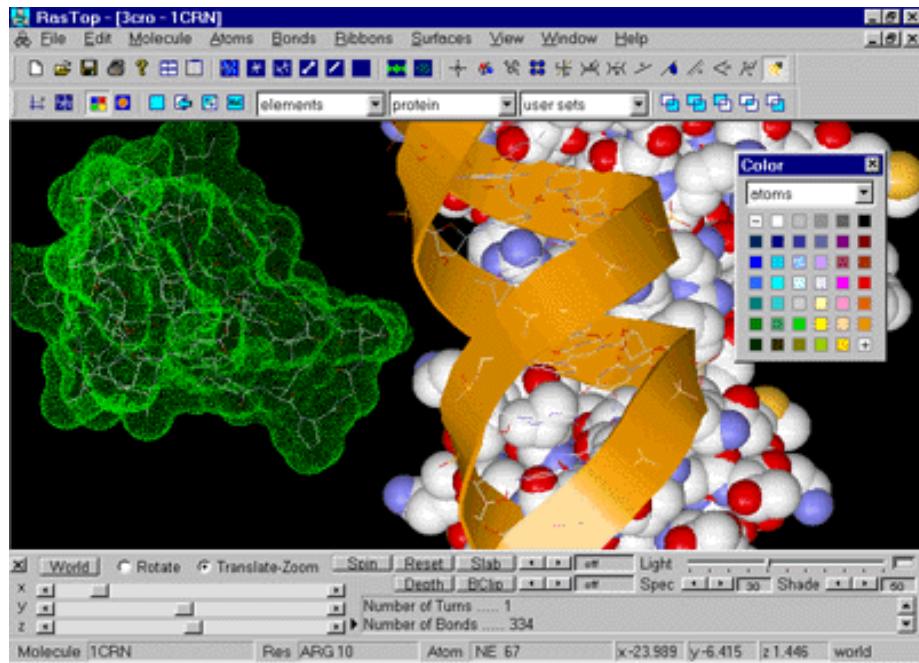
a) Vizualizace proudění

b) Proudění s barveným podbarvením tlaku

Obr.20.24: Ukázka vizualizace proudění

1 Jistě zajímavou oblastí je vizualizace v oblasti výpočetní chemie, např. struktur chemických látek,
2 proteinových struktur, genomu atd.

3



Obr.20.25: Ukázka vizualizace struktury proteinu

4

5

6

7

8

1 **21. Geometrická algebra - základy**

2

3

4 Čtenář si poslechne záznam odborné přednášky:

- 5 • Lengyel,E.: A Big Mathematical Picture for Computer Graphics, WSCG 2012 (65 min.)
6 ([CLICK off-line](#)) \Video\Lengyel.mp4

7

8

9

1 Doporučená literatura

- 2 • Foley,J.D., van Dam, A., Feiner,S.K., Huges,J.F.: Computer Graphics – Principles and Practice,
3 Addison Wesley, 1997
- 4 • Hill Jr.,F.S.: Computer Graphics Using OpenGL, Prentice Hall, 1990
- 5 • Theoharis,T., Papaioannou,G., Platis,N., Patrikalakis,N.M.: Graphics and Visualization –
6 Principles and Algorithms, A.K.Peters, 2008
- 7 • Shirley,P.: Fundamentals of Computer Graphics, A.K.Peters, 2005
- 8 • Blinn,J.: Jim Blinn's Corner: A Trip Down the Graphics Pipeline, 1996
- 9 • Solomon,D.: The Computer Graphics Manual, Vol. I a II, Springer, 2011
- 10 • Špička,I., Frischer,R.: Počítačová geometrie a grafika – Teoretické základy, VŠB Ostrava, 2012
- 11 • Sojka,E., Němec,M., Fabián,T.: Matematické základy počítačové grafiky, VŠB Ostrava, 2011
- 12 • Kolcun,A.: Počítačová grafika – Algoritmy a principy, Ostravská univerzita, 2009
- 13 • Kolcun,A.: Základy počítačovej grafiky, Ostravská univerzita, 2003
- 14 • Sobotka,B. :Počítačová grafika, Košice, 2000
- 15 • Drs,L., Všetečka,J.: Objektivem počítače - geometrie speciálních fotografických technik, SNTL
16 1981
- 17 • Žára,J., Beneš,B., Sochor,J., Felkel,P.: Moderní počítačová grafika, Computer Press, 2004
- 18 • Watt,A.: 3D Computer Graphics, Addison Wesley, 2000
- 19 • Shirley,P., Morley,R.K.: Realistic Ray Tracing, A.K.Peters, 2003
- 20 • Angel,E.: Interactive Computer Graphics, Addison Wesley, 2000
- 21 • McReynolds,T., Blythe,D.: Advanced Graphics Programming Using OpenGL, MK, 2005
- 22 • Watt,A., Polycarpo,F.: 3D Games: Real Time Rendering and Software Technology, Addison
23 Wesley, 2001
- 24 • Agoston,M.K.: Computer Graphics and Geometric Modeling, Springer 2005
- 25 • Agoston,M.K.: Computer Graphics and Geometric Modeling - Mathematics, Springer 2005
- 26 • Agoston,M.K.: Computer Graphics and Geometric Modeling – Implementation and
27 Algorithms, Springer 2005
- 28 • Schneider,P.J., Eberly,D.E: Geometric Tools for Computer GraphicsElsevier, 2003
- 29 • Ferguson,S., McMenemy,K.: A Hitchhiker's Guide to Virtual Reality, A.K.Peters, 2007
- 30 • Vince,J.: Mathematics for Computer Graphics, Springer, 2006
- 31 • Vince,J.: Matrix Transforms for Computer Games and Animation, Springer 2012
- 32 • Solomon,D.: Transformations and Projections in Computer Graphics, Springer, 2006
- 33 • Solomon,D.: Computer Graphics and Geometric Modeling, Springer, 1999
- 34 • Egerton,P.A., Hall,W.S.: Computer Graphics – Mathematical First Steps, Prentice Hall, 1998
- 35 • Ghali,S.: Introduction to Geometric Computing, Springer, 2008
- 36 • Klawonn,F.: Introduction to Computer Graphics – Using Java 2D and šD, Springer, 2012
- 37 • Chen,J.X., Wegman,E.J.: Foundations of 3D Graphics Programming – Using JOGL and JAVA 3D,
38 Springer, 2006
- 39 • Ammeraal,L.: Computer Graphics for JAVA programmers, John Wiley, 1998
- 40 • Hearn,D., Baker,M.P.: Computer Graphics with OpenGL, Prentice Hall, 2004
- 41 • Boehm,W., Prautzsch,H.: Geometric Concepts for geometric Design, A.K.Peters, 1994
- 42 • Parke,F.I., Waters,K.: Computer Facial Animation, A.K.Peters, 1996
- 43 • Pozrikidis,C.: Introduction to C++ Programming and Graphics, Springer, 2007

