Algoritmy počítačové grafiky I

(Algorithms for Computer Graphics I)

Prof.Ing.Václav Skala, CSc.

http://www.VaclavSkala.eu

Abstrakt

Algoritmy počítačové grafiky a jejich implementace je nedílnou součástí jak grafických systémů, tak i CAD/CAM systémů. V této práci jsou uvedeny základní algoritmy, metody a postupy, které se používají v různých modifikacích i v dnešních systémech

Abstract

Algorithms for computer graphics and their implementation is a part of graphical systems and CAD/CAM systems. Fundamental algorithms, methods and approaches used in today's systems are explained.

Algoritmy počítačové grafiky I
Algorithms for Computer Graphics I
Published & printed by: Vaclav Skala – UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Year 2011
ISBN 978-80-86943-19-0

Electronic version http://www.VaclavSkala.eu http://Graphics.zcu.cz

Profil autora

Prof. Ing. Václav Skala, CSc. se zabývá počítačovou grafikou od r.1975, kdy na Vysoké škole strojní a elektrotechnické v Plzni zavedl a následně vyučoval předmět Počítačová grafika a umělá inteligence. V roce 1981 obhájil disertační práci na témate relačních databází se specializací na reprezentaci relací a optimalizaci dotazů uživatele. V současné době se odborně věnuje především algoritmům počítačové grafiky a vizualizaci dat a algoritmům včetně matematických aspektů. Je vedoucím Centra počítačové grafiky a vizualizací (http://Graphics.zcu.cz) při Katedře informatiky a výpočetní techniky na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

V letech 1984-1989 působil na Brunel University v Londýně a později přednášel na NATO Advanced Study Institute v zahraničí. V roce 1996 se stal profesorem na Západočeské univerzitě, odborně působil na Bath University v U.K., Gavle University ve Švédsku a na dalších zahraničních odborných pracovištích.

Prof.Skala je řešitelem zahraničních i národních odborných výzkumných projektů, členem několika redakčních rad prestižních zahraničních odborných časopisů, členem programových výborů mezinárodních odborných konferencí. Od r.1992 je organizátorem mezinárodních odborných konferencí WSCG zaměřených na počítačovou grafiku, vizualizaci dat a počítačové vidění (http://www.WSCG.eu).

V současné době je prof. Skala profesorem na Západočeské univerzitě v Plzni a VŠB-Technické univerzitě v Ostravě. V roce2010 odborně působil též na Přírodovědné fakultě Ostravské univerzity v Ostravě (http://AlgVis.osu.cz).

V roce 2010 získal významné ocenění mezinárodní asociace pro počítačovou grafiku

„Fellow of the Eurographics Association“

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

Poznámky pro laskavého čtenáře

Toto je rekonstrukce učebního textu Algoritmy počítačové grafiky I – III, který vznikl pro studenty Západočeské univerzity v Plzni v roce 1991. Tato publikace vznikla rozšířením publikace Počítačová grafika I - II, která vznikala v roce 1989 pro potřeby studentů Vysoké školy strojní a elektrotechnické a která byla vydána v roce 1990.

Jde tedy o text zohledňující algoritmy, metody a stav technologie před více než před 20 lety. Je nutné si uvědomit, že obor počítačové grafiky je poměrně velmi mladý, bouřlivě se rozvíjející a poměrně hodně závislý na dostupných technologiích. Nicméně většina metod, postupů a algoritmů je používána dodnes.

Pochopitelně je oblast počítačové grafiky dnes podstatně širší a zahrnující mnohé jiné oblasti, které jsou odborně rozvíjeny kolegy v **Centru počítačové grafiky a vizualizací** při Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd, viz http://Graphics.zcu.cz

Rád bych proto požádal čtenáře, aby uvedený text a kvalitu tisku posoudil v kontextu té doby, kdy:

- nebyl prakticky přístup k zahraniční literatuře a publikace se velmi obtížně získávaly přes mezinárodní výpůjčky, resp. přes osobní kontakty
- nebyl k dispozici Internet, e-mail, WEB a digitální knihovny
- byly k dispozici první 8mi bitové PC systémy s „fantastickou“ kapacitou paměti 512 KB
- byly k dispozici první jehličkové tiskárny (9 tiskových bodů na textovou řádku) a „unikátní“ textové procesory T602 a výborný graficko-textový procesor Chi-Writer (http://en.wikipedia.org/wiki/ChiWriter), ve kterém byl text napsán.

Je nutné si uvědomit, že před rokem 1989 bylo velmi obtížné získávat odborné publikace a publikovat odborné výsledky v mezinárodně uznávaných časopisech a na mezinárodních konferencích.

Již v roce 1975 na Vysoké škole strojní a elektrotechnické, Fakultě elektrotechnické vzniká předmět Počítačová grafika a umělá inteligence

Nové odborné výsledky, algoritmy a metody laskavý zájemce nalezne na URL:
http://www.VaclavSkala.eu.

Také bych rád zájemce o počítačovou grafiku odkázal na publikace z mezinárodních konferencí WSCG, na časopis Journal of WSCG a GraVisMa workshopy:

- **WSCG** – International Conferences on Computer Graphics, Visualization and Computer Vision
  http://www.WSCG.eu
- **GraVisMa –** Computer Graphics, Vision and Mathematics http://www.GraVisMa.eu

Tento text, který vznikl rekonstrukcí z archivu autora, byl doplněn o seznam publikací tak, aby tvořil ucelenou publikaci. Předpokládá se, že další části původní publikace budou rekonstruovány následně.

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

Prof. M.L.V. Pitteway

Brunel University,
U.K.

Prof. Jack E. Bresenham

IBM Corp., USA
a
Winthrop University,
USA

Prof. F.R.A.Hopgood

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

Prof. David F.Rogers

U.S. Naval Academy,
USA

V Plzni dne: 2011-07-25 prof.Ing.Václav Skala, CSc.

Předmluva

V publikaci Algoritmy počítačové grafiky I - III jsem se pokusil zachytit současný stav v oblasti principů vybraných algoritmů počítačové grafiky a CAD systémů. Vzhledem k prudkému rozvoji použití technických a programových prostředků počítačové grafiky je tato publikace do určité míry pojata jako přehledová, přičemž podrobnosti lze nalézt v literatuře. Některé pasáže bylo nutno zredukovat na téměř informativní úroveň z důvodů přílišné teoretické složitosti, složitosti algoritmů, případně pro jejich nedostupnost, neboť firmy skutečně použité metody nepublikují.

Pro další studium v oblasti počítačové grafiky je nezbytné sledovat odbornou literaturu, na kterou bych rád upozornil, a to zejména na:

a) časopisy

Computer Graphics Forum, North Holland Comp.*
ACM Transaction on Graphics, ACM*
The Visual Computer, Springer Verlag+
Computer Aided Geometric Design, North Holland Comp.+
Computational Geometry, Theory and Applications, Elsevier Science Publ.+
Computer Aided Design, Butterworth & Co (Publishers) Ltd.+
Computers & Graphics, Pergamon Press

b) sborníky konferencí

EUROGRAPHICS* (EUROGRAPHICS Association),
SIGGRAPH+ (USA),
Computer Graphics International+

Publikace označené *, + je možné vypůjčit přes knihovnu Západočeské univerzity (publikace označené + netvoří celý komplet).

Za základní publikace v oblasti počítačové grafiky lze považovat [5], [11], [44], [55], [56], [66], [69], [72], [94], [97], [104], [108], [109], [112], [114], [123], přičemž ostatní, uvedené v seznamu literatury, lze označit za doplňkové, sloužící především k dalšímu studiu dané problematiky, případně partií souvisejících s počítačovou grafikou. Celkový seznam použité literatury je uveden v 3. díle této publikace.

Je mou milou povinností poděkovat všem, kteří přispěli ke vzniku této publikace, ať už vytvořením podmínek nebo stimulací k vlastní práci. Chtěl bych zejména poděkovat ing. I. Kolingerové za její konkrétní, podnětné a kritické připomínky, bývalým i současným studentům Západočeské univerzity v Plzni se zaměřením Počítačová grafika a CAD systémy, kteří byli neocenitelnými pomocníky při ověřování algoritmů a tvorbě demonstračních a výukových programů, které jsou případným zájemcům k dispozici. Skripta byla do formátu "camera ready" připravena pomocí textového procesoru ChiWriter, který byl laskavě zapůjčen firmou Horstmann Software (USA) a který byl neocenitelným pomocníkem. Většina obrázků byla realizována pomocí programu COREL DRAW, který se ukázal být velmi dobrým nástrojem pro daný účel.

Budu zavázán všem čtenářům za jakékoliv připomínky, návrhy či doplnění předkládaného textu. Uvítám rovněž případné informace o aplikaci předkládaných algoritmů, jejich modifikací či zcela nových algoritmech počítačové grafiky.

Obsah

1. Úvod 1
   1.1 Vymezení obsahu počítačové grafiky 2
   1.2 Aplikace počítačové grafiky 4
2. Grafická zařízení 12
   2.1 Vstupní zařízení 12
   2.2 Výstupní zařízení 24
   2.3 Netradiční vstupní a výstupní zařízení 45
   2.4 Příklady 47
3. Základní algoritmy počítačové grafiky 54
   3.1 Algoritmus pro kreslení čáry 54
   3.2 Digitální diferenciální analyzátor 55
   3.3 Bresenhamův algoritmus 59
   3.4 Generátor kružnice 66
   3.5 Generátor kuželoseček 76
   3.6 Kódování grafické informace 77
   3.7 Reprezentace n-úhelníků a oblastí 85
   3.8 Plnění a šrafování 90
   3.9 Antialiasing 99
   3.10 Reprezentace třírozměrných objektů 102

1. Úvod

Počítačová grafika je oblastí výpočetní techniky, která se zabývá znázorňováním dat v grafické podobě. Umožňuje velmi rychlou komunikaci mezi člověkem a počítačem nejen po stránce rychlosti vlastního zobrazování, ale i z hlediska množství předávané informace. Člověk pomocí zrakového vjemu může absorbovat grafickou informaci o několik řádů rychleji než při předávání informace pomocí tabulek čísel apod. Důležitost počítačové grafiky narůstá s neustále klesající cenou hardwaru jak jednotlivých grafických periferií, tak i osobních mikropočítačů, jejichž dnešní parametry již běžně dosahují parametrů minipočítačů.

S rozvojem technologií v oblasti výpočetní techniky nastává odklon od tradičních grafických periferií, které jsou založeny na kreslení čar, k rastrovým zařízením založeným na zobrazování jednotlivých bodů s využitím široké barevné palety. S tímto vývojem vyvstává nutnost vytváření nových algoritmů specializovaných na rastrové prostředí. Mnohé algoritmy uvedené v této publikaci jsou též použitelné pro běžná zařízení, která se v praxi používají.

Rozvoj počítačové grafiky v ČSFR lze dokumentovat na vzrůstajícím počtu systémů vhodných pro počítačovou grafiku. Kdysi k nim patřily zejména systémy ISAP založené na řadě minipočítačů SM 52/11, resp. SM 52/12, systémy IGS založené na řadě minipočítačů ADT 4500, resp. ADT 4700. Velmi rozšířené systémy PC AT, PC 386, resp. PS/2 jsou dnes k dispozici prakticky na všech pracovištích. Tyto systémy jsou v mnoha případech doplňovány různými grafickými zobrazovacími jednotkami s velkou rozlišovací schopností a akcelerátory. V současné době se rozšiřují zejména systémy PC 386/387, resp. PC 486, které ve spojení s videokartou Super VGA, resp. ARTIST apod. mají pro běžné aplikace nejen postačující kapacitu paměti a výpočetní výkon, ale i postačující výkonnost v oblasti grafických operací. V krátké budoucnosti lze očekávat používání počítačů s architekturou RISC a grafických pracovních stanic, které se vyznačují velkým pracovním výkonem, zejména pak počtem prováděných grafických operací. Patří k nim nejen pracovní stanice typů IRIS, SUN, APOLLO, VRX firmy Hewlett Packard, ale i systémy firmy INTERGRAPH, které jsou využívány zejména

v oblasti GIS (geografických informačních systémů) a pro aplikace v oblasti kartografie a geodézie.

Se systémy je dodáváno nejen základní programové vybavení pro počítačovou grafiku (např. normy GKS, PHIGS), ale i ucelené systémy orientované uživatelsky. Jsou to např. MicroStation PC (CADD) firmy INTERGRAPH, Eagle, AutoCAD, UltiBoard a UltiCap firmy ULTIMATE pro návrh plošných spojů, resp. OrCAD. Navíc např. systém MicroStation PC, který je určen pro prostředí PC a MS DOS, a systém CADD, který je určen pro pracovní stanice firmy INTERGRAPH, se chovají vůči uživateli stejně, i když se provozují na technice zcela rozdílné.

1.1 Vymezení obsahu počítačové grafiky

Vymezit obsah počítačové grafiky je poměrně složité, neboť dnes zasahuje do mnoha oblastí spojených s aplikací výpočetní techniky, které nejsou jen přímého aplikačního rázu, ale i aplikací, jejichž těžiště je především ve výzkumu nových technologií. Aplikace počítačové grafiky lze nalézt i v mnoha netradičních oblastech, např. animace filmů apod.

Zpracování grafické informace se zejména vyznačuje:

- extrémním objemem zpracovávaných dat,
- numerickou náročností jednotlivých výpočtů,
- vysokými nároky na všechny hardwarové parametry výpočetního systému, zejména pak na kapacitu paměti.

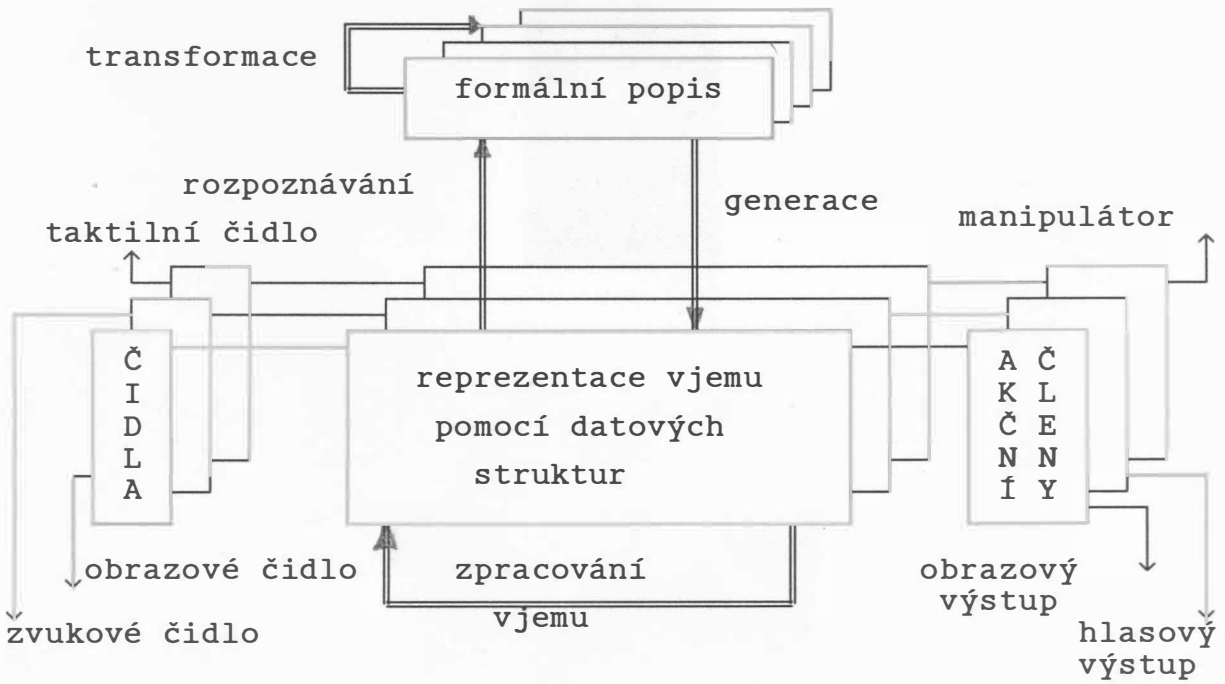
Některé typy úloh, zejména simulační, vyžadují superpočítače, kdy ani použití systému CRAY není postačující, např. z důvodů rychlosti odezvy systému. Pro efektivní použití a tvorbu nových grafických systémů je však nezbytné pochopit jednotlivé základní principy metod a algoritmů, i když bude neustále narůstat použití specializovaných grafických procesorů a akcelerátorů.

Vymezení oblasti počítačové grafiky je charakterizováno tabulkou 1.1.1, kde je počítačová grafika chápána úzce jako tzv. generativní grafika, tj. jako proces, který ze zadaného popisu vytvoří obrazový výstup. Toto vymezení, i když dosti schematické, vystihuje hlavní problémy počítačové grafiky. V současné době je jedním z hlavních směrů výzkumu např. řešení

problému věrnosti zobrazení, kde vedle výpočetních problémů je též nutné řešit problematiku použití barev a barevných systémů.

vstup je dán jako:	výstup je dán jako:		
	popis	obrázek	zvuk
popis	symbolická manipulace	počítačová grafika	hlasový výstup
obrázek	rozpoznávání obrazů	zpracování obrazů	
zvuk	rozpoznávání zvuku		zpracování zvuku

Tabulka 1.1.1



Obr.1.1.1

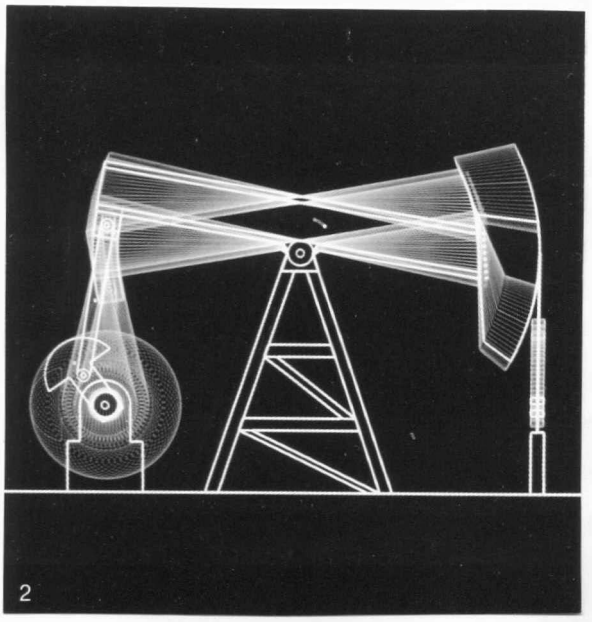
Funkční začlenění počítačové grafiky naznačuje obr.1.1.1. Při symbolické manipulaci, např. symbolické derivaci, úpravě vzorců atd., dochází vlastně k transformaci formálního popisu. V případě rozpoznávání obrazu jde o nalezení formálního popisu daného obrazu, zatímco při generaci obrazu, tj. použití aparátu počítačové grafiky, jde o postup opačný. Na mnohé problémy z oblasti počítačové grafiky a zpracování obrazu lze pohlížet jako na problémy **duální**. Z tohoto důvodu je také někdy oblast počítačové grafiky chápána šířeji a zahrnuje pak i oblasti zpracování a rozpoznávání obrazu.

Za obrazové čidlo lze považovat např. kameru, řádkový snímač (scanner), atd. Zvukovým čidlem může být např. mikrofon, zatímco taktilním čidlem může být např. spínač vymezující polohu nebo snímač tlaku. Hlasový výstup může být reprezentován např. reproduktorem, obrazový výstup např. monitorem.