- Middleton,L., Sivaswamy,J.: Hexagonal Image Processing – A practical Approach, Springer, 2005
 - Ericson,Ch.: Real-Tiem Collision Detection, Morgan Kaufmann Publ., USA, ISBN 1-55860-732-3, 2004
 - Lengyel,E.: Mathematics for 3D Game Programming and Computer Graphics, Course Technology, ISBN 978-1-4354-5886-4, 2013
 - Pharr,M., Humphreys,G.: Physically Based Rendering – From Thoery to Implementation, Morgan Kaufmann Publ., ISBN 978-0-12-375079-2, 2010

1 22. Příloha A – Vektory a geometrická interpretace

2 Pojem vektor se používá v mnohých kontextech. Při realizaci algoritmů musíme rozlišovat, co pod
3 pojmem vektor vlastně rozumíme, zejména pak musíme rozlišovat pojmenování vektor:

- 4 • *v matematickém smyslu*, kdy jde o element vektorového prostoru s jasně definovanými
5 vlastnostmi. *Vektor udává směr a má určitou délku a není vázán na počátek souřadného*
6 *systému*
- 7 • *ve smyslu datových struktur*, kdy jde vlastně o reprezentaci jednorozměrné matice, ve které
8 jsou uložena data reprezentující nějakou geometrickou, či jinou hodnotu.

9 Takže např. souřadnice bodu jsou uloženy v datové struktuře typu vektor, ale evidentně jsou svázány
10 s daným souřadným systémem. Vektory, jako datová struktura, tj. jednorozměrné matice, jsou
11 sloupcové orientovány. Pozor, některé publikace používají řádkovou notaci.

12 **Uvedeme jednoduchý příklad pro ilustraci součtu.**

13 Mějme body P , Q a R vektory u , v a w

14

15 Pak součet vektorů je dán vztahem:

$$w = u + v$$

16 Lze také pozici bodu P posunout o vektor u a pak souřadnice nového bodu R jsou určeny:

$$R = P + u$$

17 Nicméně *NELZE „sčítat“ reprezentaci dvou bodů*:

$$R = P + Q$$

18 neboť toto nemá obecně geometrickou interpretaci.

19

20 Lze proto jen doporučit používat dva různé datové typy, a to:

- 21 • *vektor (vector)* – tedy objekt, který není vázán na počátek souřadného systému
- 22 • *bod (point)* – tedy objekt, který je vázán na počátek souřadného systému

23 které se však mapují na stejnou fyzickou datovou strukturu jednorozměrného pole, tj. na lineární
24 paměťový prostor - vektor.

25

26

27

1 **22.1. Základní operace s vektory**

	Zákon
$\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$	asociativní
$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$	komutativní
$a(b\mathbf{u}) = (ab)\mathbf{u}$	asociativní
$(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$	distribuční
$a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$	distribuční
$ \mathbf{u} = \sqrt{\sum u_i^2}$	délka vektoru
$ a\mathbf{u} = a \mathbf{u} $	asociativní

2

3 **Normalizace vektoru**

$\mathbf{v} = \frac{\mathbf{u}}{ \mathbf{u} }$	pak vektor \mathbf{v} je normalizovaný, tj. $ \mathbf{v} = 1$
--	--

4

5 **Skalární součin (dot product)**

6 Výsledkem skalárního součinu je skalár, tj. hodnota. Skalární součin je definován takto:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^n u_i v_i$$

7 **Vlastnosti skalárního součinu**

$\mathbf{u} \cdot \mathbf{u} = \mathbf{u} ^2$	$(a\mathbf{u}) \cdot (b\mathbf{v}) = (ab)(\mathbf{u} \cdot \mathbf{v})$	$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
$\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \cdot \mathbf{v}) + (\mathbf{u} \cdot \mathbf{w})$		$(\mathbf{u} + \mathbf{v}) \cdot \mathbf{w} = (\mathbf{u} \cdot \mathbf{w}) + (\mathbf{v} \cdot \mathbf{w})$

8 V mnoha aplikacích lze s výhodou využít formule

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos \varphi$$

9 kde φ je úhel sevřený vektory \mathbf{u} a \mathbf{v} . Pokud $|\mathbf{u}| = 1$, pak $\mathbf{u} \cdot \mathbf{v}$ je délka vektoru \mathbf{v} po promítnutí na vektor \mathbf{u} , a tedy:

$$\mathbf{v}' = \mathbf{v} \frac{\mathbf{u}}{|\mathbf{u}|}$$

11 Vektor \mathbf{v}' je tedy promítnutým vektorem \mathbf{v} na vektor \mathbf{u} .

12

1 **Operátor Perp**

2 Operátor **perp** je aplikovatelný pouze v případě E^2 . Definuje kolmý vektor k danému vektoru ve
3 směru proti hodinovým ručičkám, tj. *counterclockwise (ccw)* a je definován takto:

$$\mathbf{v}^\perp = (v_1, v_2)^\perp = (-v_2, v_1)$$

4 Pro **perp** operátor pak platí:

$\mathbf{v}^\perp \cdot \mathbf{v} = 0$	$ \mathbf{v}^\perp = \mathbf{v} $	$(a\mathbf{u} + b\mathbf{v})^\perp = a\mathbf{u}^\perp + b\mathbf{v}^\perp$	$(\mathbf{v}^\perp)^\perp = -\mathbf{v}$
---	-------------------------------------	---	--

5

6 **Součin Perp** (Perp product)

7 Výsledkem operátoru **perp** je skalární hodnota a je definován takto:

$$\mathbf{u} \perp \mathbf{v} = \mathbf{u}^\perp \cdot \mathbf{v} = u_1 v_2 - u_2 v_1 =$$

$$\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} = |\mathbf{u}| |\mathbf{v}| \sin \varphi$$

8 kde φ je úhel sevřený vektory \mathbf{u} a \mathbf{v} , což vyjadřuje orientovanou plochu kosodélníka s hranami \mathbf{u} a \mathbf{v} .
9 Je vhodné si zde všimnout podobnost s vektorovým součinem.

10 Pro **perp** operátor platí:

$\mathbf{u} \perp \mathbf{u} = 0$	$(a\mathbf{u}) \perp (b\mathbf{v}) = (ab)(\mathbf{u} \perp \mathbf{v})$	$\mathbf{u} \perp \mathbf{v} = \mathbf{v} \perp \mathbf{u}$
$\mathbf{u} \perp (\mathbf{v} + \mathbf{w}) = \mathbf{u} \perp \mathbf{v} + \mathbf{u} \perp \mathbf{w}$		$(\mathbf{u} \perp \mathbf{v})^2 + (\mathbf{u} \cdot \mathbf{v})^2 = \mathbf{u} ^2 \mathbf{v} ^2$

11

12 **Vektorový součin** (cross product nebo outer product)

13 Výsledkem vektorového součinu je vektor a je definován takto:

$$\mathbf{v} \times \mathbf{w} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

14 kde bázové vektory jsou definovány takto: $\mathbf{i} = (1,0,0)$, $\mathbf{j} = (0,1,0)$, $\mathbf{k} = (0,0,1)$

15 Dále platí:

$\mathbf{u} \times \mathbf{u} = \mathbf{0} = (0,0,0)$	$(a\mathbf{u}) \times (b\mathbf{v}) = (ab)(\mathbf{u} \times \mathbf{v})$	$\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$
$\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = \mathbf{u} \times \mathbf{v} + \mathbf{u} \times \mathbf{w}$	$(\mathbf{u} \times \mathbf{v}) \cdot \mathbf{u} = (\mathbf{u} \times \mathbf{v}) \cdot \mathbf{v} = \mathbf{0}$	$(\mathbf{u} \times \mathbf{v})^2 + (\mathbf{u} \cdot \mathbf{v})^2 = \mathbf{u} ^2 \mathbf{v} ^2$

16 Dále platí následující vztahy:

$(\mathbf{u} \times \mathbf{v}) \times \mathbf{w} = (\mathbf{u} \cdot \mathbf{w})\mathbf{v} - (\mathbf{v} \cdot \mathbf{w})\mathbf{u}$	$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = (\mathbf{u} \cdot \mathbf{w})\mathbf{v} - (\mathbf{v} \cdot \mathbf{w})\mathbf{u}$
$\mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) \cdot \mathbf{w}$	$(\mathbf{u} \times \mathbf{v}) \cdot (\mathbf{w} \times \mathbf{x}) = (\mathbf{u} \cdot \mathbf{w})(\mathbf{v} \cdot \mathbf{x}) - (\mathbf{v} \cdot \mathbf{w})(\mathbf{u} \cdot \mathbf{x})$
$ \mathbf{u} \times \mathbf{v} = \mathbf{u} \mathbf{v} \sin \varphi$	

17 přičemž vektor $\mathbf{u} \times \mathbf{v}$ je kolmý na vektory \mathbf{u} a \mathbf{v} .

1 **Smíšený vektorový součin (triple product)**

2 Výsledkem smíšeného součinu je skalární hodnota a je definován takto:

$$[\mathbf{u} \mathbf{v} \mathbf{w}] = \mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}) = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

3 Smíšený součin vlastně vyjadřuje objem. Objem čtyřstěnu (tetrahedronu), jehož hrany jsou dány vektory \mathbf{u} , \mathbf{v} a \mathbf{w} je pak určen takto:

$$\text{Objem}_{\text{tetra}} = \frac{1}{6} [\mathbf{u} \mathbf{v} \mathbf{w}]$$

5 **Tensorový součin (tensor product)**6 Kromě výše uvedených operátorů se ještě používá *tenzorový součin*, který je definován takto:

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_x v_x & u_x v_y & u_x v_z \\ u_y v_x & u_y v_y & u_y v_z \\ u_z v_x & u_z v_y & u_z v_z \end{bmatrix}$$

7 Jde tedy vlastně o operaci sloupec*řádek, kde výsledek je matice $\mathbf{Q} = \mathbf{u} \mathbf{v}^T$.

8 Pro tenzorový součin pak platí vztah:

$$(\mathbf{u} \otimes \mathbf{v}) \hat{\mathbf{n}} = (\mathbf{v} \cdot \hat{\mathbf{n}}) \mathbf{u} \quad (22.1)$$

9 Dále platí:

$$\mathbf{u} \otimes \mathbf{v} = (\mathbf{v} \otimes \mathbf{u})^T \quad (22.2)$$

10 a

$$\mathbf{u} \otimes (a\mathbf{v} + b\mathbf{w}) = a(\mathbf{u} \otimes \mathbf{v}) + b(\mathbf{u} \otimes \mathbf{w}) \quad (22.3)$$

11

12 Lze dokázat, že pro libovolné vektory \mathbf{c} , \mathbf{d} a \mathbf{e} platí:

$$(\mathbf{c} \cdot \mathbf{d}) \mathbf{e} = (\mathbf{e} \otimes \mathbf{c}) \mathbf{d} = (\mathbf{e} \otimes \mathbf{d}) \mathbf{c} \quad (22.4)$$

13 Dále pak:

$$(\mathbf{a} \cdot \mathbf{b})(\mathbf{c} \cdot \mathbf{d}) = \mathbf{a}(\mathbf{b} \otimes \mathbf{c}) \mathbf{d}$$

14

15

16

1 **22.2. Vzdálenost bodu od přímky v E^2 a v E^3**

2 Je dána přímka L body P_0 a P_1 . Úkolem je najít vzdálenost $d(L, P)$ bodu P od přímky L , a to jak v E^2 ,
3 tak i v E^3 .

4

5 **Vzdálenost bodu od přímky v E^2**

6 V případě E^2 můžeme určit normálu přímky \mathbf{n} takto:

$$\mathbf{n} = (P_1 - P_0)^\perp$$

7 Pak vzdálenost $d(L, P)$ je vlastně určena jako průmět vektoru $P - P_0$ na normálový vektor \mathbf{n} , a tedy:

$$d(L, P) = |P - P_0| \cos \varphi = \frac{\mathbf{n}}{|\mathbf{n}|} (P - P_0) = \frac{\Delta y(x - x_0) - \Delta x(y - y_0)}{\sqrt{\Delta x^2 + \Delta y^2}}$$

8 Poznamenejme, že $d(L, P)$ je orientovaná vzdálenost bodu od dané přímky.

9

10 **Vzdálenost bodu od přímky v E^3**

11 V případě máme dva vektory, a to:

$\mathbf{u} = (P_1 - P_0)$	$\mathbf{v} = (P - P_0)$
----------------------------	--------------------------

12 Hodnota $|\mathbf{u} \times \mathbf{v}|$ je plochou kosodélníka, jehož hrany jsou vektory \mathbf{u} a \mathbf{v} . Je tedy zřejmé, že pokud
13 hodnotu plochy vydělíme délkou vektoru \mathbf{u} , pak dostaneme výšku, což je vlastně vzdálenost bodu P
14 od přímky L , a tedy:

$$d(L, P) = \frac{|\mathbf{u} \times \mathbf{v}|}{|\mathbf{u}|} = \frac{|(P_1 - P_0) \times (P - P_0)|}{|P_1 - P_0|}$$

15 Je zřejmé, že jde nyní o neorientovanou vzdálenost, neboť v E^3 není orientace obecně definována.

16

17 **Vzdálenost bodu od roviny**

18 Čtenář si odvodí analogické vztahy sám.