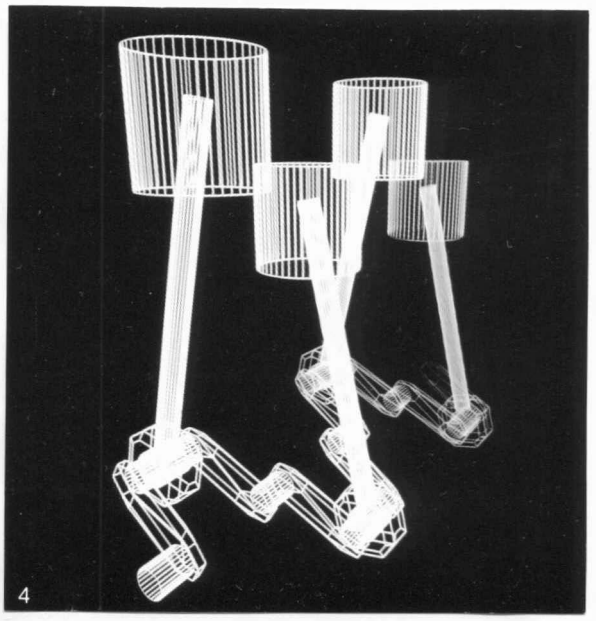
1.2 Aplikace a použití počítačové grafiky

Počítačová grafika se dnes používá v nejrůznějších odvětvích lidské činnosti. Je poněkud paradoxní, že ačkoliv byla tato oblast rozvinuta především z hlediska možných technických aplikací, je jednou z nejvíce finančně výhodných oblastí právě oblast ryze netechnická, a tou je animace filmů. V dnešní době existují specializované společnosti, které nejenže vytvářejí speciální programové vybavení, ale též i příslušné systémy. Je zřejmé, že tato oblast vyžaduje zcela odlišné prostředky než jsou technické aplikace. Mezi prvními filmy, které byly vytvořeny za pomoci animace, je známý film **TRON**. Ukázky z oblasti počítačové animace jsou na obr.1.2.1 a obr.1.2.2. Kromě vlastní generace obrázků je nutné též napodobit plynulost pohybů apod. Problematika počítačové animace není u nás zcela známá. V literatuře lze však nalézt mnoho informací, viz např. [129], [171]. Animační aspekty lze nalézt i u různých simulačních programů, např. z oblasti kosmického výzkumu.

Je pochopitelné, že pro účely počítačové animace musí být k dispozici speciální výstupní zařízení, neboť např. rozlišovací schopnost 1024 x 1024 není postačující při zvětšení např. na velikost promítací plochy v kině.

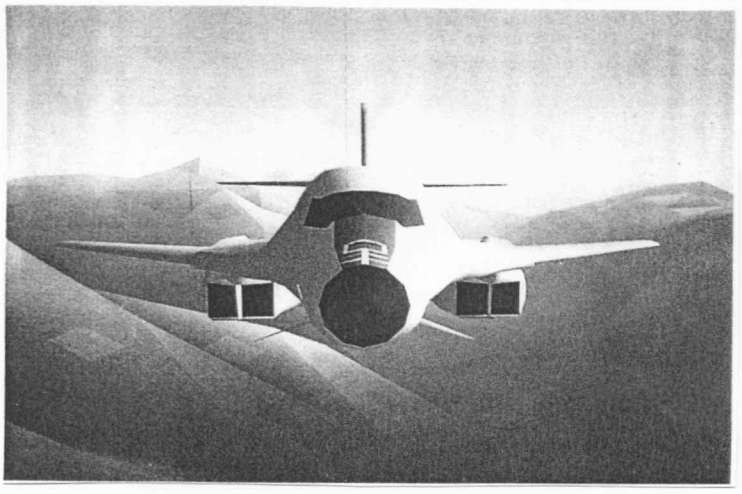


Obr. 1. 2. 1

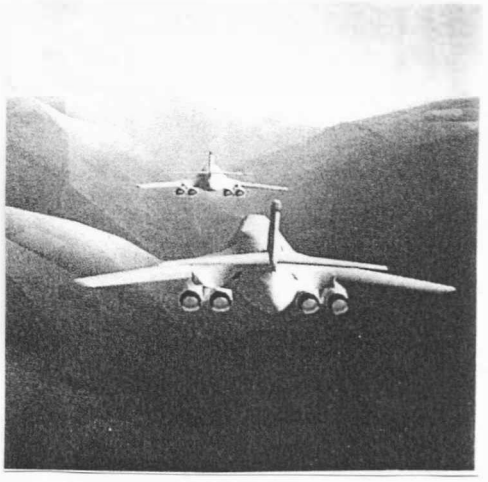


Obr. 1. 2. 2

Jinou oblastí aplikace počítačové grafiky je použití simulátorů pilotáže letadel, lodí, aut apod. Tyto systémy byly v první fázi motivovány především možnostmi využití při výzkumu vesmíru, ve vojenství apod. Skutečné simulátory, např. pilotáže letadel, obsahují též složité prvky mechanického charakteru, které zajišťují náklon kabiny, vytváření pocitu přetížení apod. Mezi špičkové systémy patří např. systémy firmy SINGER Link Miles (U.K.) a Marconi Radar Systems Ltd., viz obr. 1. 2. 3. a obr. 1. 2. 4.

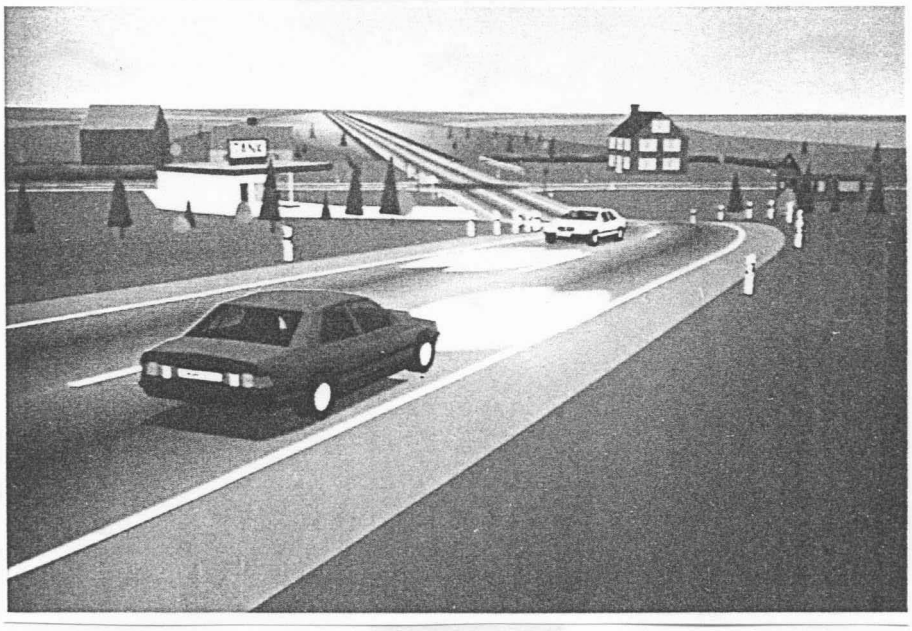


Obr. 1. 2. 3

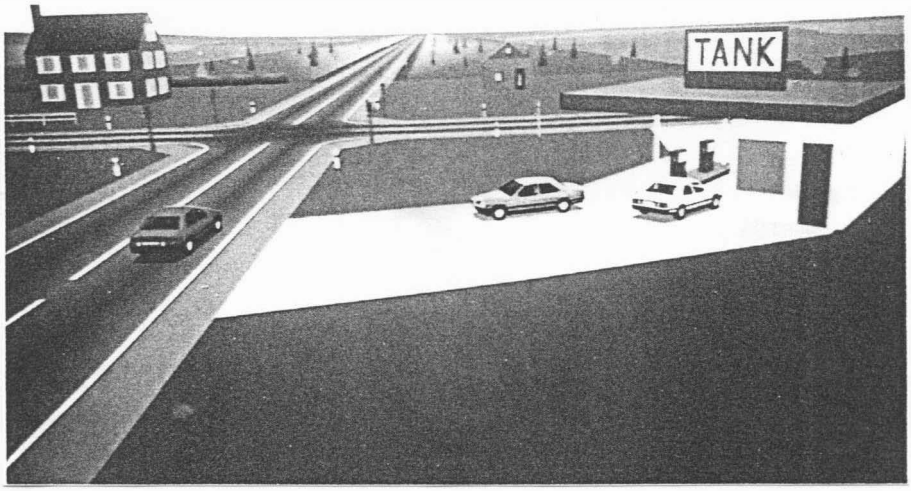


Obr. 1. 2. 4

Navíc uvedené systémy umožňují simulaci chování přesně podle aerodynamických parametrů a též simulaci nejrůznějších poruch apod. Z důvodů reálnosti poskytují tyto systémy vjem **hloubky prostoru**. Dnešní simulační systémy navíc umožňují např. simulaci letu nad rozsáhlým terénem, neboť rozvoj paměťových technologií umožňuje vytvářet obrovské databáze (systém CT6 umožňuje reprezentovat 10 000 čtverečních mil při hustotě až 1.5 miliónů n-úhelníků na čtvereční míli). Na obr.1.2.5 a obr.1.2.6 jsou ukázky grafického výstupu u automobilového trenažéru.

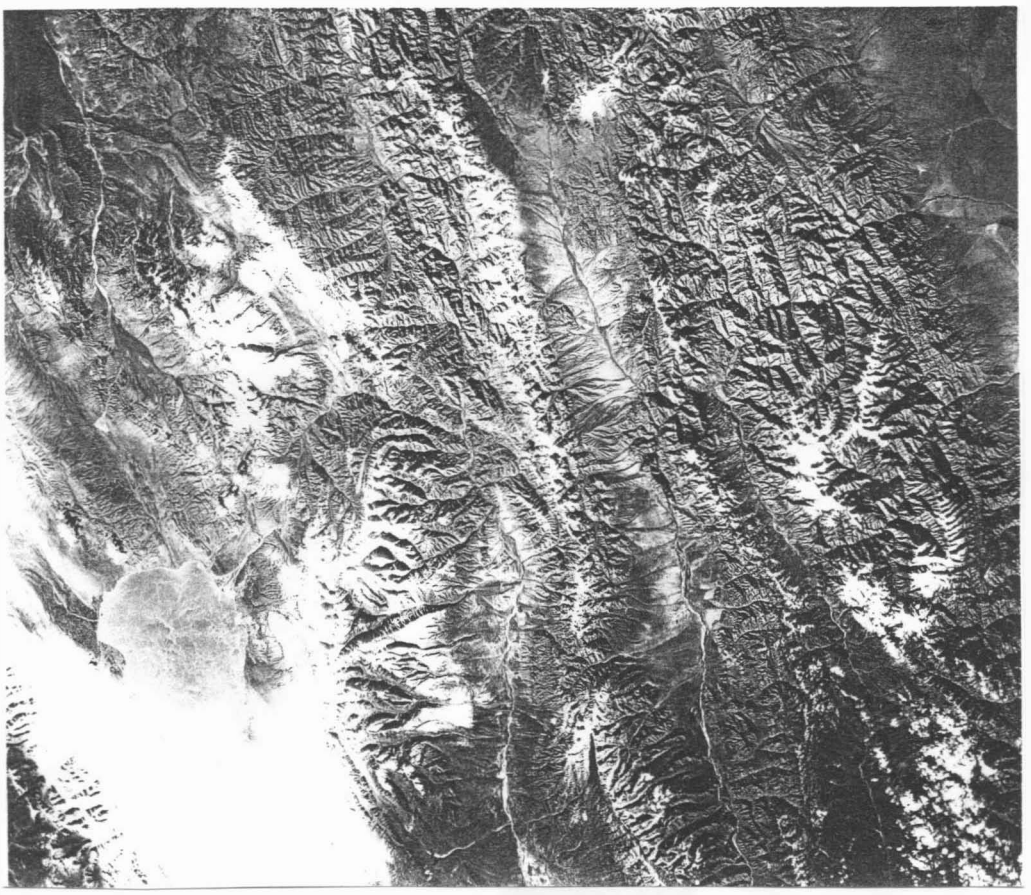


Obr.1.2.5



Obr.1.2.6

Vedle generativní počítačové grafiky je též výstup při zpracování obrazu velmi důležitou aplikací. Jako příklad si lze uvést snímek části Tibetu s jezerem Hara Nuur, které je ve výšce asi 5000 m/m. Tento snímek byl pořízen v průběhu letu stanice Spacelab-1, viz obr. 1.2.7.

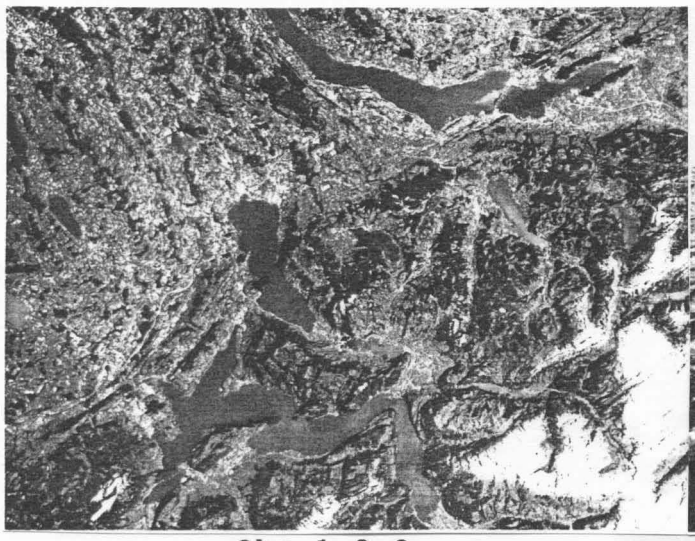


Obr.1.2.7

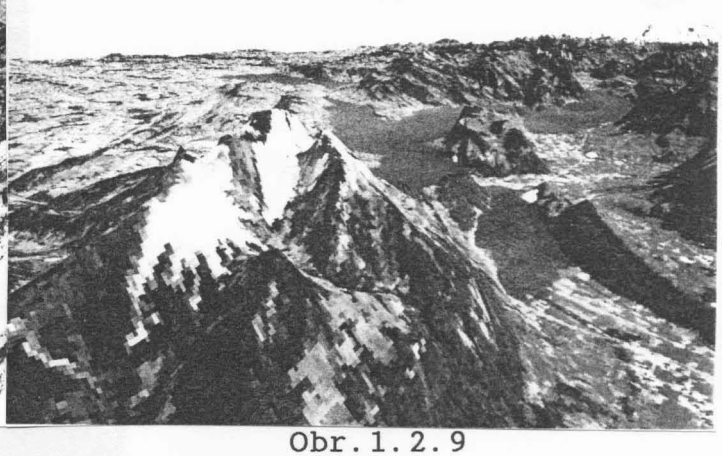
Další ukázkou je snímek střední části Švýcarska (asi 60 x 80 km okolí Lucernu), který byl pořízen systémem Landsat - TM, viz obr.1.2.8, a který byl následně zpracován tak, že byl vytvořen perspektivní pohled z místa, které je označeno na původním obrázku šipkou vlevo dole, viz obr.1.2.9. Originál obrazu obsahuje 2500 x 3000 pixelů, po perspektivní transformaci pak 1000 x 710 pixelů.

Je zřejmé, že techniky dálkového průzkumu Země umožňují i monitorování určité oblasti. Na obr.1.2.10 je snímek oblasti Černobylu před havárií jaderné elektrárny. Snímek na obr.1.2.11 ukazuje situaci 12 dní po havárii. Zvýšená teplota vody v nádrži je identifikována změnou barvy, resp. vyšším jasem. Takové monitorovací techniky lze použít v mnoha oblastech, zejména pak

při použití např. kamer pracujících v infračervené části spektra.



Obr. 1. 2. 8



Obr. 1. 2. 9

Obr. 1. 2. 10

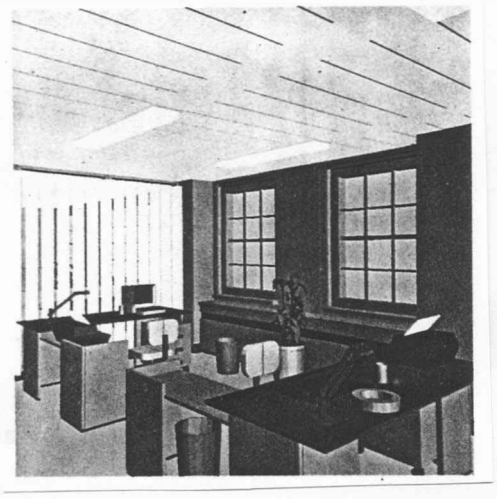
Obr. 1. 2. 11

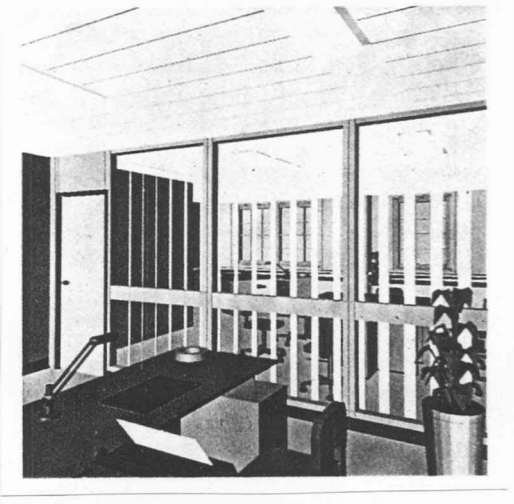
Jedno z nejznámějších center dálkového průzkumu Země je RAE Farnborough (U.K.). Téměř každodenní aplikaci počítačové grafiky lze vidět na obrazovce TV přijímače při předpovědi počasí, viz obr. 1. 2. 12. Další aplikace počítačové grafiky lze nalézt v oblasti architektury, stavebnictví apod. Na obr. 1. 2. 13 jsou ukázány výstupy z kamery Polaroid, které jsou vlastně podkladem pro účast v konkursu na vybavení místností nábytkem. Systémy tohoto druhu se dají využít např. při rekonstrukci ulic pro stavební účely a i pro čistě urbanistické studie, kdy se např. budova "zasadí" do skutečně existující krajiny.

Obr. 1. 2. 12

Kromě výše uvedených aplikací je nezbytné též upozornit na ty, které nejsou sice tak "efektní", nicméně stejně důležité. Jednou z nich je použití počítačové grafiky pro komunikaci s člověkem v systémech řízení technologických procesů a ve výrobě vůbec. Z hlediska samotné počítačové grafiky jde o "nezajímavý" problém, neboť jde o výstupy téměř statické a jednoduché, které zobrazují např. schéma zapojení v rozvodně elektrického proudu, schéma potrubního hospodářství v petrochemickém provozu apod. Do těchto základních "statických"

výstupů se pak promítají změny, např. přepnutí rozvaděčů v rozvodně s barevným vyjádřením těch částí, které jsou pod napětím apod.





Obr. 1. 2. 13

Kromě uvedených aplikací je a bude těžiště aplikací počítačové grafiky především v technických aplikacích, kdy půjde zejména o podporu konstrukčních prací. Takové systémy (někdy též nazývané CAD systémy) však neobsahují jen prostředky počítačové grafiky, které musejí být z technického hlediska velmi výkonné, ale musejí též obsahovat části pro práci s rozsáhlými databázemi a vazbu na vlastní přípravu výroby. Tyto systémy, které se připravují "na míru" určité aplikace, doznaly asi největšího použití v automobilovém, leteckém a loďařském průmyslu.

Další oblastí aplikace počítačové grafiky je oblast systémů usnadňujících publikační činnost. Jednodušší systémy se nazývají textovými procesory, resp. WP systémy (Word Processor Systems), mezi které lze zařadit WordPerfect, Chi-Writer apod. Tato třída systémů je určena pro psaní rozsahem malých publikací. Systémy typu DTP systémy (Desk Top Publishing Systems) jsou určeny k usnadnění činností nutných k vydávání rozsáhlých publikací, které mají být tištěny v kvalitě knižního tisku. Do této třídy lze zahrnout systém VENTURA, resp. TEX. Obě třídy dnes velmi uspokojivě řeší problematiku práce pouze s černobílými obrázky.