1 **22.3. Další operátory**

2 Kromě běžných operátorů s vektory uvedeme ještě operátor \sqcap , který je definován takto:

$$\mathbf{a} \sqcap \mathbf{b} = \mathbf{a}^T \mathbf{b} \mathbf{I} - \mathbf{a} \mathbf{b}^T = \mathbf{a} \cdot \mathbf{b} \mathbf{I} - \mathbf{a} \otimes \mathbf{b} =$$

$$\begin{bmatrix} a_y b_y + a_z b_z & -a_x b_y & -a_x b_z \\ -a_y b_x & a_x b_x + a_z b_z & -a_y b_z \\ -a_z b_x & -a_z b_y & a_x b_x + a_y b_y \end{bmatrix}$$

3 kde \mathbf{I} je identická matice.

4 Tento operátor se používá např. pro výpočet inerciálního tenzoru množiny daných bodů. Je-li dána
5 množina bodů $\{\mathbf{x}_i\}_{i=1}^N$, pak matice inerciálního tenzoru \mathbf{IT} je určena vztahem:

$$\mathbf{IT} = \sum_{i=1}^N (\mathbf{x}_i - \mathbf{x}_T) \sqcap (\mathbf{x}_i - \mathbf{x}_T)$$

6 kde \mathbf{x}_T je těžiště bodů dané množiny.

7 Podrobně viz:

- 8 • Fusello,A., Murino,V.: Augmented Scene Modeling and Visualization by Optical and Acoustic
9 Sensor Integration, IEEE Trans.on Visualization and Computer Graphics, Vol.10, No.6, 2004

10

11

12 Vektory a jejich geometrická interpretace je předpokladanou znalostí a lze doporučit např.:

- 13 • Sunday,D.: Basic Linear Algebra, <http://softSurfer.com> ([CLICK off-line](#))

14

23. Příloha B – Matice a operace s maticemi

2 [Ref.: http://en.wikipedia.org/wiki/Matrix_calculus]

3 **Použitá notace:**

4 a – skalár, tj. skalární hodnota nebo funkce

5 \mathbf{a} – sloupcové vektor, tj. $\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = [a_1, \dots, a_n]^T$

6 \mathbf{A} – obdélníková matice $n \times m$, tj. $\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$

7 V textu se používá běžná notace pro zápis vektorových a maticových operací, nicméně je nutné
8 upozornit, že v mnohých publikacích se vedle sloupcové notace také používá pro vektory řádková
9 notace, což může někdy vést k podstatným chybám při realizaci algoritmů. Zejména je nutné věnovat
10 pozornost transpozicím vektorů a matic.

23.1. Základní operace s maticemi

12 Níže uvedené operace předpokládají, že rozměry matic danou operaci připouštějí. Na vektory lze
13 pohlížet jako na jednorozměrné maticy

Transpozice	$\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = [a_1, \dots, a_n]^T$
Součet matic	$\mathbf{C} = \mathbf{A} + \mathbf{B}$ $c_{ij} = a_{ij} + b_{ij}$
Násobení matic	$\mathbf{C} = \mathbf{AB}$ $c_{ij} = \sum_{k=1}^p a_{ik} b_{kj}$
Transpozice součinu matic	$\mathbf{C} = \mathbf{AB}$ $\mathbf{C}^T = \mathbf{B}^T \mathbf{A}^T$
Inverze součinu matic	$\mathbf{C} = \mathbf{AB}$ $\mathbf{C}^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$
Blokové maticy	$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}$
Sčítání blokových matic	$\mathbf{A} + \mathbf{T} = \begin{bmatrix} \mathbf{B} + \mathbf{P} & \mathbf{C} + \mathbf{Q} \\ \mathbf{D} + \mathbf{R} & \mathbf{E} + \mathbf{S} \end{bmatrix}$
Násobení blokových matic	$\mathbf{AT} = \begin{bmatrix} \mathbf{BP} + \mathbf{CR} & \mathbf{BQ} + \mathbf{CS} \\ \mathbf{DP} + \mathbf{ER} & \mathbf{DQ} + \mathbf{ES} \end{bmatrix}$
Inverze blokových matic	$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B})^{-1} \\ -(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} & (\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B})^{-1} \end{bmatrix}$
Inverze blokových matic*	$\begin{bmatrix} \mathbf{0} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{R} \end{bmatrix}^{-1} = \begin{bmatrix} -(\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F})^{-1} & (\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{R}^{-1} \\ \mathbf{R}^{-1} \mathbf{F} (\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F})^{-1} & \mathbf{R}^{-1} - \mathbf{R}^{-1} \mathbf{F} (\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}^T \mathbf{R}^{-1} \end{bmatrix}$ $(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{C}^{-1} + \mathbf{D} \mathbf{A}^{-1} \mathbf{B}) \mathbf{D} \mathbf{A}^{-1}$ $(\mathbf{M}\mathbf{a}) \times (\mathbf{M}\mathbf{b}) = (\det \mathbf{M}) \mathbf{M}^{-T} (\mathbf{a} \times \mathbf{b})$

14 * viz <http://math.stackexchange.com/questions/353073/inverse-of-a-partitioned-matrix>

15 Tato formule umožňuje inverzi matice s blokovou NULOVOU submaticí za předpokladu, že
16 matice $\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F}$ je invertovatelná.

17

18 Další operace s blokovými maticemi jsou dostupné v odborné literatuře.

19

20 Kromě jednoduchých operací, jako je násobení a sčítání, se ještě používají složitější operace, např.
21 derivace maticových a vektorových formulí.

1 23.2. Determinanty

2 Operace s determinanty jsou všeobecně známé, uvedeme proto jen formule, které nejsou běžně
3 dostupné, avšak mohou být velmi užitečné

$$|A + BC| = |A| \cdot |I + CA^{-1}B|$$

4

5

6

7

24. Příloha C – Derivace vektorových a maticových funkcí

Ve fyzikálních aplikacích vektory a matice představují v mnoha případech vlastně vektory nebo matice funkcí, tj. prvek vektoru nebo matice není skalární hodnota, ale je funkcí jedné či více proměnných. Protože fyzikální problémy jsou velmi často popsány v diferenciální nebo integrální formě, je vhodné uvést základní pravidla pro diferenciální operace s vektory a maticemi.

Je nutné především rozlišit dva základní případy, a to:

$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \vdots \\ \frac{\partial f_n}{\partial x} \end{bmatrix} = \left[\frac{\partial f_1}{\partial x}, \dots, \frac{\partial f_n}{\partial x} \right]^T$	$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right]$
Derivace vektorové funkce skalární proměnnou. Výsledkem je sloupcový vektor gradient funkce je tedy řádkový vektor	Derivace skalární funkce vektorovou proměnnou. Výsledkem je řádkový vektor (gradient funkce je tedy řádkový vektor)

Kromě výše uvedených operací jsou důležité matice známé pod názvy Jakobián (Jacobi matrix) a Hessián (Hessian Matrix), které se zejména používají při řešení optimalizačních problémů, kdy funkční hodnotu vektorové funkce \mathbf{f} a vektorového argumentu \mathbf{x} nahradíme approximací Taylorového rozvoje

$$\mathbf{y} = \mathbf{f}(\mathbf{x} + \Delta \mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta \mathbf{x} + \frac{1}{2}\Delta \mathbf{x}^T \mathbf{H}(\mathbf{x})\Delta \mathbf{x}$$

POZOR $\mathbf{H}(\mathbf{x})$ je obecně „matice se třemi rozměry“

kde: $\mathbf{J}(\mathbf{x})$ je Jakobián a $\mathbf{H}(\mathbf{x})$ je Hessián.

12

V reálných případech se doporučuje použít metody založené na kvazi-Newtonově algoritmu, např. BFGS algoritmu (Broyden-Fletcher-Goldfarb-Shannon algoritmus). Pro úplnost uvedeme tyto matice

$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$	$\mathbf{H}(\mathbf{x}) = \mathbf{J}(\nabla \mathbf{f}(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$
Jakobiho matice	Hessián pro případ skalární funkce f

15

24.1. Základní pravidla pro derivaci s vektory a maticemi, resp. vektorových a maticových funkcí.

Čím derivujeme		Co derivujeme		
		Skalár	Vektor	Matice
Skalár	$\frac{\partial a}{\partial x}$	$\frac{\partial \mathbf{a}}{\partial x}$	$\frac{\partial \mathbf{A}}{\partial x}$	
Vektor	$\frac{\partial \mathbf{a}}{\partial x}$	$\frac{\partial \mathbf{a}}{\partial x}$	--	
Matice	$\frac{\partial a}{\partial X}$	--	--	

18

1 Derivace skalárni proměnnou

$\frac{\partial a}{\partial x} = q$	Derivace skalárni hodnoty: Výsledkem je skalárni hodnota q
$\frac{\partial \mathbf{a}}{\partial x} = \begin{bmatrix} \frac{\partial a_1}{\partial x} \\ \vdots \\ \frac{\partial a_n}{\partial x} \end{bmatrix} = \left[\frac{\partial a_1}{\partial x}, \dots, \frac{\partial a_n}{\partial x} \right]^T$ $\mathbf{q} = [q_1, \dots, q_n]^T$	Derivace vektoru: Výsledkem je vektor \mathbf{q} $\mathbf{a} = [a_1, \dots, a_n]^T$
$\frac{\partial \mathbf{A}}{\partial x} = \frac{\partial}{\partial x} \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$	Derivace matice: Výsledkem je matice \mathbf{Q} $\mathbf{Q} = \begin{bmatrix} \frac{\partial a_{11}}{\partial x} & \cdots & \frac{\partial a_{1m}}{\partial x} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n1}}{\partial x} & \cdots & \frac{\partial a_{nm}}{\partial x} \end{bmatrix}$

2

3 Derivace vektorovou proměnnou

$\frac{\partial a}{\partial \mathbf{x}} = \left[\frac{\partial a}{\partial x_1}, \dots, \frac{\partial a}{\partial x_m} \right]$ $\mathbf{q}^T = [q_1, \dots, q_m]$	Derivace skalárni hodnoty: Výsledkem je řádkový vektor \mathbf{q}^T $\mathbf{x} = [x_1, \dots, x_m]$
$\frac{\partial \mathbf{a}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \cdots & \frac{\partial a_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial x_1} & \cdots & \frac{\partial a_n}{\partial x_m} \end{bmatrix}$	Derivace vektoru: Výsledkem je matice $\mathbf{Q} = \begin{bmatrix} q_{11} & \cdots & q_{1m} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nm} \end{bmatrix}$ $\mathbf{a} = [a_1, \dots, a_n]^T$ $\mathbf{x} = [x_1, \dots, x_m]^T$
$\frac{\partial \mathbf{A}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$	Derivace vektoru: Výsledkem je $3D$ matice \mathbf{Q} $\mathbf{Q} = \begin{bmatrix} \frac{\partial a_{11}}{\partial x_1} & \cdots & \frac{\partial a_{1m}}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n1}}{\partial x_1} & \cdots & \frac{\partial a_{nm}}{\partial x_1} \end{bmatrix} \cdots \begin{bmatrix} \frac{\partial a_{11}}{\partial x_n} & \cdots & \frac{\partial a_{1m}}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n1}}{\partial x_n} & \cdots & \frac{\partial a_{nm}}{\partial x_n} \end{bmatrix} \cdots$

4 Gradient je tedy řádkový vektor.

5

6 Příklad:

7 Uvažme směrovou derivaci skalárni funkce více proměnných $f(\mathbf{x})$ v prostoru \mathbf{x} ve směru vektoru \mathbf{u} ,
8 která je definována

$$\nabla_u f(\mathbf{x}) = \nabla f(\mathbf{x}) \mathbf{u} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \mathbf{u}$$

9 Derivace maticovou proměnnou

$\frac{\partial a}{\partial \mathbf{X}} = \mathbf{Q} = \begin{bmatrix} \frac{\partial a}{\partial x_{11}} & \cdots & \frac{\partial a}{\partial x_{m1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial a}{\partial x_{1n}} & \cdots & \frac{\partial a}{\partial x_{mn}} \end{bmatrix}$	Derivace skalárni a hodnoty maticí \mathbf{X} - výsledkem je matice \mathbf{Q}
--	--