Jistě neposlední oblastí aplikace počítačové grafiky jsou nadstavby pro operační systémy, či jiné programové celky, které umožňují uživateli "příjemnou" komunikaci se systémem. Tyto systémy umožňují, aby i běžný uživatel výpočetní techniky mohl

tuto techniku využívat velmi efektivně ve své profesi. Jako příklad uveďme alespoň prostředí GEM, známé především od počítačů řady ATARI, resp. OS-Shell známý ze systémů typu

Ve všech výše uvedených aplikacích počítačové grafiky je použito barev k získání výsledného žádaného vjemu. Barva je pravděpodobně novou dimenzí, která vstupuje do oblasti vizualizace dat.

U všech systémů a periferních zařízení je zcela jasný postup stále častějšího využívání barev, přičemž rozlišovací schopnost a poskytovaná paleta barev je stále větší. U některých zařízení, např. u laserových nebo inkoustových tiskáren bude však vždy omezen počet tzv. základních barev na určitý počet. Je proto nutné nejen pochopit některé základní principy použití barev v počítačové grafice, ale i porozumět principům jejich míchání a speciálním technikám, které umožňují, aby pozorovatel získal vjem bohaté palety barev na výsledném obrázku, i když skutečný počet barev poskytovaný zařízením je velmi omezený.

2. Grafická zařízení

Grafická zařízení lze rozdělit podle mnoha hledisek či principů. Pro výklad algoritmů je vhodné rozdělit grafická zařízení takto:

- grafická zařízení
  - vektorová (paměťové obrazovky)
  - rastrová
    - se sekvenčním přístupem k bodu (zapisovače apod.)
    - s přímým přístupem k bodu (monitory apod.)

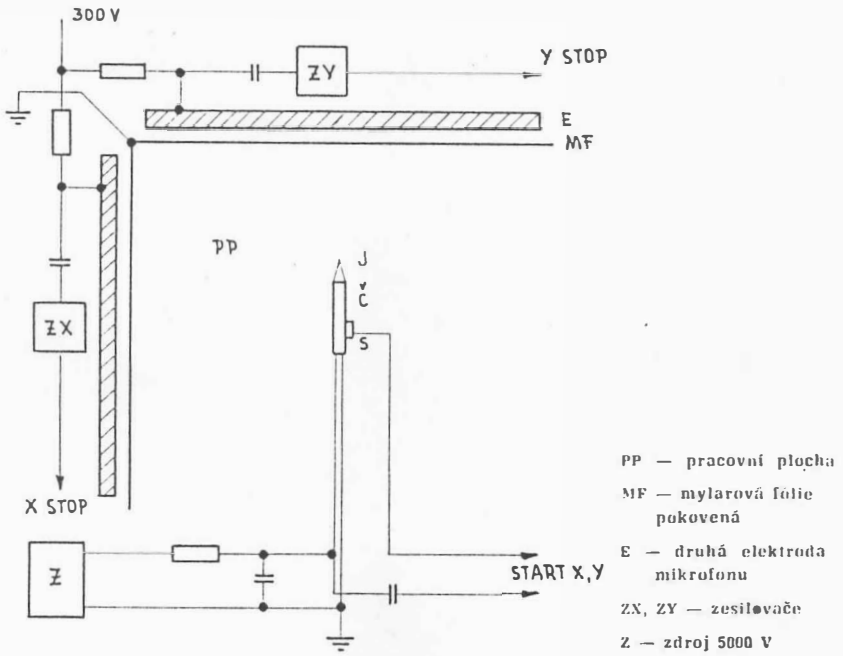
Vektorová grafická zařízení využívají analogové interpolace, zatímco rastrová grafická zařízení využívají číslicovou interpolaci. Z toho vyplývá, že u vektorových zařízení může zobrazovaná čára procházet i mimo uzlové body mřížky, jejíž hustota je určena citlivostí např. D/A převodníků.

2.1 Vstupní zařízení

Zařízení pro snímání grafické informace mohou být členěna podle různých hledisek, např. podle principu (kapacitního, magnetostrikčního, indukčního, magnetického, odporového, mechanického atd.), přesnosti, velikosti snímané plochy apod. U zařízení s velkou přesností nebo velikostí pracovní plochy se používá i systémů, které jsou vybaveny servomechanismy pro posuv mechanických dílů, které umožňují, aby operátor pouze určoval rychlost, směr a smysl pohybu a nemusel překonávat tření a momenty setrvačnosti.

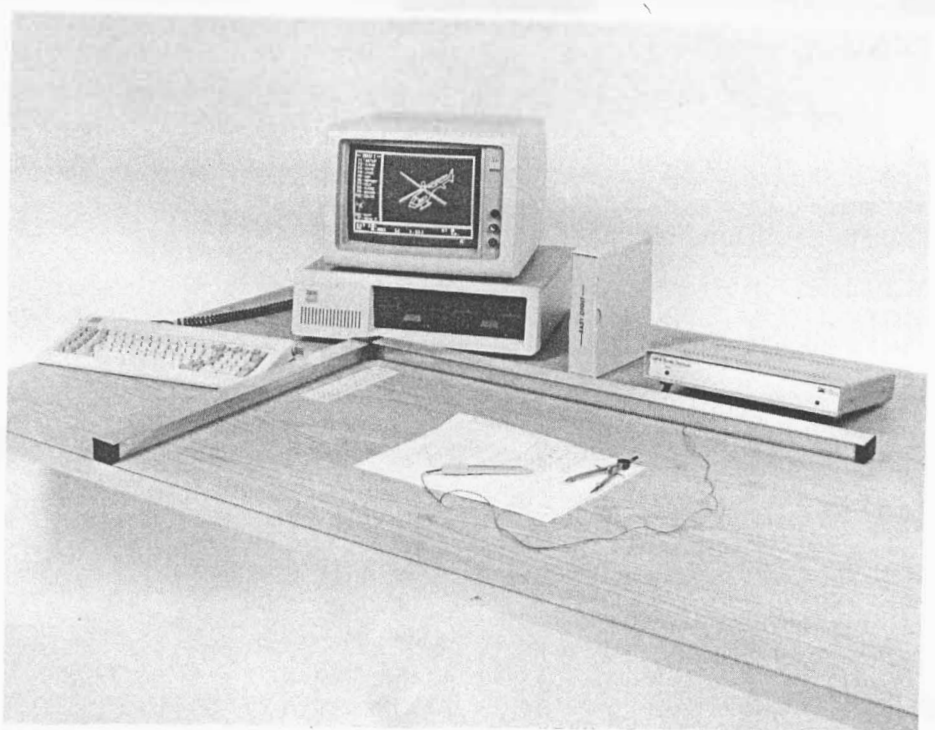
Zvukové pero

Zvukové pero je poměrně přesné zařízení, které při dobré přesnosti (až 0,1 mm) a pracovní ploše (až 1 x 1,5m) umožňuje snímání souřadnic jak v rovině, tak i v prostoru, viz [44]. Je založeno na měření času, který uplyne, než vygenerovaný zvukový impuls (z důvodů strmé náběhové hrany se používá ultrazvuk) dosáhne membrány lineárního mikrofonu, viz obr.2.1.1.a. Vzhledem k tomu, že v některých aplikacích je nutné dodržet přesnost i v neklimatizovaných prostorách, je nutné používat např. redundantní mikrofony k automatické kalibraci.



Princip zvukového pera

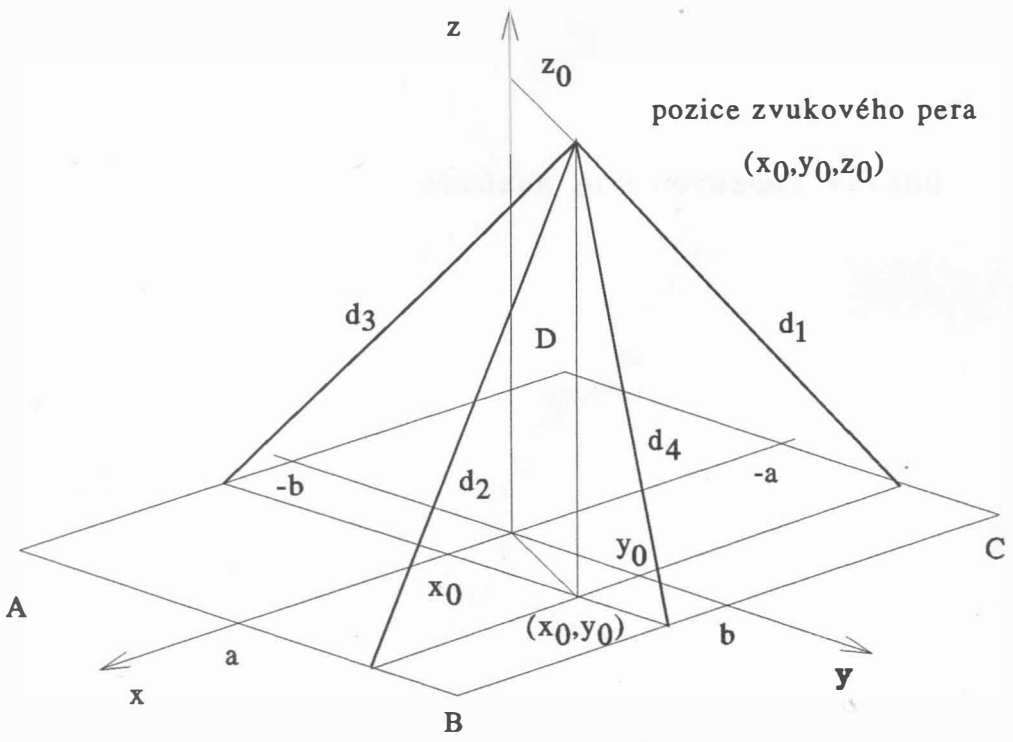
Obr. 2.1.1.a



Zvukové pero firmy Science Accessories Corp.

Obr. 2.1.1.b

I když nejjednodušší zvukové snímače používaly k odměřování souřadnic v prostoru uspořádání lineárních mikrofonů do tří ortogonálních os, je z mnoha důvodů výhodnější použití čtyř lineárních mikrofonů uspořádaných v rovině na krajích snímací plochy, viz obr.2.1.2.a.



Obr.2.1.1.a

Z obr.2.1.2.a vyplývá, že

$$z_0^2 + (a + x_0)^2 = d_1^2$$

$$z_0^2 + (a - x_0)^2 = d_2^2$$

přičemž (x_0, y_0, z_0) jsou souřadnice měřené pozice.

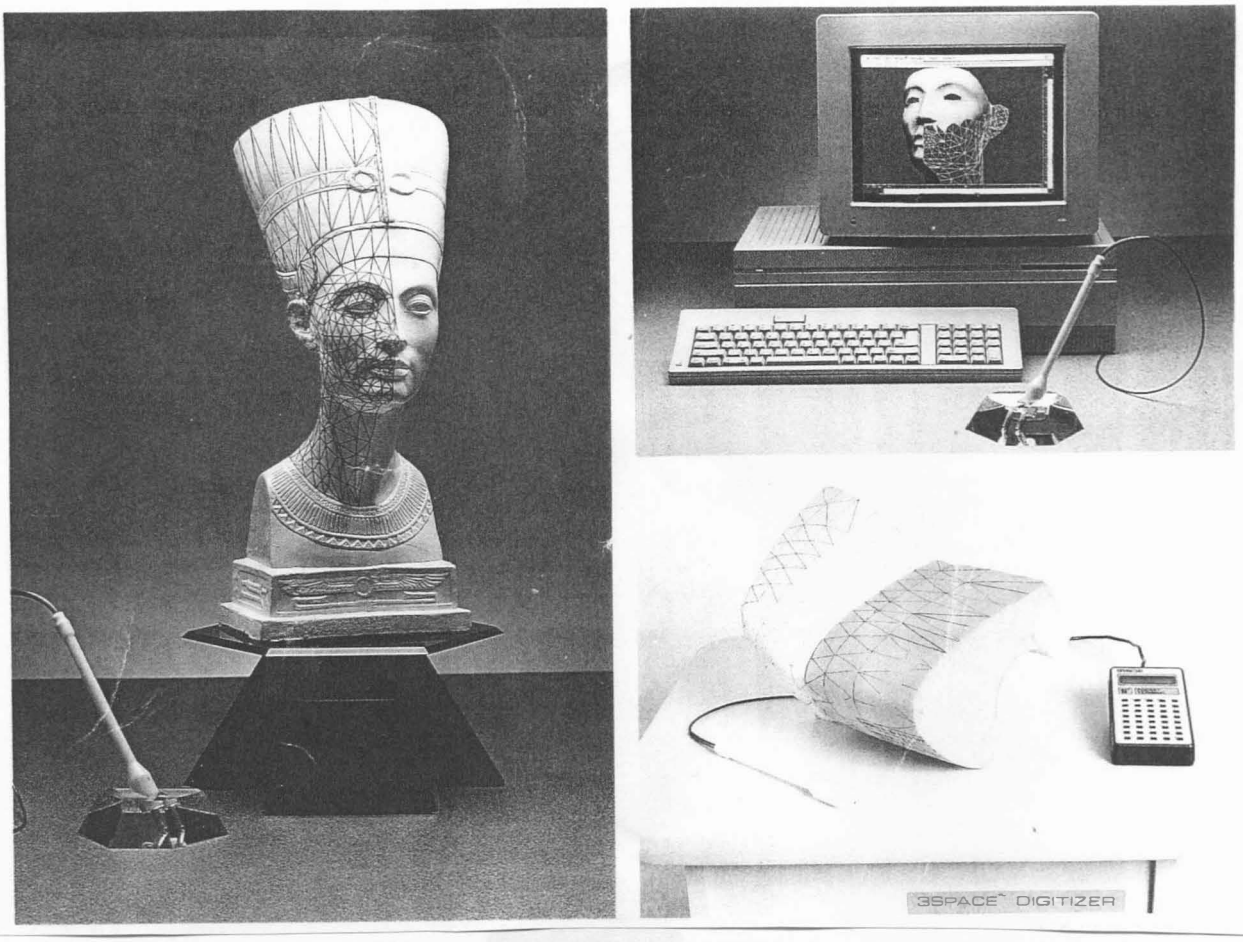
Vyjádříme-li $d_1^2 - d_2^2$, pak

$$d_1^2 - d_2^2 = z_0^2 + (a + x_0)^2 - z_0^2 - (a - x_0)^2 =$$

$$= a^2 + 2ax_0 + x_0^2 - a^2 + 2ax_0 - x_0^2 = 4ax_0$$

a tedy

$$d_1^2 - d_2^2 = 4ax_0$$



3D ultrazvukové pero

Obr. 2.1.2.b

Analogicky lze pro souřadnici y psát

$$d_3^2 - d_4^2 = 4by_0$$

Nyní je nezbytné určit souřadnici z_0 z jednotlivých rovnic, neboť platí

$$z_0^2 + (a + x_0)^2 = d_1^2 \qquad z_0^2 + (a - x_0)^2 = d_2^2$$

$$z_0^2 + (b + y_0)^2 = d_3^2 \qquad z_0^2 + (b - y_0)^2 = d_4^2$$

Pak

$$z_0^2 = [(d_1^2 + d_2^2 + d_3^2 + d_4^2) - 2 (x_0^2 + y_0^2 + a^2 + b^2)] / 4$$

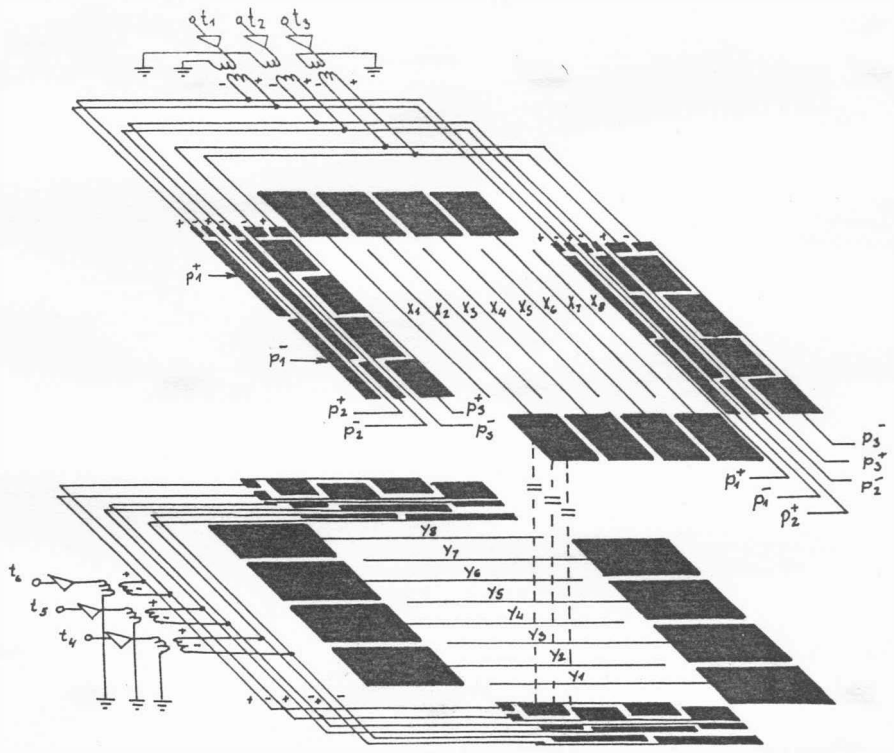
Uvedené vztahy nejenže poskytují možnosti přesného výpočtu polohy bodu (x_0, y_0, z_0), ale navíc umožňují kontrolu naměřených hodnot a případnou kalibraci zařízení. Vzhledem k tomu, že se v současných zařízeních používají mikroprocesory, není výpočet uvedených vztahů nijak problematický, a to i v případě, kdy

lineární mikrofony jsou nahrazeny bodovými mikrofony, které jsou pak umístěny v rozích snímací oblasti. Takové zařízení bylo vyvinuto v Lincoln Lab. a mikrofony byly umístěny v rozích obrazovky displaye. Na obr.2.1.1.b je pak aplikace zvukového pera pro 3D aplikace.

Rand-tablet

Tato zařízení, která se též někdy nazývají digitizéry, jsou obvykle založena na kapacitním, magnetostrikčním a elektromagneticko-indukčním principu. Některá zařízení, např. MAX firmy Terminal Display Systems Ltd., nabízejí i 6 stupňů volnosti, neboť umožňují odečítat nejen souřadnice x a y, ale i náklon ve směru osy x, y a též pootočení ukazovátka a tlak, kterým na něj operátor působí.

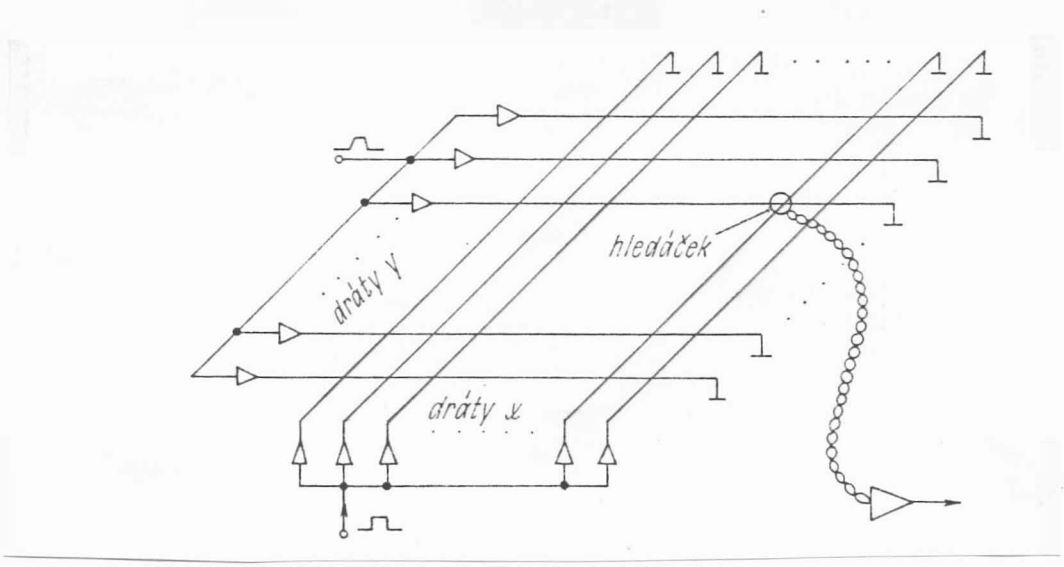
Zařízení založené na **kapacitním principu** je zobrazeno na obr.2.1.3.



kapacitní princip
Obr.2.1.3

Pracovní plocha je zhotovena z tenké fólie, na kterou jsou naneseny elektricky vodivé obrazce. Na jedné straně je soustava vodičů rovnoběžných s osou x, na straně druhé je pak soustava vodičů rovnoběžných s osou y. Nad těmito soustavami se pohybuje čidlo, které kapacitně přijímá impulsy od souřadnicových vodičů, které jsou připojeny přes kódovací zařízení na zdroj impulsů z čítačů. Ke kódování se používá většinou Grayův kód. Na ploše 30x30 cm lze dosáhnout rozlišení asi 1024x1024 bodů.

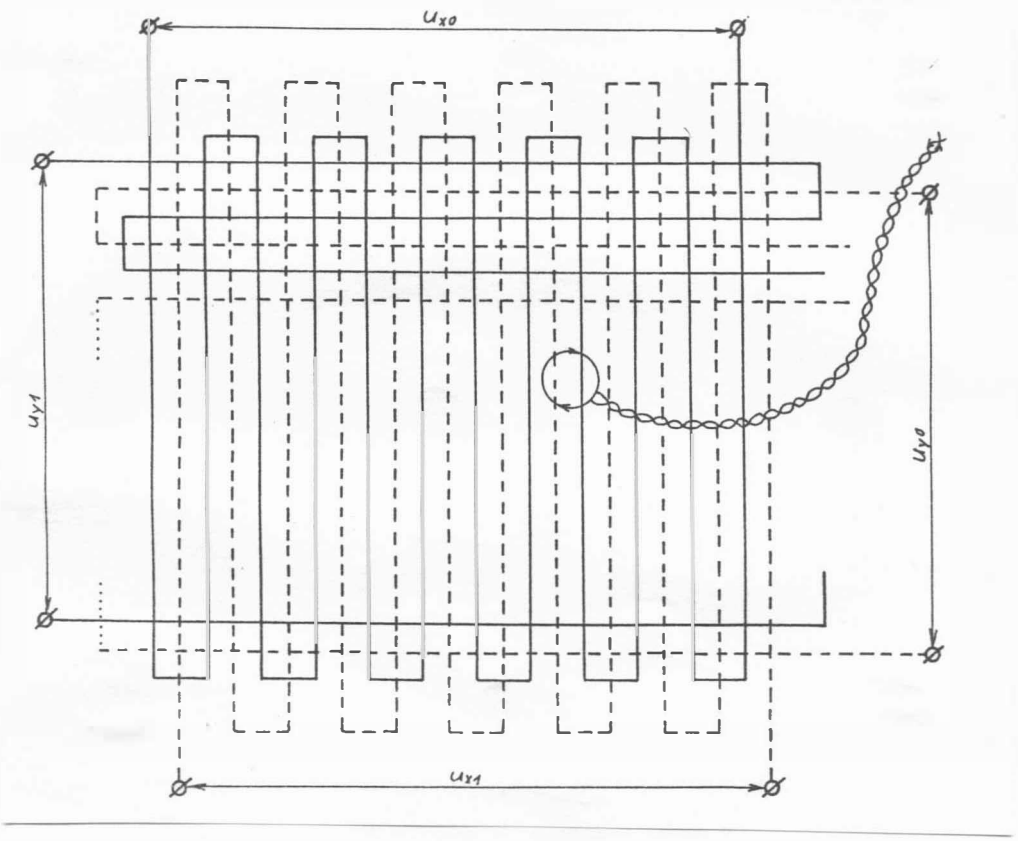
Princip magnetostrikce, který je používán u snímačů, je ukázán na obr.2.1.4 a byl používán především u ultrazvukových magnetostrikčních linek. Princip je založen na pružné mechanické deformaci magnetického materiálu, která je vyvolána magnetickým polem. Je-li do levé cívky vyslán proudový impuls, vzniká mechanická deformace magnetického drátu, která se šíří podél drátu rychlostí asi 5 km/s a v místech, jimiž prochází, způsobuje přechodnou změnu okamžité hodnoty permeability. Při průchodu deformační vlny okolím permanentního magnetu vyvolá změna permeability drátu změnu indukce magnetického toku procházejícího snímací cívkou, čímž je na cívce indukován napěťový impuls. Hodnota příslušné souřadnice je pak určena dobou mezi vysláním impulsu a jeho detekcí. U těchto zařízení lze dosáhnout při pracovní ploše 1x1,5 m přesnosti asi 0,1 mm.



Princip magnetostrikce

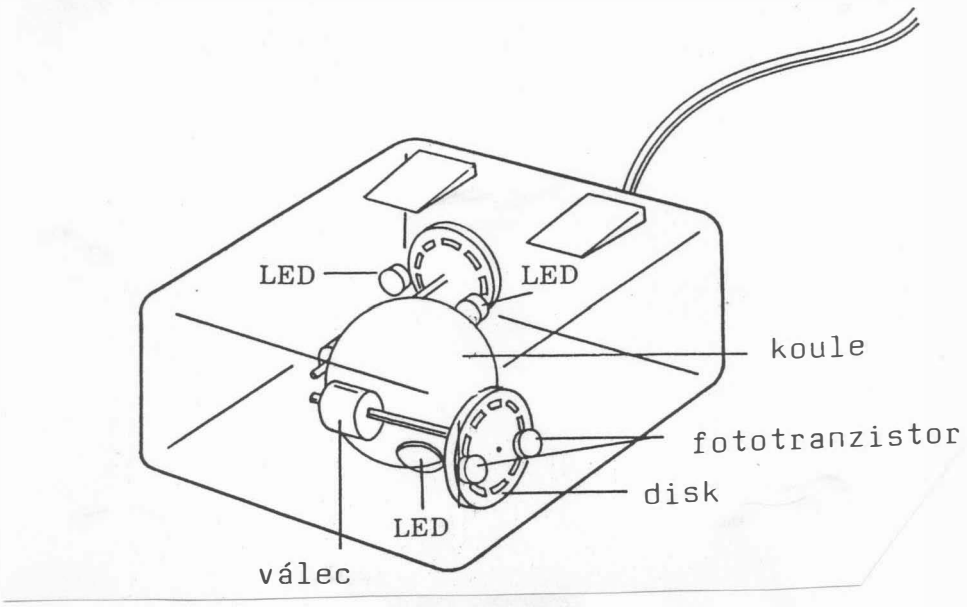
Obr.2.1.4

Odčítání s větší přesností (až 0,02 mm) je umožněno **elektromagneticko-indukčním principem**. Ve snímacím zařízení je umístěna kruhová cívka a v pracovní ploše jsou pak umístěna čtyři meandrová vedení s konstantní roztečí. Pro každou souřadnici jsou dvě meandrová vedení a ta jsou navzájem posunuta o čtvrtinu rozteče, viz obr.2.1.5. Cívka ve snímači je obvykle buzena střídavým proudem se sinusovým průběhem a měří se indukovaná napětí v soustavě meandrových vinutí. Měření změny souřadnice v rámci jedné rozteče se provádí analogově, počet roztečí, o něž se snímací zařízení posune, je registrován obousměrnými čítači.



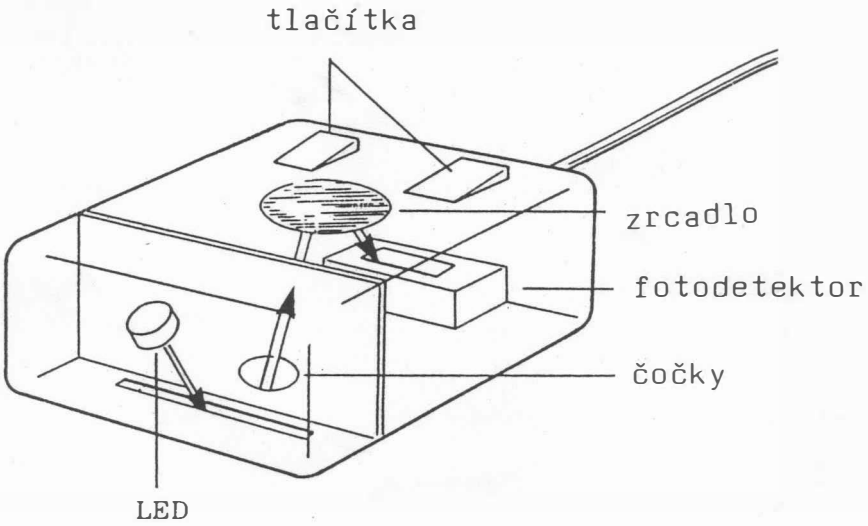
el.mag-indukční princip

Obr.2.1.5



Myš mechanická

Obr.2.1.6.a



Myš optoelektronická

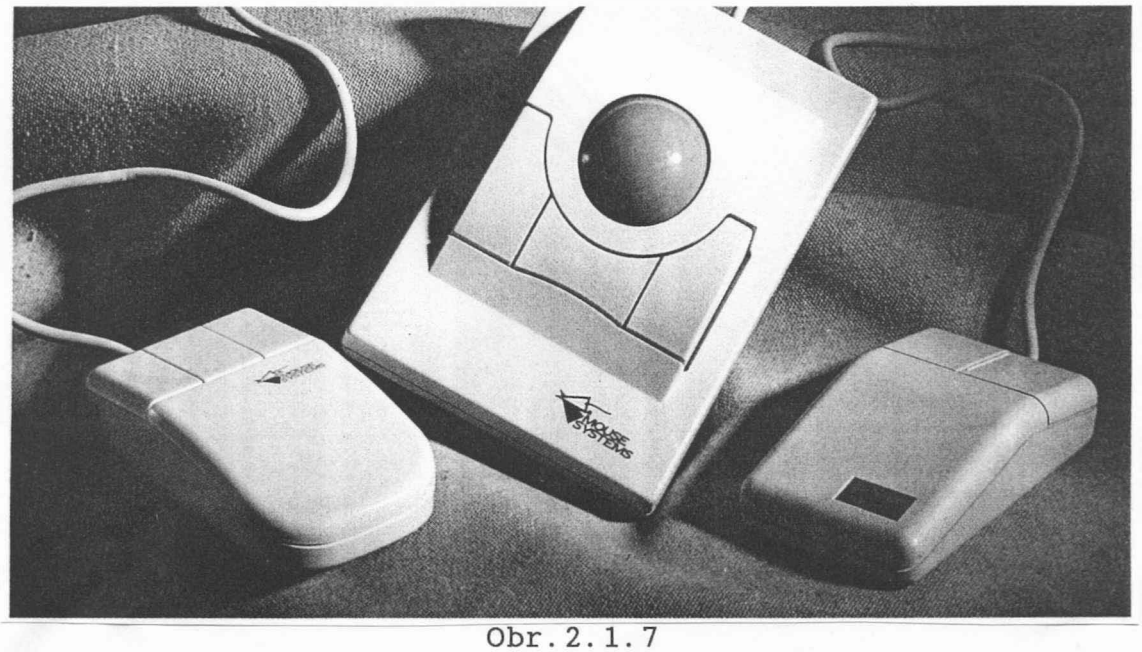
Obr.2.1.6.b

Myš

Toto zařízení, které je založeno buď na **mechanickém**, nebo **optoelektrickém** principu, se používá zejména ve spojení s osobními počítači a umožňuje rychlou komunikaci operátora s výpočetním systémem. Pohyb myši je přenášen na obrazovku a slouží především k ovládání polohy kurzoru na obrazovce. U mechanického principu se pohyb myši převádí na odvalování mechanické kuličky, jejíž otáčení se pomocí mechanického převodu rozkládá do dvou kolmých směrů. Tento pohyb se pak registruje pomocí čítačů, viz obr.2.1.6. U optoelektrického principu se používá pro každou souřadnou osu zdroj světla a optické čidlo, které registruje, zda byl či nebyl paprsek, který se odráží od speciální podložky, přerušen. Přerušení je způsobeno barevnou mřížkou, pro jeden směr je použita vždy jedna barva. Zdroje světla vyzařují nebo čidla registrují jen určitou vlnovou délku, která je opět jiná pro každý směr, viz obr.2.1.6.a.

Kulový ovladač

Na podobném mechanickém principu jako mechanická myš je řešen i kulový ovladač. Jde vlastně o obrácenou myš vzhůru dnem, viz obb.2.1.7. Operátor pak ovládá přímo pohyb koule, zatímco zařízení polohu nemění. U některých systémů bývají místo kulového ovladače použity dva potenciometry s osami na sebe kolmými.



Obr.2.1.7

Pákový ovladač

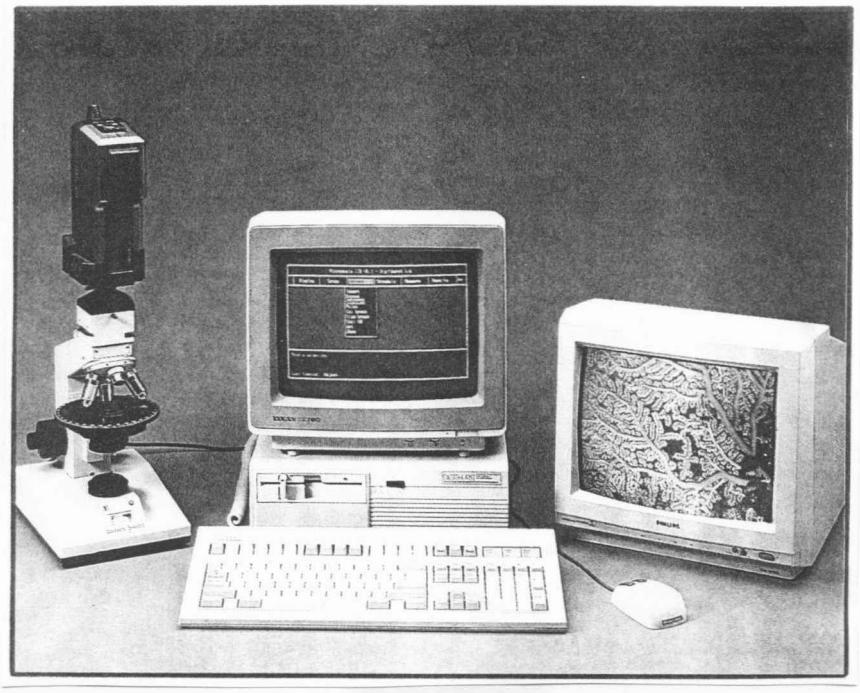
Pákový ovladač je velmi rozšířeným zařízením zejména u domácích počítačů. Jde o zařízení, které převádí výchylku do jednotlivých směrů buď na analogový signál, který je úměrný výchylce v daném směru, nebo na impuls, pokud vychýlení je takové, že dojde k sepnutí koncových spínačů. Některé typy umožňují pootočením osy páčky ovladače zadávání třetí souřadnice. Pákového ovladače se s výhodou používá k nastavení počáteční polohy u některých zapisovačů a grafických displayů. K masovému rozšíření pak došlo zejména ve spojení s domácími počítači a herními automaty .

Světelné pero

Kromě výše uvedených zařízení se k určení pozice na obrazovce používá též světelné pero, které pracuje na optickém principu. Optický detektor, který je buď přímo v ukazovátku, nebo mimo něj s přívodem optické informace pomocí optického vlákna, převádí optický záblesk, který vzniká při opětovné aktivaci světelného bodu na obrazovce, na signál, z něhož se odvodí pozice snímaného bodu. Kromě vlastního optického snímače má světelné pero spínač oznamující platnost výběru. Spínač může být řešen mechanickým nebo kapacitním způsobem. Nevýhodou světelného pera je menší přesnost, zakrytí žádané pozice vlastním perem a v neposlední řadě i únava ruky operátora při delší práci.

Zařízení pro vstup obrazové informace

Zařízení pro vstup obrazové informace je většinou tvořeno kamerou, resp. videopřehrávačem, a převodníkem, který převádí analogový video signál na číslicovou formu vhodnou k dalšímu zpracování na počítači, viz obr.2.1.8. Rozvoj prvků CCD podstatně zjednodušil tento typ zařízení, při zvětšení rozlišovací schopnosti (660 x 480 bodů s 24 bity pro odlišení barvy na jeden bod). Tato zařízení slouží především v oblasti zpracování obrazových dat v aplikacích, kdy musí docházet k složitému zpracování obrazové informace, např. v oblasti nasazení robotů, dálkového průzkumu Země, optického čtení písma a jeho následného rozpoznávání apod.

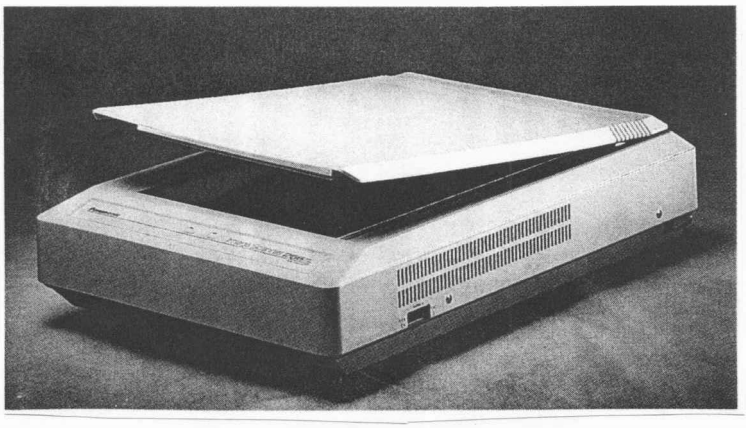


Kamera připojená k PC-386 firmy Digithurst Ltd.
Obr.2.1.8

Podobného principu využívají i tzv. **scannery**, viz obr.2.1.9. V tomto případě se dosahuje vyšší hustoty snímaných bodů (až 800 dpi, resp. 7.7 bodů/mm) i při velkých formátech snímaných dokumentů, a to až formátu A0. Nejčastěji se používají formáty A4 / A3, kdy při hustotě 400 dpi může být nasnímáno teoreticky až 16 miliónů barev, např. scanner FS1-S firmy RICOH. Pro reprezentaci barev se používá RGB systém s tím, že signál odvozený od každé základní barvy může mít až 256 úrovní. Pro každou základní barvu jsou pak ve scanneru použity oddělené snímací prvky CCD. Nezbytnou částí systémů používající scannery je pak programové vybavení zajišťující převod nasnímaných informací z rastrového do vektorového tvaru.

Scanner formátu A0 firmy INTERGRAPH

Obr. 2.1.9



Scanner formátu A4 firmy Panasonic

Obr. 2.1.10

2.2 Výstupní zařízení

Výstupní grafická zařízení lze opět dělit do skupin podle principu, přesnosti a rychlosti kreslení, velikosti kreslicí plochy, opakovatelnosti kresby apod. Převážná většina výstupních grafických zařízení je ve své podstatě rastrového typu, i když se z vnějšího hlediska uživatele chovají jako vektorová, např. kreslicí stoly, neboť velikost fyzického kroku je uživatelem nepozorovatelná.

Tiskárny

Nejčastěji používaným zařízením pro grafický výstup byla a stále je tiskárna. V počátcích rozvoje počítačové grafiky byly k dispozici pouze tiskárny alfanumerické, které umožňovaly tisk jen jednoduchých grafů a tabulek. I když potřeba rychlých výstupních zařízení způsobila, že byly vyvinuty laserové tiskárny, výstup grafických informací byl stále velmi omezen. Teprve s velkým pokrokem v polovodičové technologii, který umožnil výrobu velkých pamětí za přijatelnou cenu, nastal rozvoj tiskáren s možností grafického výstupu. Jako první se v masovém měřítku začaly používat tzv. mozaikové tiskárny, které umožňovaly tisk alfanumerických znaků v rastru 8 x 8 nebo 24 x 24 bodů a výstup grafických informací.