10

1 24.2. Základní identity

2 Pro odvozování formulí je vhodné připomenout základní pravidla pro derivace:

$\frac{\partial \mathbf{a}}{\partial \mathbf{x}} = \mathbf{0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix}$	$\frac{\partial \mathbf{x}}{\partial \mathbf{x}} = \mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$
$\frac{\partial(\mathbf{u} + \mathbf{v})}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$	$\begin{aligned} \frac{\partial(\mathbf{u} \cdot \mathbf{v})}{\partial \mathbf{x}} &= \frac{\partial(\mathbf{u}^T \mathbf{v})}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right)^T \mathbf{v} + \mathbf{u}^T \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \\ &= \mathbf{u}^T \frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \mathbf{v}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \end{aligned}$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{A} \mathbf{x} = \mathbf{A}$	$\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{A} = \mathbf{A}^T$
$\frac{\partial(\mathbf{u} \times \mathbf{v})}{\partial \mathbf{x}} = \mathbf{u} \times \frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \times \mathbf{v}$	$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} = \frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$
$\frac{\partial}{\partial \mathbf{x}} a \mathbf{u}(\mathbf{x}) = a \frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_n}{\partial x_1} & \cdots & \frac{\partial u_n}{\partial x_m} \end{bmatrix}$	$\begin{aligned} \frac{\partial(\mathbf{u} \cdot \mathbf{Av})}{\partial \mathbf{x}} &= \frac{\partial(\mathbf{u}^T \mathbf{Av})}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right)^T \mathbf{Av} + \mathbf{u}^T \mathbf{A}^T \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \\ &= \mathbf{u}^T \mathbf{A} \frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \mathbf{v}^T \mathbf{A}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \end{aligned}$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{A} \mathbf{u}(\mathbf{x}) = \mathbf{A} \frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}$	$\frac{\partial f(g(\mathbf{u}))}{\partial \mathbf{x}} = \frac{\partial f(g)}{\partial g} \frac{\partial g(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$
$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} (\mathbf{Ax} + \mathbf{b})^T \mathbf{C} (\mathbf{Dx} + \mathbf{e}) \\ &= (\mathbf{Dx} + \mathbf{e})^T \mathbf{C}^T \mathbf{A} \\ &\quad + (\mathbf{Ax} + \mathbf{b})^T \mathbf{C} \mathbf{D} \end{aligned}$	$\frac{\partial(\mathbf{a} \cdot \mathbf{x})}{\partial \mathbf{x}} = \frac{\partial(\mathbf{a}^T \mathbf{x})}{\partial \mathbf{x}} = \mathbf{a}^T$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{b}^T \mathbf{Ax} = \mathbf{b}^T \mathbf{A}$	$\frac{\partial}{\partial \mathbf{x}} \ \mathbf{x} - \mathbf{a}\ = \frac{(\mathbf{x} - \mathbf{a})^T}{\ \mathbf{x} - \mathbf{a}\ }$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{Ax} = \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T)$	$\frac{\partial^2}{\partial \mathbf{x}^2} \mathbf{x}^T \mathbf{Ax} = \mathbf{A} + \mathbf{A}^T$
$\frac{\partial^2}{\partial \mathbf{x}^2} \mathbf{x}^T \mathbf{x} = 2 \mathbf{x}^T$	$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} (\mathbf{a}^T \mathbf{x}^T \mathbf{x} \mathbf{b}) &= \mathbf{x}^T (\mathbf{ab}^T + \mathbf{ba}^T) \\ \mathbf{x}^T \mathbf{x} &\text{ je symetrická matice } \mathbf{Q}! \end{aligned}$
$\frac{d}{d \mathbf{A}} = \begin{bmatrix} \frac{\partial}{\partial a_{11}} & \cdots & \frac{\partial}{\partial a_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial a_{n1}} & \cdots & \frac{\partial}{\partial a_{nm}} \end{bmatrix}$	$\frac{d}{d \mathbf{A}} (\mathbf{u}^T \mathbf{Av}) = \mathbf{uv}^T$

3
4

25. Příloha D – Diferenciální a integrální operátory

Při popisu fyzikálních, zejména dynamických, objektů se používá v mnoha případech diferenciální forma. U dynamických systémů derivace podle času vyjadřující časovou změnu.

- **rychlosť** (speed) - derivace polohy podle času
- **zrychlení** (acceleration) - derivace rychlosti podle času
- **ryv** (jerk) - derivace zrychlení. Změna zrychlení je významným faktorem, i když velmi často opomíjeným. Např. u CNC strojů je limitujícím faktorem pro akceleraci změny pohybu nástroje. V případě lidského vnímání je faktorem pohody v dopravních prostředcích, nebo naopak nepohody v případě pouťových atrakcí, např. horské dráhy.

Někdy je nutné zobrazovat fyzikální vlastnosti v jiném než kartézském souřadném systému. Většinou při vizualizacích dat bude nutné určovat normálu plošky pro stínování, apod. Vektor normály se ztotožňuje s vektorem gradientu potenciálového pole, např. při extrakci iso-plochy.

25.1. Diferenciální operátory

Základní operace s operátory:

$\nabla(f + g) = \nabla f + \nabla g$	$\operatorname{div}(\mathbf{a}f) = f \operatorname{div} \mathbf{a} + \mathbf{a} \operatorname{grad} f$
$\nabla(fg) = f \nabla g + g \nabla f$	$\operatorname{rot}(\mathbf{a}f) = f \operatorname{rot} \mathbf{a} + \mathbf{a} \times \operatorname{grad} f$
$\nabla(\nabla f) = \operatorname{div} \operatorname{grad} f = \Delta f$	$\operatorname{div}(\mathbf{a} \times \mathbf{b}) = \mathbf{b} \operatorname{rot} \mathbf{a} - \mathbf{a} \operatorname{rot} \mathbf{b}$
$\operatorname{rot} \operatorname{grad} f = 0$	$\operatorname{rot}(\operatorname{rot}(\mathbf{a})) = \operatorname{grad} \operatorname{div}(\mathbf{a}) - \Delta \mathbf{a}$
$\operatorname{rot}(\mathbf{a} \times \mathbf{b}) = (\mathbf{b} \operatorname{grad}) \mathbf{a} - (\mathbf{a} \operatorname{grad}) \mathbf{b} + \mathbf{a} \operatorname{div} \mathbf{b} - \mathbf{b} \operatorname{div} \mathbf{a}$	

Níže uvedené vzorce a převodní vztahy je nutné respektovat i pro vizualizaci dat (některé vektory jsou vektory **řádkové**). V dalším předpokládáme, že f je skalární hodnota, \mathbf{a} je vektor