Elektrostatické tiskárny

Elektrostatické tiskárny jsou tiskárny, jejichž rozvoj je spjat především s rozvojem integrace pamětí. Tyto tiskárny lze v zásadě rozdělit takto:

- elektrostatické tiskárny
  - suchý proces
    - laserové - typ XEROX
    - termografické
  - mokrý proces
    - inkoustové tiskárny
    - typ COSTAR

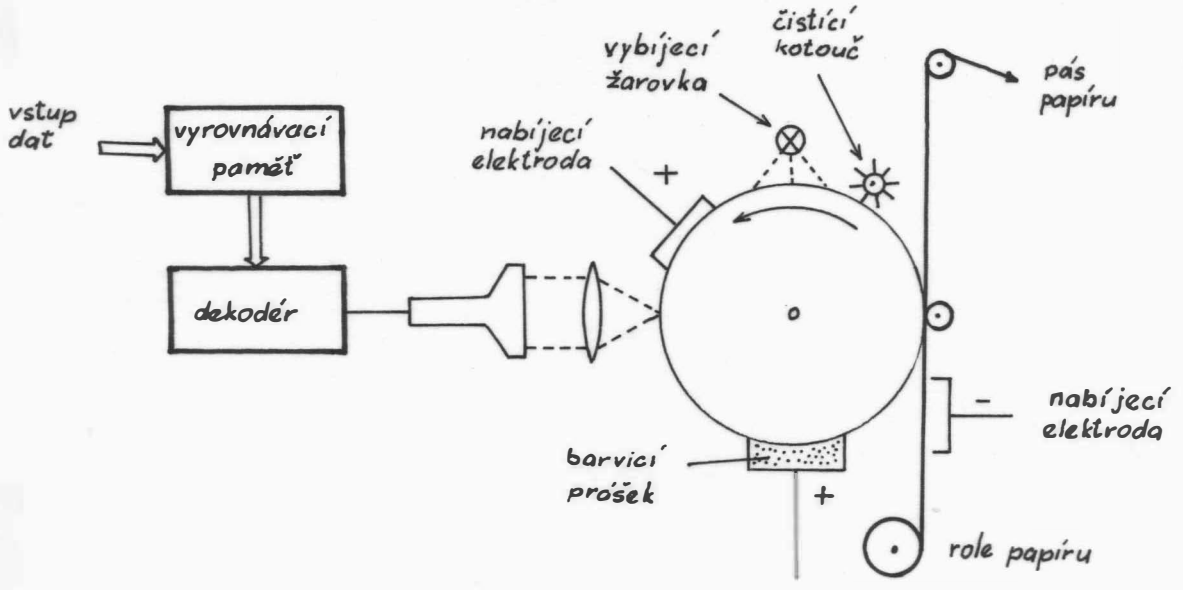
Jistou nevýhodou všech elektrostatických výstupních zařízení je nutnost vytvoření binárního obrazu v paměti a jeho následný výstup na fyzické médium, což klade velké nároky zejména na paměť vlastního zařízení nebo systému. V případě, že je

využívána paměť systému, kladou se též velké nároky na přenosovou cestu ze systému do zařízení. Uvážíme-li, že obraz má rozměr 8192 x 8192 bodů, což není příliš mnoho, chceme-li dosáhnout přijatelné kvality výstupu, pak je nutná paměť o kapacitě 8 MB.

Někdy se pod pojmem elektrostatické tiskárny označují pouze elektrostatické tiskárny (ať už založené na suchém či mokrém procesu), které mají záznamovou hlavu (většinou uspořádanou jako řadu elektrod), která nabíjí dané médium.

Laserové tiskárny

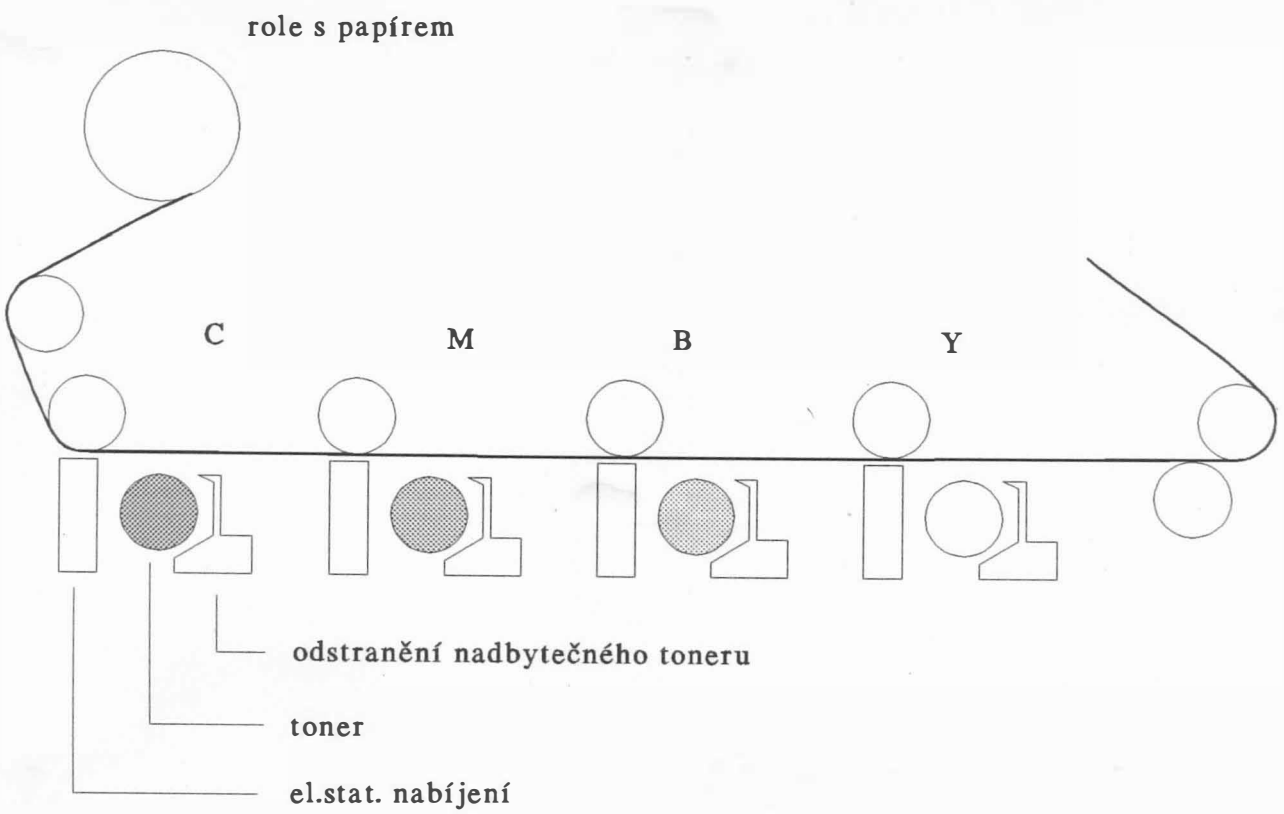
Jsou založeny na obdobném principu jako rozmnožovací stroje typu Xerox, kdy se selénový válec pomocí laserového paprsku osvětluje. Osvětlené části se stanou elektricky vodivými a jejich náboj se odvede přes kovové části. Na tyto části se pak přichytí kladně nabité barvivo (toner), které se přenáší na záporně nabitý papír. Barvivo se poté musí tepelně stabilizovat. Současné laserové tiskárny dosahují hustoty až 600 bodů/palec (dpi-dot per inch). Princip laserové tiskárny je znázorněn na obr. 2.2.1.



Princip laserové tiskárny
Obr. 2.2.1

Tiskárny se záznamovou hlavou

Tyto tiskárny mají záznamovou hlavu, která umožňuje nabíjení média v jednotlivých bodech. Na obr.2.2.2 je ukázán princip pro vícebarevný tisk s použitím suchého procesu. Tiskárny mohou používat buď mokrý proces (typ COSTAR - barvivo se nezapéká, ale použitá vývojka s barvivem se musí odsát a papír osušit), nebo suchý. U tohoto typu tiskáren se většinou používá role papíru. Typickým výrobcem tiskárny se suchým procesem je firma VERSATEC, zatímco tiskárnu s mokrým procesem vyrábí ARITMA pod označením EC 7140.



Princip vícebarevné tiskárny používaný firmou VERSATEC
Obr.2.2.2

Termografické tiskárny

Jde opět o elektrostatickou tiskárnou se záznamovou hlavou s tím rozdílem, že papír je napuštěn nebo pokryt vrstvou speciální látky, která buď mění barvu účinkem tepla, které vzniká při nabíjení, nebo v případě barevného tisku se barva teplem přenáší na bílý papír, přičemž se využívá barevného přetisku. Např. tiskárny firmy Shinko Electric Co. dosahují hustoty 300 dpi a používají 7 barev. Při použití vzorů nebo ditheringu je možné docílit až 4096 barev. Rychlost tisku je asi 90 sec pro formát A4. Kapacita použité vnitřní paměti je až 6 MB.

Inkoustové tiskárny

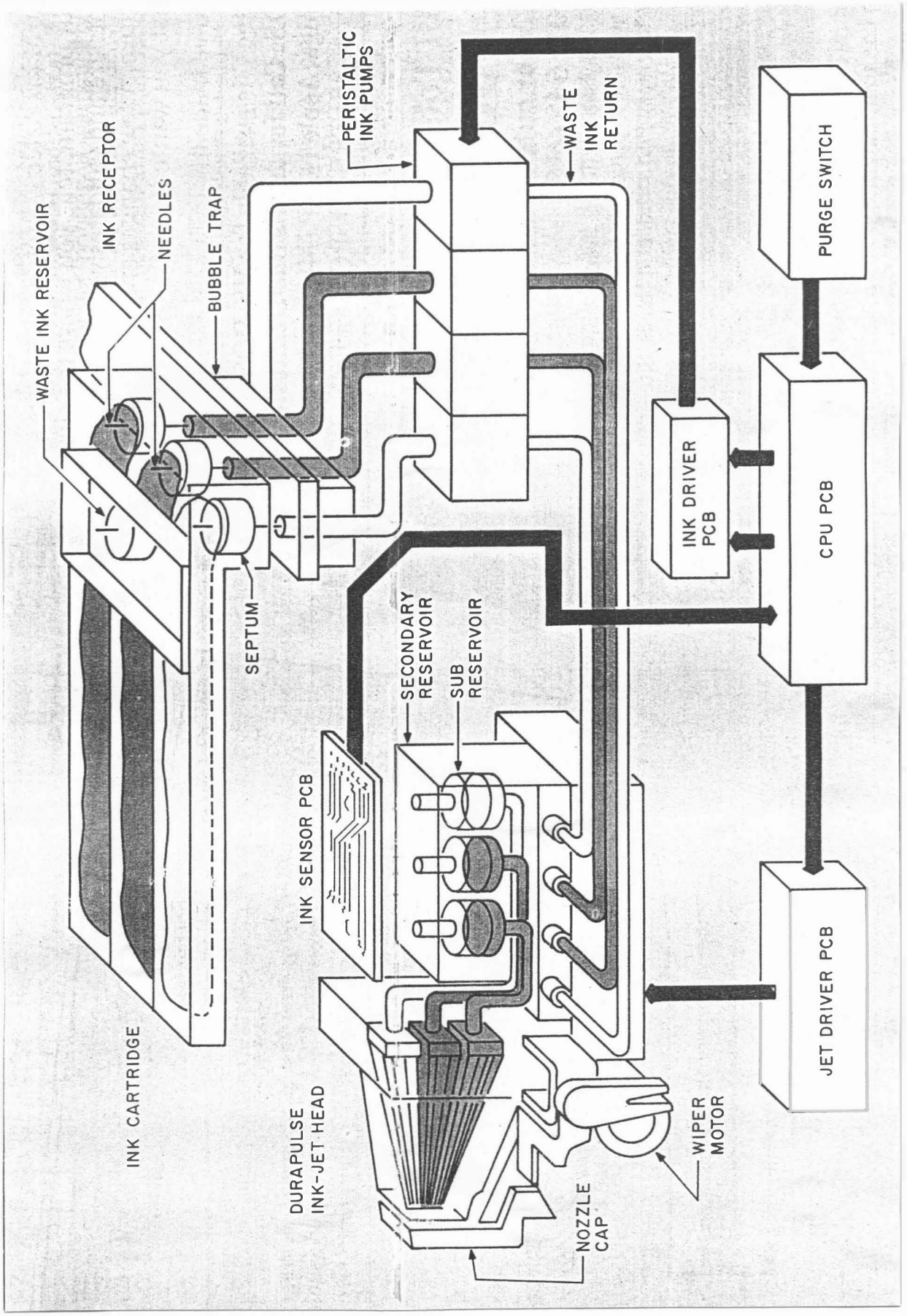
Inkoustové tiskárny lze rozdělit na dvě základní skupiny, a to:

- se stálým paprskem - proud malých kapiček (typicky 50 000/s) buď dopadne na papír, nebo je odkloněn pomocí elekrostatickéko náboje,
- kapičky jsou vystřikovány na základě požadavku ve vhodný časový okamžik.

Princip inkoustové tiskárny je na obr.2.2.3. Je vhodné poznamenat, že tiskárny obvykle mají i černou barvu z důvodů ekonomičtějšího provozu, i když je možné černé barvy docílit složením tří barev. Tímto způsobem se navíc docílí lepší sytosti černé barvy.

Kreslicí stoly

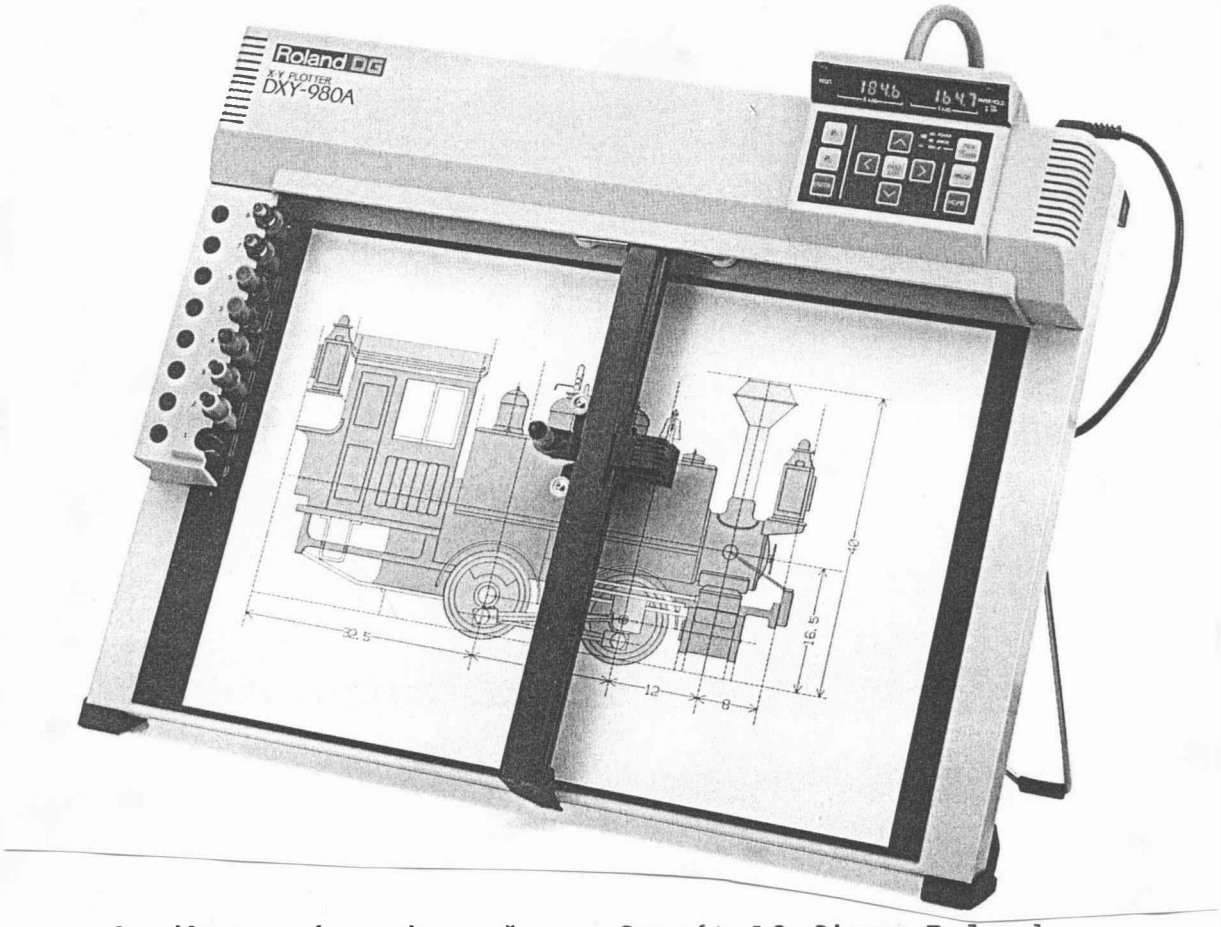
Kreslicí stoly mají většinou tvar desky formátu A0 nebo větší, přičemž vzhledem k požadované přesnosti kresby 0,01 mm je nutná poměrně robustní mechanická konstrukce, což způsobuje pomalejší kresbu z důvodu zvětšených momentů setrvačnosti. Typickým představitelem je např. DIGIGRAF 1208-4G. Tato zařízení se používají s různými technologickými hlavami pro vytváření různých předloh, např. masek integrovaných obvodů.



Zjednodušené schéma inkoustové tiskárny

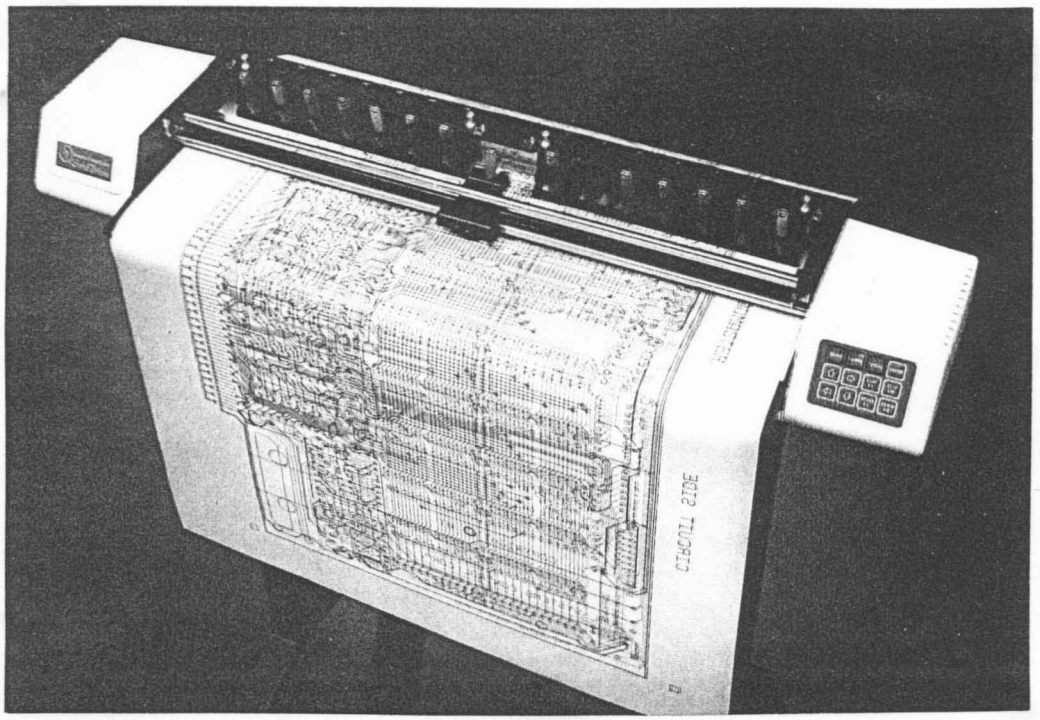
Obr. 2.2.3.a

Inkoustová tiskárna pro tisk velkých formátů



Osmibarevný zapisovač pro formát A3 firmy Roland

Obr. 2. 2. 4



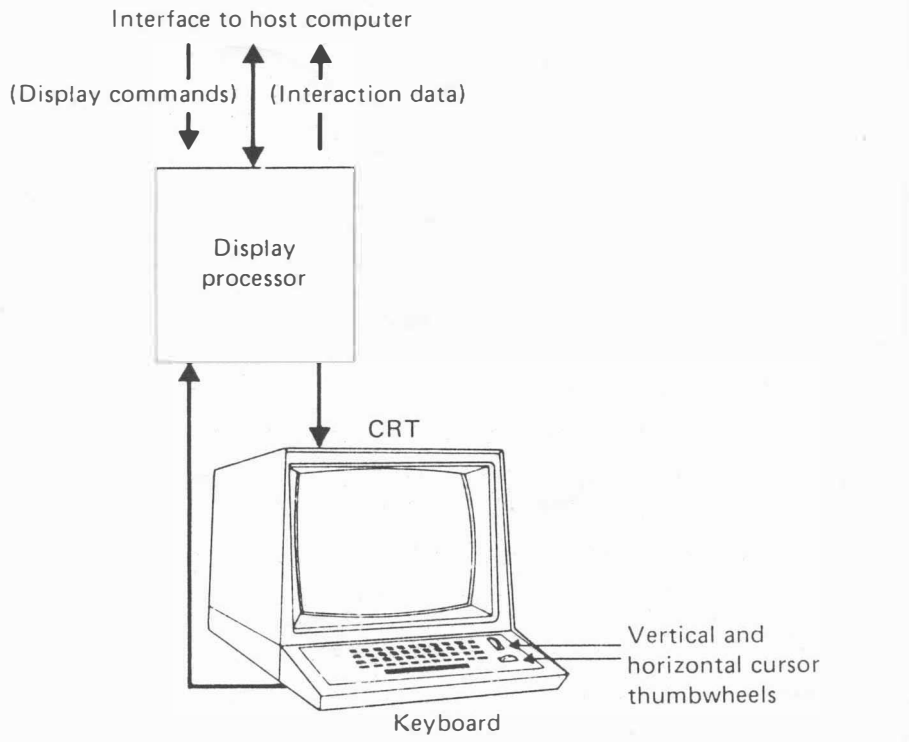
Čtrnáctibarevný zapisovač formátu A1 firmy Houson Instruments
Obr. 2.2.5

Zapisovače

Zapisovače tvoří třídu výstupních zařízení, která se vyznačuje především velkou rychlostí kreslení vektorů (jde o zařízení se sekvenčním přístupem k bodu) při velké ploše kresleného motivu a dobré přesnosti. Tato zařízení jsou vhodná pro většinu výstupů v technických aplikacích. První zapisovače neměly původně interpolátor, avšak s rozvojem integrace byly postupně vybavovány nejen interpolátory, ale i mikroprocesory zajišťujícími komunikaci na vyšší úrovni, např. v jazyce BGL nebo HPGL. Zapisovače dnes mají dvojí podobu, a to buď "zmenšených" kreslících stolů na formát A3 nebo A4, viz obr. 2.2.4 (ROLAND, COLORGRAF-ARITMA), nebo válcových zapisovačů s papírovou rolí se šířkou až 2 m, viz obr. 2.2.5 (CalComp).

Displaye s paměťovou obrazovkou

Display s paměťovou obrazovkou patří k zařízením vektorového typu, kdy úsečka zobrazovaná na stínítku je vytvářena tak, že na vychylovací soustavu paprsku se přivádí většinou lineárně rostoucí napětí. Paprsek pak na stínítku zobrazí úsečku. V době, kdy cena pamětí byla velmi vysoká, byla vyvinuta obrazovka, která měla schopnost po dlouhou dobu zachovat informaci, jež byla na ní zobrazena. Využívala k tomu tzv. sekundární emisi, která umožňuje, aby obraz, zaznamenaný na mřížce před luminoforem elektrostatickým nábojem, byl vlastně neustále obnovován. Tento typ obrazovek umožňoval vyrábět laciné grafické displaye, které měly jedinou nevýhodu, a to, že v případě opravy kresleného motivu musel být celý obraz smazán a znovu překreslen. Typickým výrobcem těchto displayů byla firma TEKTRONIX. V současné době se paměťové obrazovky nepoužívají vzhledem k prudce klesající poměru ceny pamětí a její kapacity.

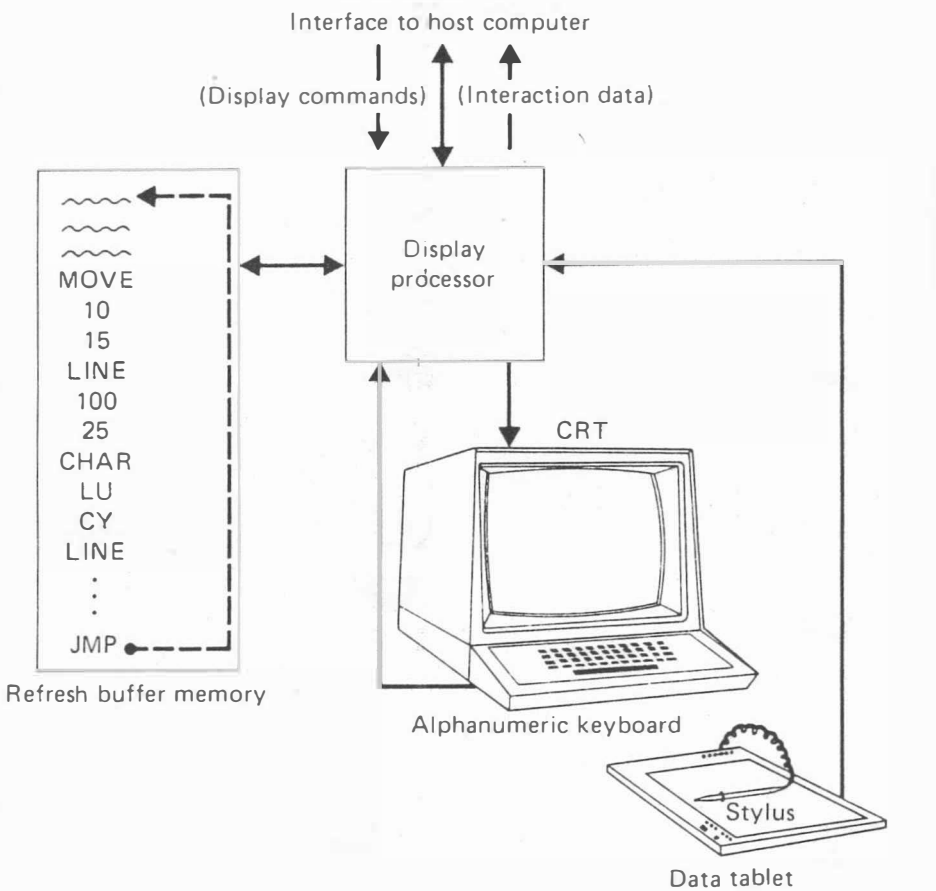


Schematické uspořádání displaye s paměťovou obrazovkou

Obr. 2.2.6

Displaye s obnovovaným obrazem

Rozvoj elektroniky a pokles ceny pamětí, spolu s novými požadavky kladenými na grafická výstupní zařízení, způsobily vývoj a následnou výrobu grafických displayů s obnovovaným obrazem. Tato zařízení vlastně byla založena na principu neustálého obnovování obrazu na základě interpretace tzv. display souboru, který reprezentoval zobrazovanou informaci a umožňoval též základní operace vkládání, rušení apod. Zařízení založená na tomto principu byla použita i v grafickém systému SM 52/11-ISAP. Výhodou těchto displayů je to, že je možné část obrazu vymazat pouhým zrušením nebo speciálním označením příslušných dat a příkazů v tzv. display souboru, který je uložen v lokální paměti displaye.

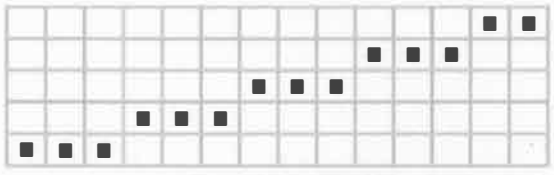


Schematické uspořádání displaye s obnovovaným obrazem

Obr. 2. 2. 7

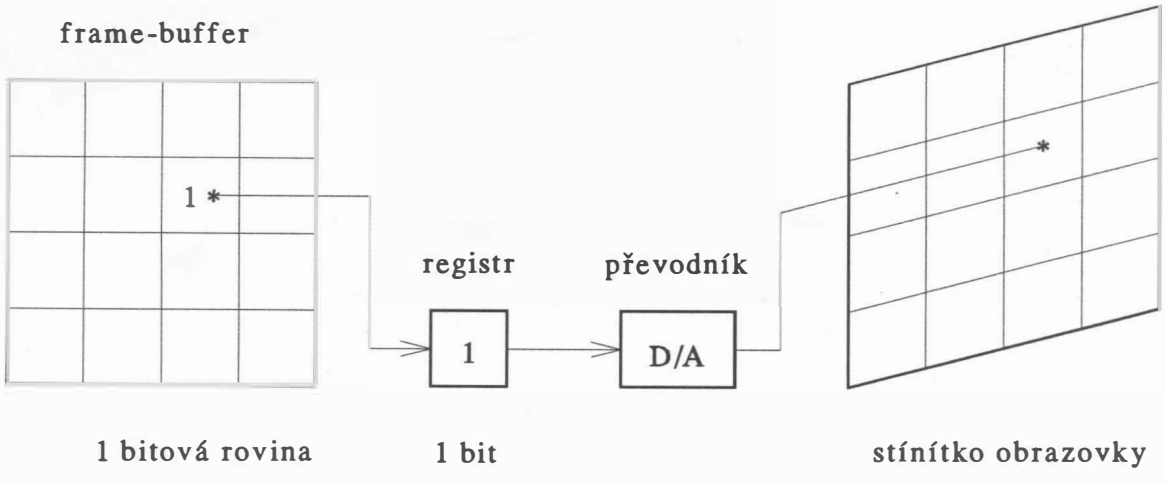
Rastrové displaye

Jedním z nejčastěji používaných grafických zařízení je grafický display rastrového typu. Tato zařízení jsou vesměs založena na zobrazování části paměti, která se nazývá většinou video-paměť nebo frame-buffer.



Obr. 2. 2. 8

Na obr. 2. 2. 8 je znázorněn rozdíl mezi úsečkou kreslenou na vektorovém zařízení, tj. zařízení, které zobrazuje čáry, a na rastrovém zařízení. U rastrového zařízení je úsečka aproximována řadou jednotlivých elementů obrazu, které se nazývají pixely. Nejčastěji používanou realizací pro grafické displaye rastrového typu (dále jen displaye) je tzv. frame-buffer. Na obr. 2. 2. 9 je zobrazen princip displaye s jedinou bitovou rovinou frame-bufferu. Toto umožňuje zobrazování bodu s jedinou barvou, resp. úrovní šedi. **Rozlišovací schopností** pak rozumíme údaj, který uvádí, kolik bodů je zobrazitelných v ose x a v ose y, zatímco **adresovatelnost** je údaj o rozměru bitové roviny (nebo bitových

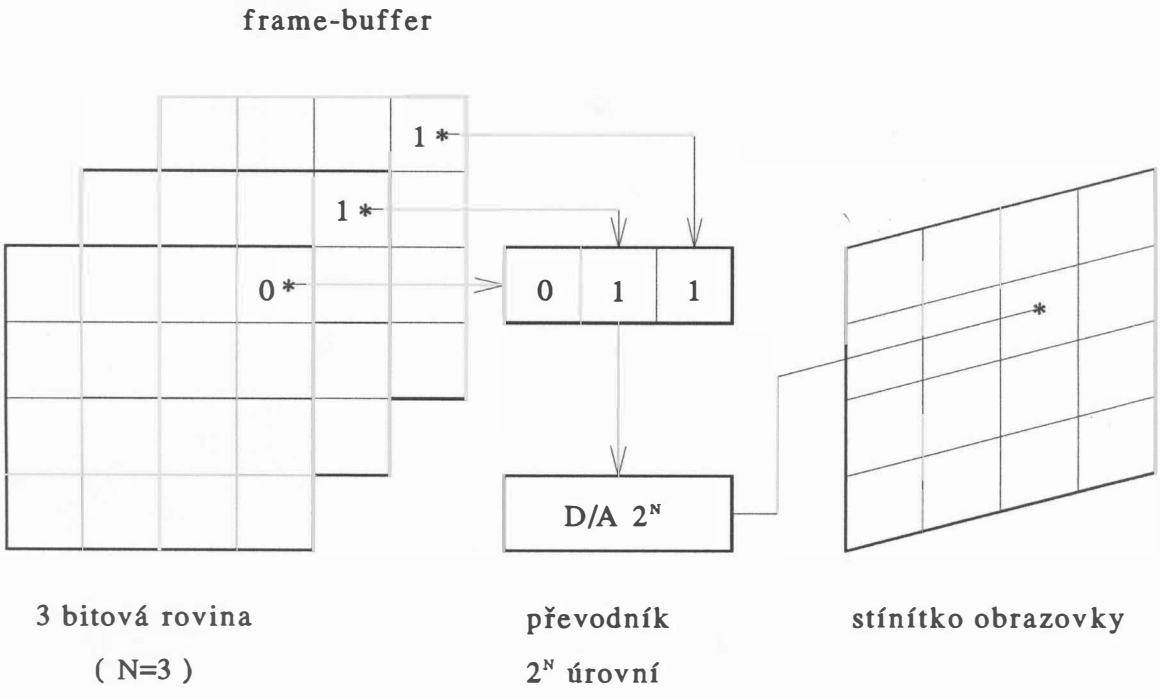


Frame buffer s jednou bitovou rovinou

Obr. 2. 2. 9

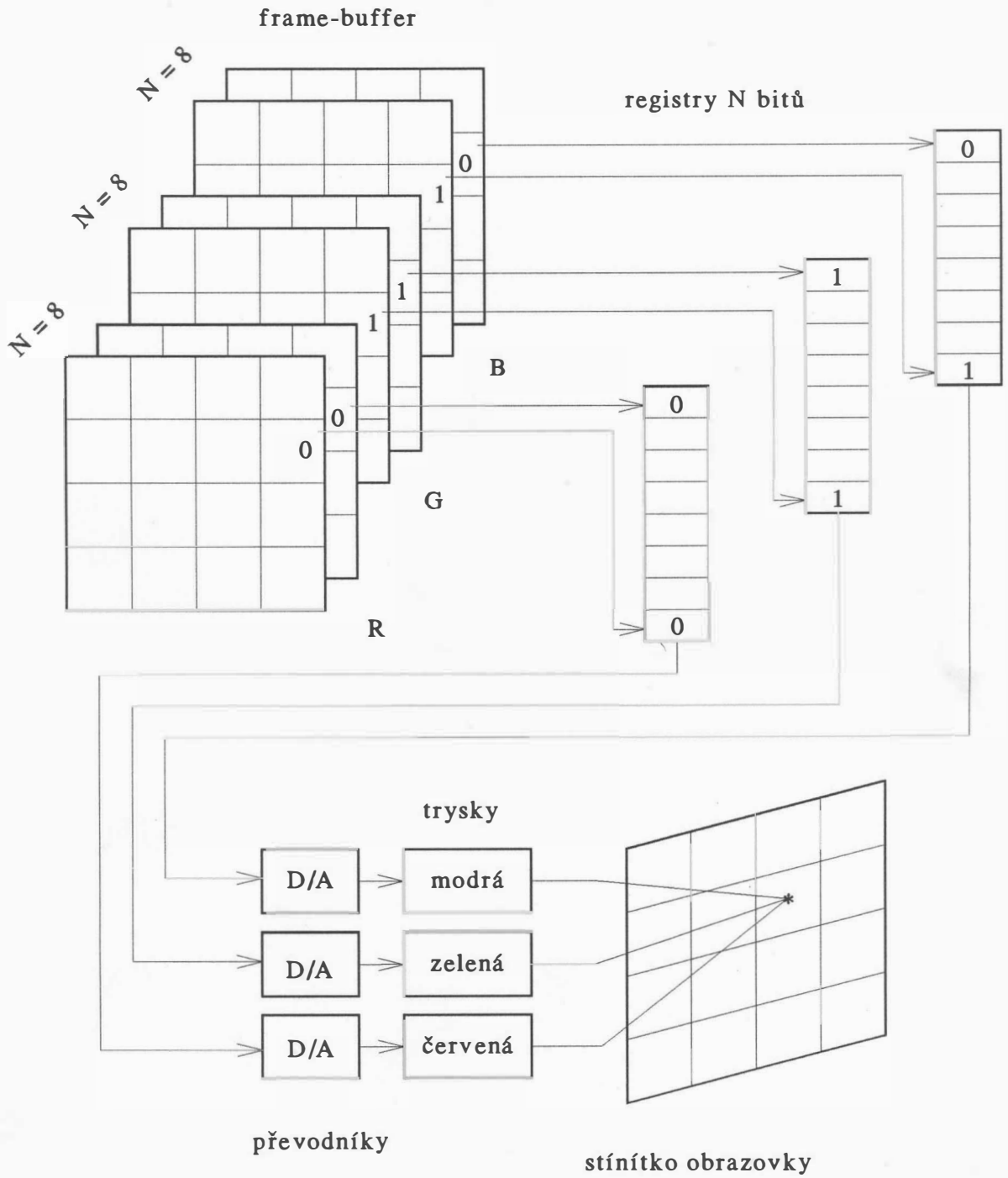
Rozlišitelnost a **adresovatelnost** jsou obecně dva **rozdílné** údaje, které by neměly být zaměňovány. V dalším výkladu se pro jednoduchost přidržíme toho, že rozsah frame-bufferu je stejný jako rozsah zobrazení na obrazovce, tj. rozlišitelnost a adresovatelnost jsou stejné.

Pro adresovatelnost 1024 x 1024, která je nezbytná pro konstruktérskou praxi, dostáváme kapacitu paměti 1Mb. V určitých aplikacích je nezbytně nutné mít několik úrovní šedi a obvykle je počet úrovní šedi vyjádřen číslem 2^N, tj. v rozsahu 0 až $2^N - 1$, kde **N** je počet bitových rovin. Princip takového displaye je zobrazen na obr.2.2.10. V daném případě se bod na obrazovce zobrazí s úrovní šedi odpovídající hodnotě 3, přičemž je možné v daném případě mít osm různých intenzit současně na výstupu.



Frame buffer se třemi bitovými rovinami
Obr.2.2.10

Pro některé aplikace je vhodné mít více **možných** úrovní šedi, než je aktuální kapacita frame-bufferu. V takovém případě se používá ještě tabulka, která umožňuje překódování a nazývá se překódovací tabulka (look-up table), viz obr.2.2.11.



Frame buffer s překódovací tabulkou

Obr. 2.2.11

Uvedený princip umožňuje snadnou změnu dané úrovně šedi všech pixelů na novou úroveň přepisem obsahu příslušné řádky překódovací tabulky místo překódování všech příslušných pixelů na novou hodnotu. Tabulka pochopitelně nesmí být v tomto případě realizována pamětí typu ROM, ale pamětí typu RAM.