Souřadný systém kartézský	(x, y, z)
Operátor ∇	$\nabla \stackrel{\text{def}}{=} \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right]^T$
Gradient ∇f Výsledek je vektor	$\operatorname{grad} f = \nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right]^T$
Divergence $\nabla \cdot \mathbf{a}$ Výsledek je skalár	$\operatorname{div} \mathbf{a} = \nabla^T \mathbf{a} = \nabla \cdot \mathbf{a} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z}$
Rotace $\nabla \times \mathbf{a}$ Výsledek je vektor	$\operatorname{curl} \mathbf{a} = \operatorname{rot} \mathbf{a} = \nabla \times \mathbf{a} = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right]^T \times [a_x, a_y, a_z]^T$ $= \left[\frac{\partial a_z}{\partial y} - \frac{\partial a_y}{\partial z}, \frac{\partial a_x}{\partial z} - \frac{\partial a_z}{\partial x}, \frac{\partial a_y}{\partial x} - \frac{\partial a_x}{\partial y} \right]^T$
$\nabla^2 f = \Delta f$ (gradient divergence) Výsledek je skalár	$\nabla^2 f = \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$

Souřadný systém válcový	(r, θ, z)
Operátor ∇	$\nabla \stackrel{\text{def}}{=} \left[\frac{\partial}{\partial r}, \frac{1}{r} \frac{\partial}{\partial \theta}, \frac{\partial}{\partial z} \right]^T$
Gradient ∇f Výsledek je vektor	$\text{grad } f = \nabla f = \left[\frac{\partial f}{\partial r}, \frac{1}{r} \frac{\partial f}{\partial \theta}, \frac{\partial f}{\partial z} \right]^T$
Divergence $\nabla \cdot \mathbf{a}$ Výsledek je skalár	$\text{div } \mathbf{a} = \nabla^T \mathbf{a} = \nabla \cdot \mathbf{a} = \left[\frac{\partial a_r}{\partial r} + \frac{a_r}{r} + \frac{1}{r} \frac{\partial a_\theta}{\partial \theta} + \frac{\partial a_z}{\partial z} \right]$
Rotace $\nabla \times \mathbf{a}$ Výsledek je vektor	$\text{curl } \mathbf{a} = \text{rot } \mathbf{a} = \nabla \times \mathbf{a} = \left[\frac{\partial}{\partial r}, \frac{1}{r} \frac{\partial}{\partial \theta}, \frac{\partial}{\partial z} \right]^T \times [a_r, a_\theta, a_z]^T$ $= \left[\frac{1}{r} \frac{\partial a_z}{\partial \theta} - \frac{\partial a_\theta}{\partial z}, \frac{\partial a_r}{\partial z} - \frac{\partial a_z}{\partial r}, \frac{1}{r} \left(\frac{\partial (ra_\theta)_z}{\partial r} - \frac{\partial a_r}{\partial \theta} \right) \right]^T$
$\nabla^2 f = \Delta f$ (gradient divergence) Výsledek je skalár	$\nabla^2 f = \Delta f = \frac{\partial^2 f}{\partial r^2} + \frac{1}{r} \frac{\partial f}{\partial r} + \frac{1}{r^2} \frac{\partial^2 f}{\partial \theta^2} + \frac{\partial^2 f}{\partial z^2}$

1

Souřadný systém sférický	(r, ϕ, θ)
Operátor ∇	$\nabla \stackrel{\text{def}}{=} \left[\frac{\partial}{\partial r}, \frac{1}{r} \frac{\partial}{\partial \phi}, \frac{1}{r \sin \phi} \frac{\partial}{\partial \theta} \right]^T$
Gradient ∇f Výsledek je vektor	$\text{grad } f = \nabla f = \left[\frac{\partial f}{\partial r}, \frac{1}{r} \frac{\partial f}{\partial \phi}, \frac{1}{r \sin \phi} \frac{\partial f}{\partial \theta} \right]^T$
Divergence $\nabla \cdot \mathbf{a}$ Výsledek je skalár	$\text{div } \mathbf{a} = \nabla^T \mathbf{a} = \nabla \cdot \mathbf{a} = \left[\frac{1}{r^2} \frac{\partial (r^2 a_r)}{\partial r} + \frac{1}{r \sin \phi} \left(\frac{\partial (\sin \phi a_\phi)}{\partial \phi} + \frac{\partial a_\theta}{\partial \theta} \right) \right]$
Rotace $\nabla \times \mathbf{a}$ Výsledek je vektor	$\text{curl } \mathbf{a} = \text{rot } \mathbf{a} = \nabla \times \mathbf{a} = \left[\frac{\partial}{\partial r}, \frac{1}{r} \frac{\partial}{\partial \phi}, \frac{1}{r \sin \phi} \frac{\partial}{\partial \theta} \right]^T \times [a_r, a_\phi, a_\theta]^T =$ $= \left[\frac{1}{r \sin \phi} \left(\frac{\partial (\sin \phi a_\phi)}{\partial \phi} - \frac{\partial a_\theta}{\partial \theta} \right), \frac{1}{r} \left(\frac{1}{\sin \phi} \frac{\partial a_r}{\partial \theta} - \frac{\partial (ra_\theta)}{\partial r} \right), \frac{1}{r} \left(ra_\phi - \frac{\partial a_r}{\partial \phi} \right) \right]^T$
$\nabla^2 f = \Delta f$ (gradient divergence) Výsledek je skalár	$\nabla^2 f = \Delta f = \frac{\partial^2 f}{\partial r^2} + \frac{2}{r} \frac{\partial f}{\partial r} + \frac{1}{r^2 \sin^2 \phi} \frac{\partial^2 f}{\partial \theta^2} + \frac{\cos \phi}{r^2 \sin \phi} \frac{\partial f}{\partial \phi} + \frac{1}{r^2} \frac{\partial^2 f}{\partial \phi^2}$
Gradient ve sférických souřadnicích	$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} \cos \theta \sin \phi & r \cos \theta \cos \phi & -r \sin \theta \sin \phi \\ \sin \theta \sin \phi & r \sin \theta \cos \phi & r \cos \theta \sin \phi \\ \cos \phi & -r \sin \phi & 0 \end{bmatrix} \begin{bmatrix} dr \\ d\phi \\ d\theta \end{bmatrix}$

2

3 Další podrobnosti lze nalézt

- 4 • <http://hyperphysics.phy-astr.gsu.edu/hbase/gradi.html#c2>
- 5 • http://en.wikipedia.org/wiki/Spherical_coordinate_system
- 6 • <http://mathworld.wolfram.com/topics/CoordinateGeometry.html>