Je zřejmé, že pro architekturu z obr.2.2.11 bude platit, že

$$W \geq N$$

kde W je počet bitů řádky překódovací tabulky

N je počet bitových rovin, tj. počet bitů reprezentující číslo zvolené úrovně šedi

Pro délku L překódovací tabulky, tj. počet řádek, musí platit

$$2^N \leq L \leq 2^W$$

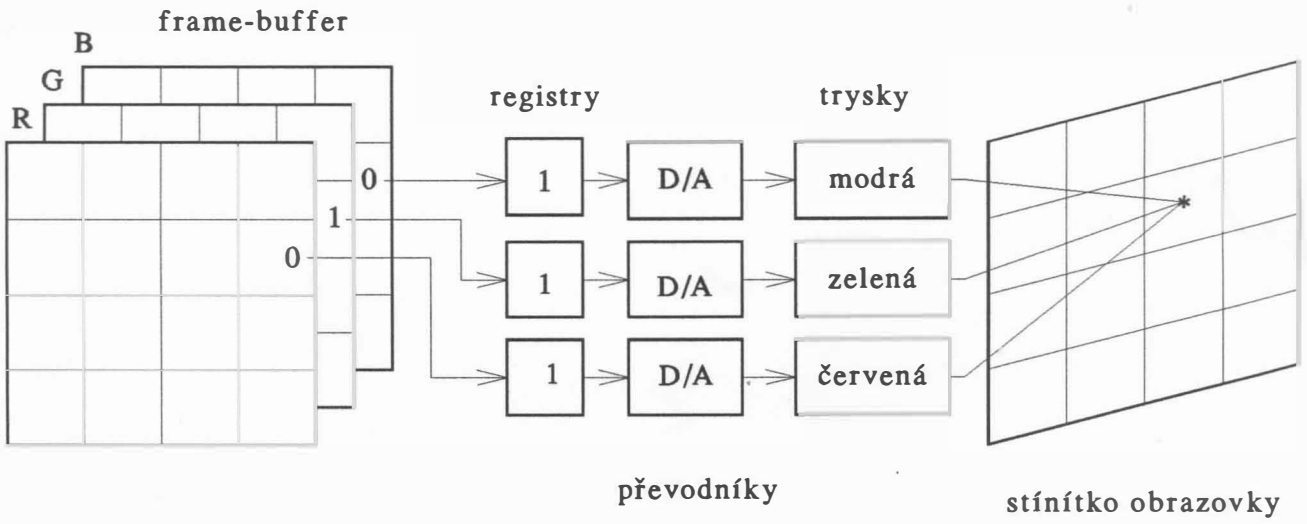
Celková paměť frame - bufferu pro N = 8 (tj. 256 úrovní šedi) při adresovatelnosti 1024 x 1024 bodů je rovna 1 MB.

Zkusme nyní výše uvedené architektury použít v případě zobrazování barev. Nejjednodušší je použít 1-bitovou rovinu pro každou barvu. Budeme pracovat v systému RGB, kde je R - red (červená), G - green (zelená), B - blue (modrá). Pak pro jednotlivé kombinace dostáváme:

barva	R	G	B
černá	0	0	0
modrá	0	0	1
zelená	0	1	0
modro-zelená	0	1	1
červená	1	0	0
purpurová	1	0	1
žlutá	1	1	0
bílá	1	1	1

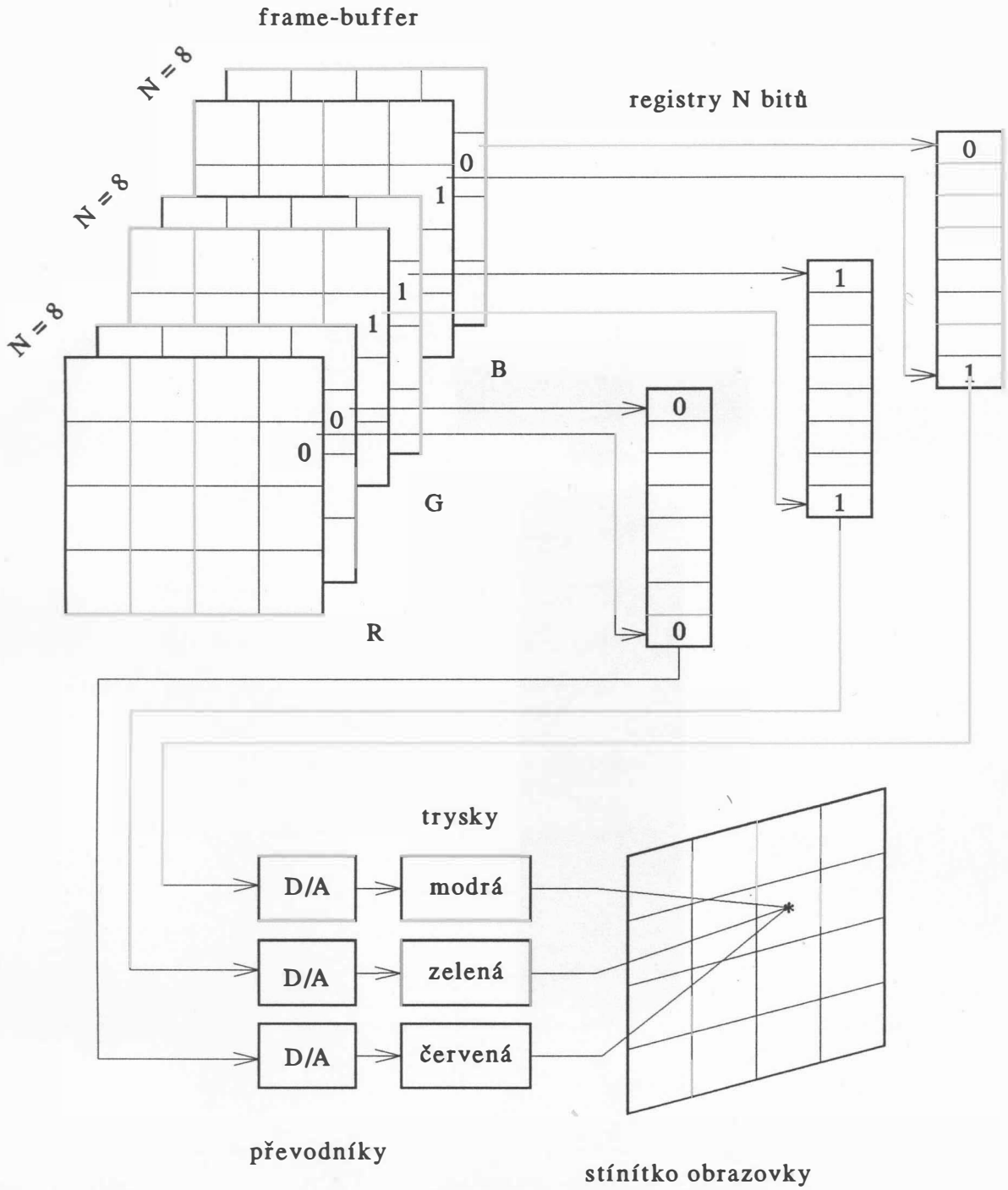
Je známo, že barevné obrazovky mají tři trysky, a to pro každou základní barvu jednu. Nejjednodušší případ je zobrazen na obr.2.2.12.

Nyní je možné každé barvě přiřadit ne jednu, ale několik bitových rovin a získat tak různé barevné odstíny. Dostáváme případ znázorněný na obr.2.2.13.



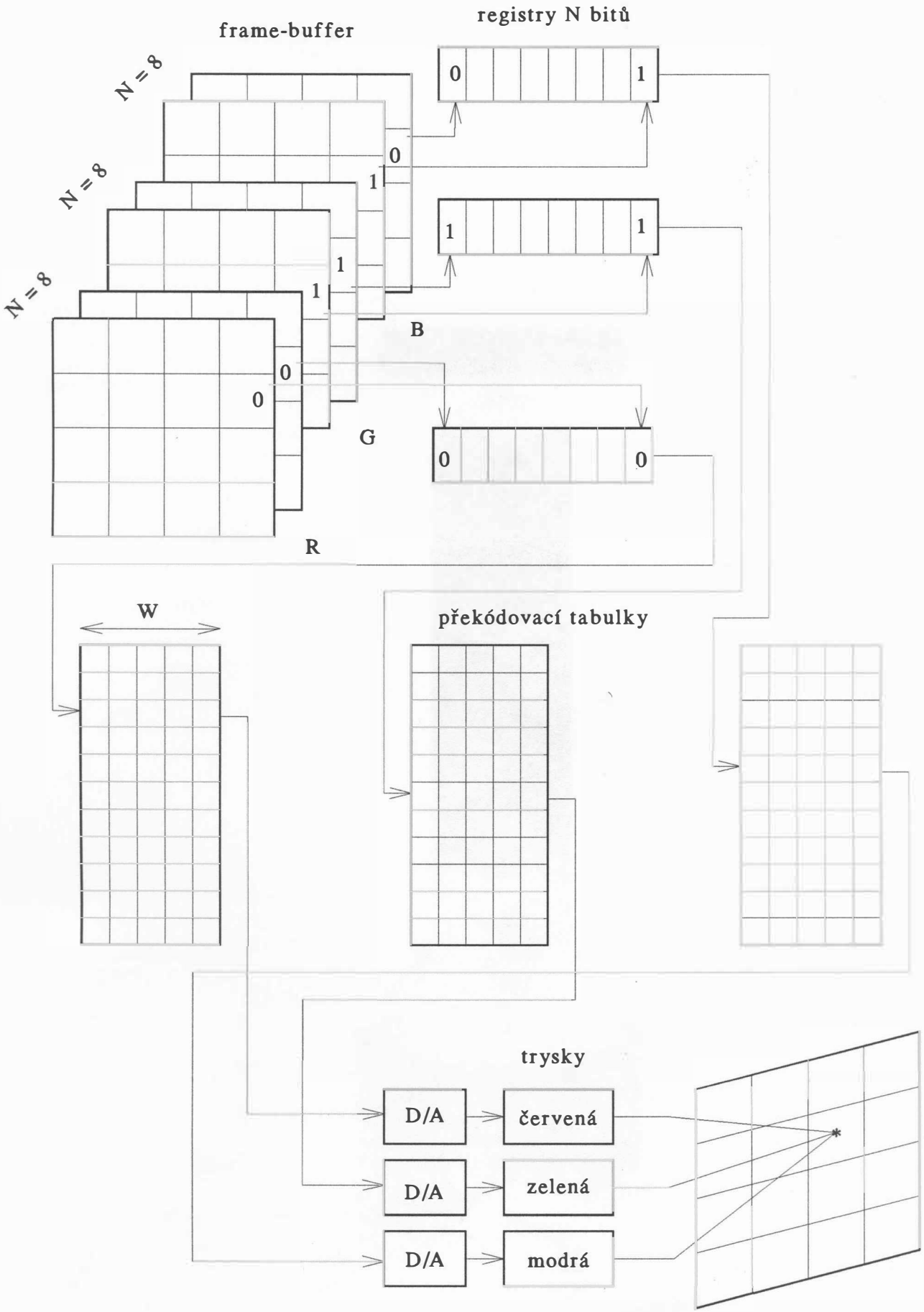
Frame buffer s jednou bitovou rovinou pro každou barvu

Obr.2.2.12



Frame buffer s osmi bitovými rovinami pro každou barvu

Obr.2.2.13



převodníky Obr. 2.2.14 stínítko obrazovky

Uvážíme-li nyní opět adresovatelnost 1024 x 1024 bodů při 8-mi bitových rovinách pro každou barvu, pak dostáváme kapacitu paměti rovnu

3 * 8 * 1024 * 1024 = 24 Mb = 3 MB

Pro zvýšení počtu dostupných barev je možné zařadit opět překódovávací tabulku, čímž dostaneme architekturu znázorněnou na obr.2.2.14.

Nyní je nutné opět poukázat na pojmy adresovatelnost a rozlišitelnost, které se velmi často ztotožňují a zaměňují, zejména v oblasti popisu systémů ze zahraničí. Dnešní grafické displaye obvykle mají adresovatelnost 4096 x 4096 bodů při rozlišitelnosti 1024 x 1024 bodů, přičemž např. z palety 16 777 216 (2^{24}) barev je zobrazitelných na obrazovce pouze 512 barev. To znamená, že **adresovatelnost je 4096 x 4096 x 16777216**, zatímco **rozlišitelnost**, tj. to, co se opravdu zobrazí pokud je k dispozici odpovídající monitor, **je pouze 1024 x 1024 x 512**.

Z obr.2.2.14 vyplývá, že paleta barev, z které můžeme vybírat, je rovna

$$(2^3)^W = (2^W)^3$$

zatímco počet zobrazitelných barev v daném čase na obrazovce je

$$(2^3)^N = (2^N)^3$$

a tedy pro případ s 256 barvami dostáváme:

počet bitových rovin pro barvu: N = 3

počet bitů pro překódovací tabulku: W = 8

kapacita paměti frame-bufferu P:

4096 x 4096 x 3 x 3 = 144 Mb = 18 MB

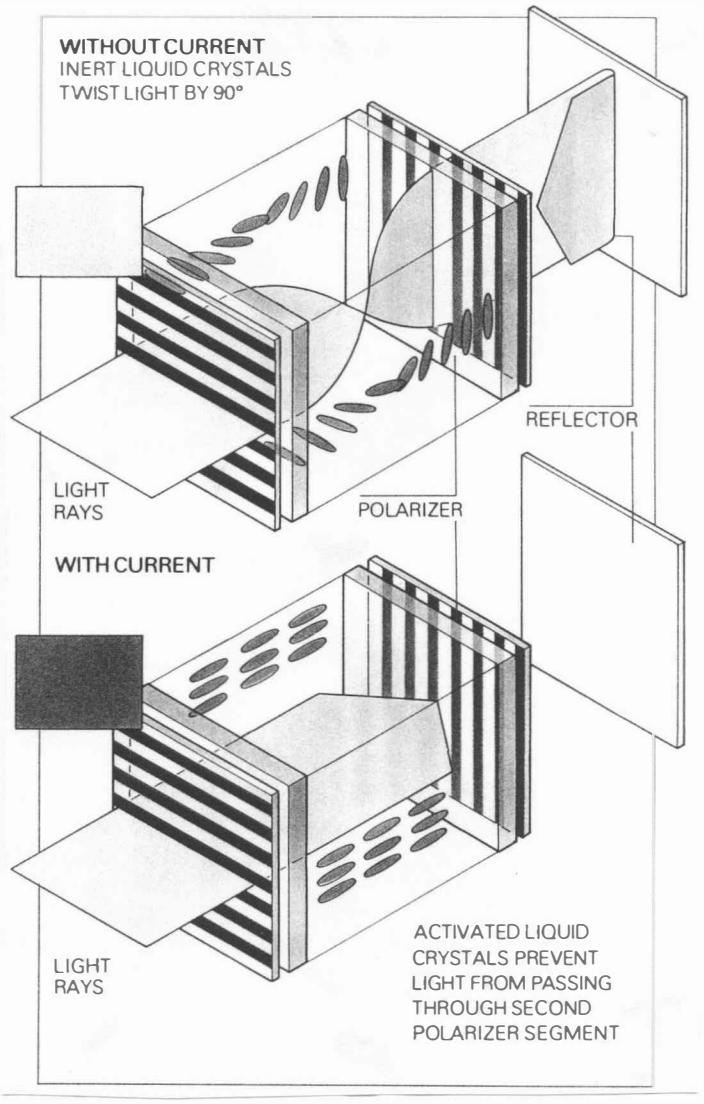
kapacita překódovacích tabulek T:

256 x 8 x 3 = 6144 b = 768 B

Je zřejmé, že jednotlivé bity nelze vybírat postupně z časových důvodů, ale vybírá se vždy skupina bodů (doba přístupu pamětí je okolo 100 ns).

Displaye s tekutými krystaly

Displaye s tekutými krystaly (Liquid Crystal Display - LCD) patří mezi displaye, které neemitují světlo, ale modulují dopadající světlo. Tekuté krystaly tvoří vlastně vrstvu organických molekul, které mají tvar tyčinek. Polohu tyčinek je možné měnit elektrickým polem. Bez elektrického pole dochází k otočení polarizace světla o 90°, které je druhým polarizačním filtrem propuštěno a odraženo zpět. Pokud působí elektrické pole, pak ke změně polarizace světla nedochází a světlo není druhým polarizačním filtrem propuštěno, a tedy nemůže být ani odraženo zpět. Ve spojení s polarizačním filtrem je pak možné vytvořit prvek, který buď propouští světlo, či nikoliv, viz obr.2.2.15.

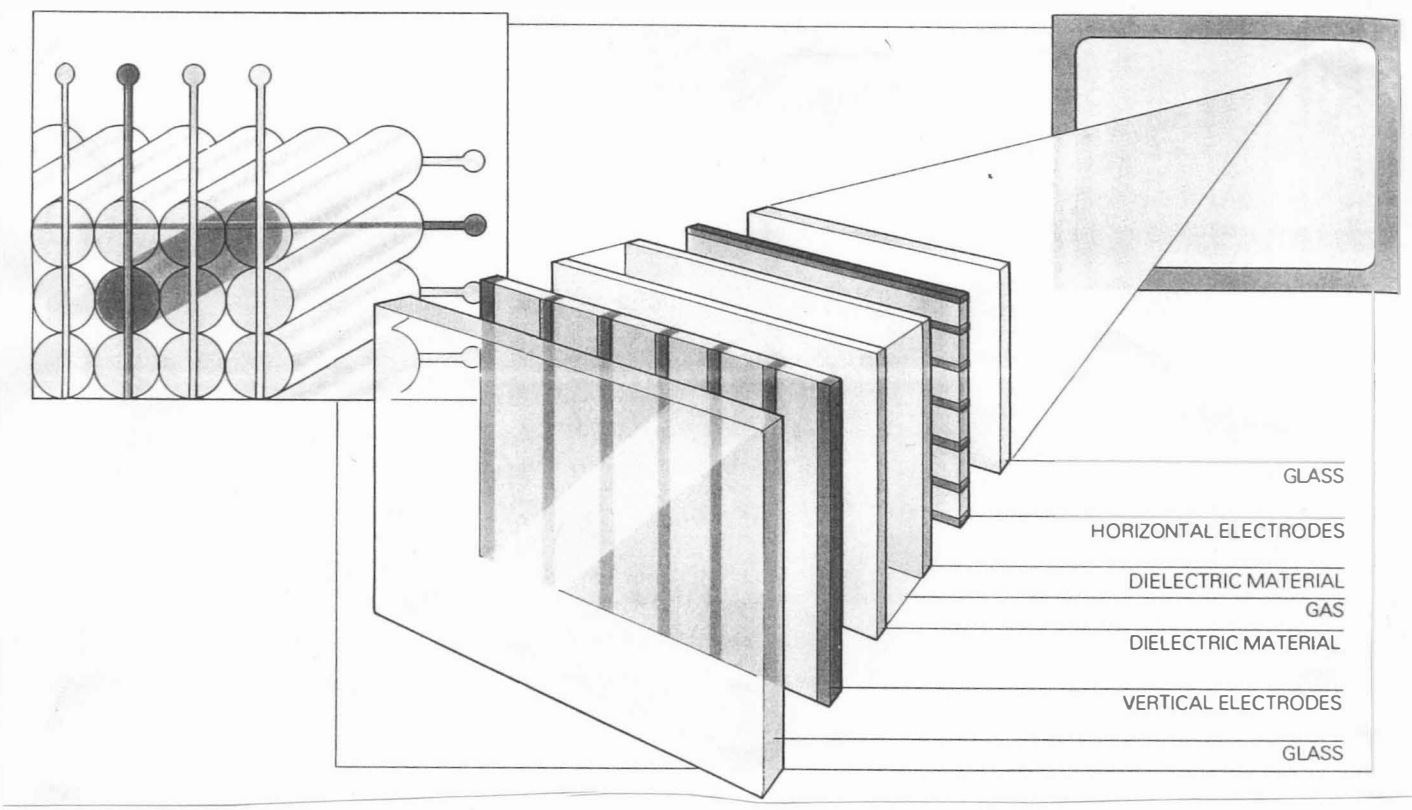


Princip LCD displaye LEWELL
Obr.2.2.15

LCD displaye se vyrábějí ve formě jak odrazných, tak i propustných panelů. Odrazný typ, známý např. z digitálních hodinek, se používá ve spojení s osobními počítači třídy **LAPTOP**. Forma propustných panelů se pak používá např. ve spojení se zpětnými projektory a ty se pak používají k výukovým a demonstračním účelům. Dosahovaná rozlišovací schopnost je asi 1100 x 800 bodů při 16 úrovních šedi, např. viz laptop **NEWS** firmy **SONY**. Největším problémem je realizace barevných LCD displayů.