7 Z výše uvedeného je zřejmé, že výpočet gradientu, resp. normály povrchu, např. ve sférickém
8 souřadném systému je dán vektorem:

$$\left[\frac{\partial f}{\partial r}, \frac{1}{r} \frac{\partial f}{\partial \phi}, \frac{1}{r \sin \phi} \frac{\partial f}{\partial \theta} \right]^T$$

9 a nelze tedy použít

$$\left[\frac{\partial f}{\partial r}, \frac{\partial f}{\partial \phi}, \frac{\partial f}{\partial \theta} \right]^T$$

1 Při řešení problémů je obecně možno využít toho, že daný problém se převede za určitých podmínek
2 do jiného souřadného systému, ve kterém se výpočet provede snadněji, a vypočtené výsledky se
3 potom transformují zpět do původního souřadného systému.
4

5 **25.2. Integrální operátory**

6 Vedle diferenciální formulace problémů existuje také integrální formulace. Velmi často se využívají
7 integrální transformace, např. Laplaceova k převedení diferenciální reprezentace na reprezentaci
8 algebraickou, pomocí které se daný problém řeší.

$$F(s) = \mathcal{L}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-st} dt$$

9 kde s je komplexní číslo. Pak lze ukázat, že problém z časové domény se převede do domény
10 algebraické např. takto:

$$\boxed{\mathcal{L}\{f'(t)\} = sF(s) - f(0) \quad \mathcal{L}\{f''(t)\} = s^2F(s) - sf(0) - f'(0)}$$

12 Výřešený problém se pak převádí inverzní Laplaceovou transformací zpět do časové reprezentace.
13

14 Laplaceovy transformace, většinou jednostranné, se využívá v mnohých technických oborech, např.
15 při návrhu systémů řízení technologických procesů apod.

16 Další informace lze najít v příslušné odborné literatuře, neboť integrální transformace nejsou
17 předmětem tohoto textu.
18

19

20

1 26. Příloha E – Řecká abeceda

2 Pro úplnost ještě uvedeme řeckou abecedu, neboť není běžnou součástí vzdělání v současné době.

^+A	α	^+A	α	alfa	a	1
^+B	β	B	β	beta	b	2
Γ	γ	Γ	γ	gama	g	3
Δ	δ	Δ	δ	delta	d	4
^+E	ε	^+E	ε	epsílon	e	5
F	$\sigma\tau$	f	S	digamma	di	6
^+Z	ζ	^+Z	ζ	dzéta	dy	7
^+H	η	^+H	η	éta	é	8
Θ	ϑ	Θ	ϑ	théta	th	9
^+I	ι	^+I	ι	(i)jota	i	10
^+K	κ	^+K	κ	kappa	k	20
Λ	λ	Λ	λ	lambda	l	30
^+M	μ	^+M	μ	mý	m	40
^+N	η	^+N	η	ný	n	50
Ξ	ξ	Ξ	ξ	ksí	ks	60
^+O	^+o	^+O	^+o	omíkron	o	70
Π	π	Π	π	pí	p	80
				san	z	
Ω				koppa		90
^+P	ρ	^+P	ρ	ró	r	100
Σ	σ	Σ	σ	sigma	s	200
	ς		ς	sigma – konec slova	s	200
^+T	τ	^+T	τ	tau	t	300
^+Y	^+v	^+Y	^+v	ypsilon	y	400
Φ	φ	Φ	φ	fí	f	500
^+X	χ	^+X	χ	chí	ch	600
Ψ	ψ	Ψ	ψ	psí	ps	700
Ω	ω	Ω	ω	ómega	ó	800
\eth				sampi		900

3 Písmena označená “ $^+$ ” se neužívají, aby se nezaměňovala s písmeny latinské abecedy.

4

5 Řecká čísla byla vyjadřována pomocí řecké abecedy. Dřívější systém byl typu „acrophonic“, podobný římským číslicím, a to:

6 I = 1, Π = 5, Δ = 10, H = 100, X = 1000, M = 10000

7

8 Pravděpodobně ve 4. stol. př.n.l. byl systém nahrazen kvazi-dekadickým systémem, který se nazýval „Ionic numeral system“.

9

- 1 Pro odlišení textu a čísel se používal znak „ „ za číslem, takže
 2 $\sigma\mu\alpha' = 200+40+1 = 241$
- 3 Znak „ „ před číslem znamenala tisíce, takže $\beta\delta' = 2000+4 = 2004$
- 4 $\omega - \omega$ se „střesem“
- 5 ζ - jako σ ale na konci „světa“
- 6
- 7 Refs:
- 8 • Mikulčák,J. a kol.: Matematické fyzikální chemické tabulky pro střední školy, SPN, 1970
- 9 • http://www.wordiq.com/definition/Greek_numerals
- 10

Letter	Value	Letter	Value	Letter	Value
α'	<u>1</u>	$\acute{\iota}$	<u>10</u>	ρ'	<u>100</u>
β'	<u>2</u>	κ'	<u>20</u>	σ'	<u>200</u>
γ'	<u>3</u>	λ'	<u>30</u>	τ'	<u>300</u>
δ'	<u>4</u>	μ'	<u>40</u>	υ'	<u>400</u>
ϵ'	<u>5</u>	ν'	<u>50</u>	ϕ'	<u>500</u>
ζ' or τ'	<u>6</u>	ξ'	<u>60</u>	χ'	<u>600</u>
ζ'	<u>7</u>	\circ'	<u>70</u>	ψ'	<u>700</u>
η'	<u>8</u>	π'	<u>80</u>	ω'	<u>800</u>
θ'	<u>9</u>	ς'	<u>90</u>	$\grave{\alpha}'$	<u>900</u>

11

Řecká písmena	12
<u>A α Alpha</u>	<u>B β Beta</u>
<u>G γ Gamma</u>	<u>D δ Delta</u>
<u>E ϵ Epsilon</u>	<u>F φ Digamma</u>
<u>Z ζ Zeta</u>	<u>H η Eta</u>
<u>Theta θ</u>	<u>I ι Iota</u>
<u>K κ Kappa</u>	<u>L λ Lambda</u>
<u>M μ Mu</u>	<u>N ν Nu</u>
<u>Xi ξ</u>	<u>O \circ Omicron</u>
<u>Pi π</u>	<u>San</u>
<u>Qoppa Ω</u>	<u>Rho ρ</u>
<u>Sigma σ</u>	<u>Tau τ</u>
<u>Upsilon υ</u>	<u>Phi ϕ</u>
<u>Chi χ</u>	<u>Psi ψ</u>
<u>Omega ω</u>	<u>Sampi $\grave{\alpha}$</u>