Plasmové displaye

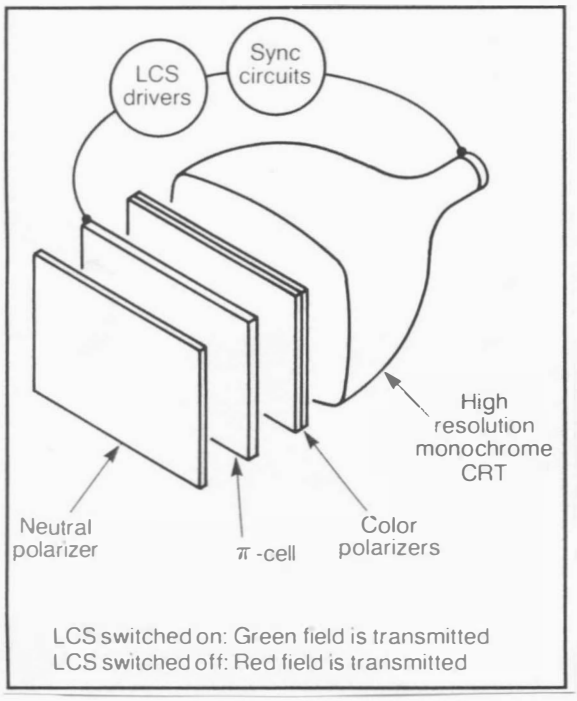
Plasmové displaye jsou realizovány jako dvourozměrná pole malých neónových zdrojů světla. Každý zdroj světla je možné vypnout či zapnout asi za 20 μs. U těchto displayů lze dosáhnout 16 úrovní jasu při rozlišení 1024 x 768 pixelů. Princip plasmových displayů je znázorněn na obr.2.2.16.



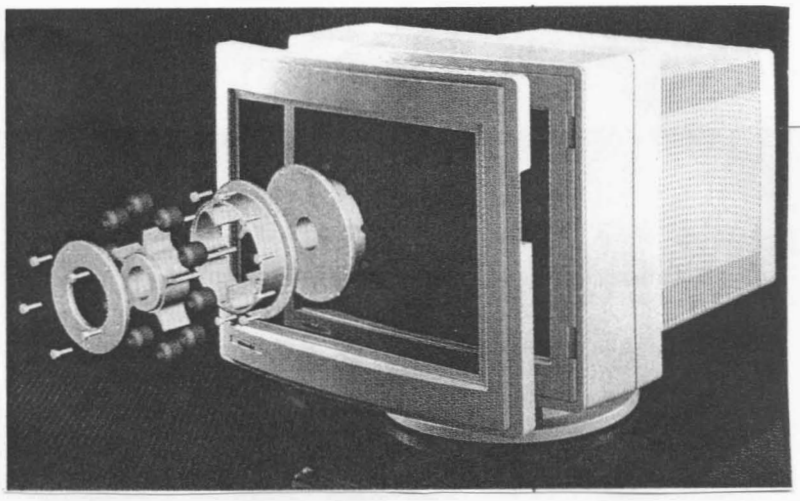
Princip plasmového displaye LEWELL

Obr.2.2.16

Jak plasmové, tak i LCD displaye tvoří alternativu k displayům obrazovkovým. Je však nutné počítat s tím, že obrazovkové displaye se budou ještě dlouhou dobu používat zejména v průmyslu.



Princip steroskopického displaye



Display SGS/625
firmy TEKTRONIX Inc.

Obr. 2.2.18

3D displaye

Až dosud předkládané principy zařízení umožňovaly zobrazení třírozměrných objektů po projekci na promítací plochu, tj. více méně zobrazení dvourozměrných útvarů. Získání skutečného vjemu třírozměrných objektů operátorem je nezbytné pro mnoho aplikací. Některé systémy, zejména simulátory pilotáže letadel apod., používají např. projekčních obrazovek ve spojení s parabolickými zrcadly. Tyto systémy umožňují získat vjem třetího rozměru. Jiné systémy používají brýlí s polarizačními či barevnými filtry apod. Firma Genisco Computers Corp. vyvinula display s vibrujícím zrcadlem. Ten pak poskytuje operátorovi vjem skutečného třírozměrného prostoru. Princip zařízení vychází z jednoduché myšlenky, a to: pohybujeme-li předmětem před zrcadlem vpřed a vzad, pak pozorovatel vnímá tuto změnu. Tento jev je pak využit tak, že předmět (ve skutečnosti obraz na obrazovce) stojí a pohybuje se zrcadlo. Místo pevného zrcadla se pak používá pružná membrána, která má zrcadlově odrážející povrch a její prohnutí či vydutí je řízeno počítačem synchronně se zobrazováním objektu na obrazovce. Jde vlastně o zobrazování jednotlivých "řezů" směrem od pozorovatele v jednotlivých časových úsecích. Pokud jsou všechny "řezy" zobrazovány rychleji než za 1/30 s, pak dostáváme neblikající obraz se vjemem hloubky. Podrobnosti lze nalézt v [57].

Jistou náhradou výše uvedeného systému je použití brýlí s barevnými nebo polarizačními filtry. I když toto řešení je známé dlouhou dobu, firma TEKTRONIX Inc. přišla na trh s řadou displayů SGS, které jsou realizovány pomocí monochromního displaye. Využívá se barevného rozkladu bílého světla produkovaného touto obrazovkou. Před stínítkem je umístěn speciální filtr z tekutých krystalů s vysokou rozlišovací schopností a přepínací rychlostí. Tento filtr umožňuje, aby se buď propouštěla barva červená, anebo zelená. Při použití brýlí s barevnými filtry je pak výsledkem stereoskopický vjem, přičemž je možné zobrazovat všechny barvy od červené až po zelenou včetně barevných odstínů. Je zřejmé, že tento systém musí zobrazovat dvojici stereoskopických snímků alespoň 30 krát za sec. Na obr.2.2.18 je pak ukázán princip displaye a jeho skutečná podoba.

2.3 Netradiční vstupní a výstupní zařízení

Dříve uvedená vstupní a výstupní zařízení, která jsou běžně vyráběna a dodávána, jsou v mnoha případech finančně nákladná nebo jinak nedostupná. Typickou ukázkou může být např. scanner pro větší formáty. V mnoha aplikacích může být nezbytné nasnímat předlohy větší než formát A4. Pokud však úloha snímání není často prováděna, pak zakoupení scanneru např. formátu A1 nebude pravděpodobně finančně únosné. V tomto případě pak lze:

- rozstříhat snímanou předlohu na formáty A4 (s problémy co s okraji) a zakoupit nebo vytvořit speciální programové vybavení, které tyto části "složí" opět do jednoho celku
- zadat tuto úlohu firmě, která zajistí nasnímání dané předlohy bez jejího zničení. V tomto případě je nutné řešit problém přenosu dat, neboť přenos pomocí disket je **nerealizovatelný**. Pokud nepoužijeme technik komprese dat, pak pro formát A4 při hustotě 400 dpi a při osmi bitech na složky RGB je nutná kapacita paměti k uložení nasnímaných dat asi **44 MB**, při nasnímání formátu A0 je pak již zapotřebí asi **704 MB**.

Z uvedeného je zřejmé, že úloha vlastního sejmutí předlohy formátu A0/A1 nebude tak častá, neboť zpracování odpovídajícího objemu dat bude časově velmi náročné. Z tohoto důvodu lze pro méně náročné aplikace použít zařízení, které zajistí nasnímání předlohy s tím, že např. se použije plotter, na kterém je v držáku pro pero upevněna speciální sonda pro snímání. Takovým způsobem lze nasnímat předlohy s rozlišením 12 nebo 16 úrovní šedi. Velikost nasnímaného formátu je pak dána velikostí předlohy a velikostí kreslící plochy plotteru. Takové řešení je finančně nenáročné a je plně postačující pro mnoho aplikací, zejména však tam, kde je nutné občas nasnímat i tzv. prodloužené formáty.

Podobným způsobem je možné i řešit problém snímání bodů z předloh různého formátu, tj. náhradu snímačů souřadnic, kdy pomocí kláves ovládajících polohu kurzoru, myši nebo speciální klávesnice určujeme posuvy na plotteru v obou směrech tak, aby se nitkový kříž kryl se snímaným bodem. Stisknutím klávesy **ENTER** se pak provede odečtení souřadnic nasnímaného bodu

(většina plotterů je schopna určit současnou pozici pera). Klávesa **ESC** pak může rušit platnost nasnímaných dat.

Dalším zvláštním zařízením je zařízení nazývané **PENMAN**, které je podobné myši, ale s tím rozdílem, že jde o zařízení výstupní, viz obr.2.3.1.

Robotic Plotter firmy Zericon
Obr.2.3.1

Zařízení obsahuje dva krokové motorky, které jsou umístěny v těžkém kovovém kvádru, na kterém jsou umístěna též kreslící pera. Zařízení položené na kreslící ploše se pak pohybuje vpřed nebo vzad, přičemž je možné měnit směr pohybu. Toto zařízení vyniká svojí jednoduchostí a minimem požadovaných mechanických součástí pro výrobu při zachování dostatečné přesnosti kresby.

Mezi netradiční výstupní zařízení lze počítat i tzv. **technologické grafické displaye**, které se vyznačují stálostí většiny zobrazovaných prvků, přičemž se většinou mění jen několik numerických údajů (napětí, resp. průtok) nebo barva symbolů (odpojeno/připojeno, resp. uzavřeno/otevřeno). Vzhledem k malému počtu měnících se dat je možné použít pro komunikaci i sériové rozhraní, neboť přenos dat definujících "pozadí" výsledného obrazu se přenáší jen při inicializaci.

PC video výstupy

Vzhledem k rozvoji osobních systémů je vhodné uvést základní typy videoadaptérů (karet) a jejich rozlišovací schopnosti:

Color Graphics Adapter - CGA : 640 x 200 bodů při dvou současně zobrazitelných barvách; modifikace umožňují až 16 barev současně.

Hercules Graphics Card - HGC : 720 x 348 bodů při dvou barvách - pouze textový režim; modifikace HGC Plus umožňuje i grafický režim s případnou volbou barev.

Enhanced Graphics Adapter - EGA : 640 x 350 až 16 současně zobrazitelných barev ze 64 možných (v závislosti na konfiguraci paměti karty).

Video Graphics Array - VGA : 640 x 480 až 16 barev z 256 možných, nebo 320 x 480 při 256 barvách.

Hercules GSC (Graphics Station Card) : používá procesor 34010 a umožňuje při rozlišení 1024 x 768 až 256 barev.

Future Graphics Adapter - FGA : 1280 x 1024 až 256 barev z palety 16.7M; adaptér umožňuje obsluhovat dva monitory, nebo překrýt výstupy pomocí oken na jeden monitor, přičemž překrytí je řešeno hardwarově.

Velká většina dnes dodávaných videokaret používá procesor 8514/A, který umožňuje zobrazit 56 barev z 256K mořných při rozlišení 1024 x 768 a který je též vyráběn firmou Western Digital jako PWGA1 (Personal Workstation Graphics Array 1), nebo využívá novou architekturu TIGA (Texas Instrument Graphics

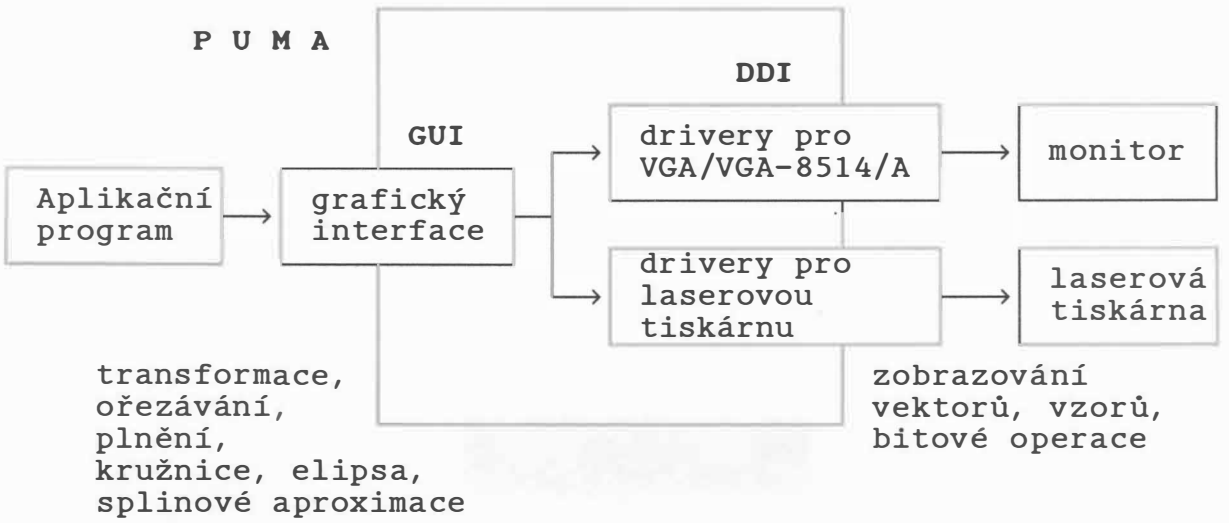
Architecture), který je založen na použití procesorů řady 340x0 a který umožňuje zobrazit až 566M barev při rozlišení 1280 x 1024 bodů.

Kromě výše uvedených základních typů existuje celá řada nejen jejich modifikací (Super VGA 1024 x 1024 až 256 barev apod.), ale též i karet s vyššími užitkovými parametry, z nichž uvedeme jen některé (údaje jsou uvedeny ve tvaru rozlišovací schopnost, počet zobrazovaných barev, počet možných barev).

ARTIST 1 MONO	1024 x 768	
ARTIST 1 PLUS	1024 x 768 / 4 / 4096	
ARTIST 10	1024 x 768 / 256 / 262K	
ARTIST TI	1280 x 1024 / 256 / 16 M	
TVGA	1280 x 1024 / 256 / 16 M	
SOTA 340i	1280 x 1024 / 256	
Q2000	2048 x 2048 / 256 / 16 M	
TVGA 8900	2048 x 2048 / 256 / 16 M	
Spectrum/24	1024 x 768	až 24 bitů na 1 pixel
ColorBOARD/24	1024 x 768	
DirectColor/24	1152 x 882	

S nástupem pracovních stanic však uvedené videokarty neposkytují dostatečnou výkonnost vzhledem k požadované rychlosti, rozlišovací schopnosti a počtu zobrazitelných barev. Ukázkou je pak např. procesor IMS G180 [176] firmy INMOS, který pracuje na frekvenci 200 MHz a který poskytuje až 32 bitů na pixel.

Grafický akcelerátor PUMA [177] firmy Chips je zajímavý nejen svými výkonnostními parametry, ale i použitím grafického rozhraní na vysoké úrovni. Na obr. 2.3.2 je ukázán předpokládaný způsob použití.



GUI (Graphics User Interface) - uživatelský grafický interface
DDI (Device Dependent Interface) - interface závislý na zařízení

Obr. 2.3.2

Vedle videoadaptérů jsou vyráběny ještě karty umožňující zpracování videosignálů jak v normě PAL, tak i NTSC.

	rozlišení	signál	formát dat	
ColorCapture	640 x 480	NTSC	TIFF	16 bitů
	512 x 575	PAL	IMAGE	
NuVista	756 x 485	NTSC	TGA	32 bitů
	768 x 575	PAL	PICT	

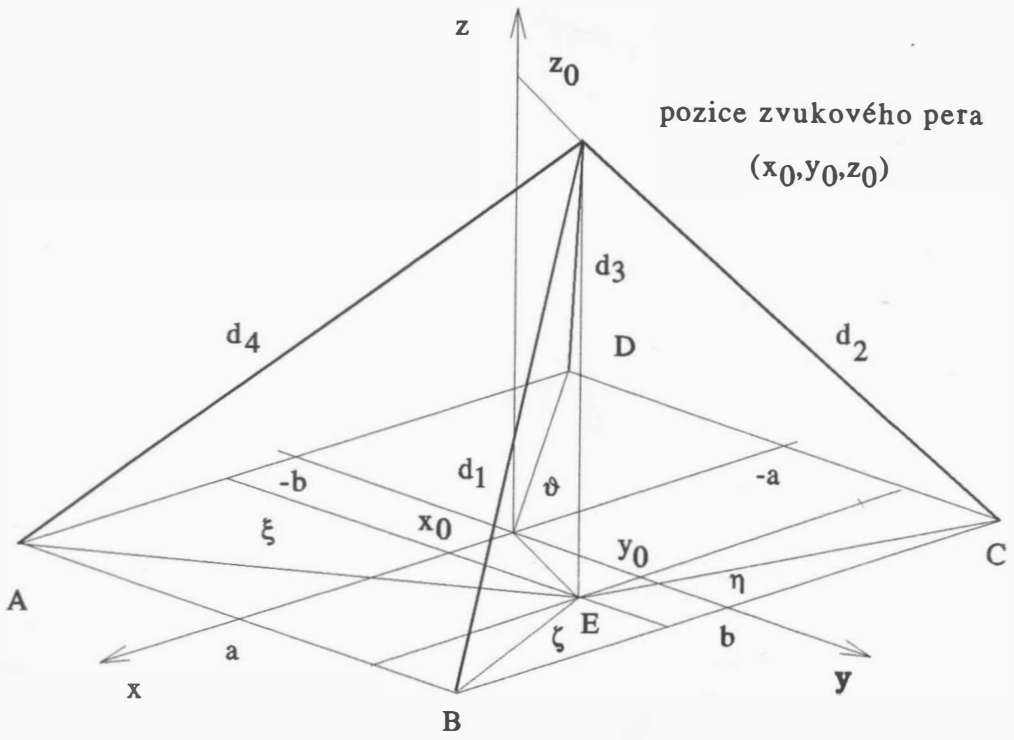
Některé nejnovější systémy nabízejí vyšší rozlišovací schopnost při 24 bitech pro rozlišení barev a podporují celou řadu formátů pro ukládání obrazových dat, jako: TIFF, TGA, IFF, Targa, GIF atd.

Kromě uvedených základních principů zařízení se již dnes začínají prosazovat specializované procesory, v nichž se využívá paralelního zpracování k podstatnému urychlení realizovaných operací.

2.4 Příklady

Příklad$_1$

Odvoďte základní vztahy pro odměřování v třírozměrném prostoru při použití zvukového pera. Předpokládejte, že jsou použity čtyři bodové mikrofony, které jsou umístěny v rozích měřící oblasti.



Odměřování s použitím čtyř mikrofonů
Obr. 2.4.1

Z obr. 2.4.1 vyplývá, že platí

$$d_1^2 = z_0^2 + \zeta^2 \qquad d_2^2 = z_0^2 + \eta^2$$

$$d_3^2 = z_0^2 + \vartheta^2 \qquad d_4^2 = z_0^2 + \xi^2$$

Pro ζ , ξ , η , ϑ platí následující vztahy

$$\zeta^2 = (a - x_0)^2 + (b - y_0)^2$$

$$\eta^2 = (a + x_0)^2 + (b - y_0)^2$$

$$\vartheta^2 = (a + x_0)^2 + (b + y_0)^2$$

$$\xi^2 = (a - x_0)^2 + (b + y_0)^2$$

Pak po úpravách dostáváme

$$d_3^2 - d_4^2 = 4ax_0 \qquad d_2^2 - d_1^2 = 4ax_0$$

Z výše uvedených vztahů vyplývá, že

$$| d_3^2 - d_4^2 - d_2^2 + d_1^2 | \leq \varepsilon$$

kde ε je hranicí možných nepřesností při odměřování apod.
Analogicky pro y_0 dostáváme

$$d_4^2 - d_1^2 = 4by_0$$

resp.

$$d_3^2 - d_2^2 = 4by_0$$

Na základě výše uvedených vztahů je možné určit souřadnice x_0 , y_0 takto:

$$x_0 = (d_3^2 - d_4^2 + d_2^2 - d_1^2) / (8\,a)$$

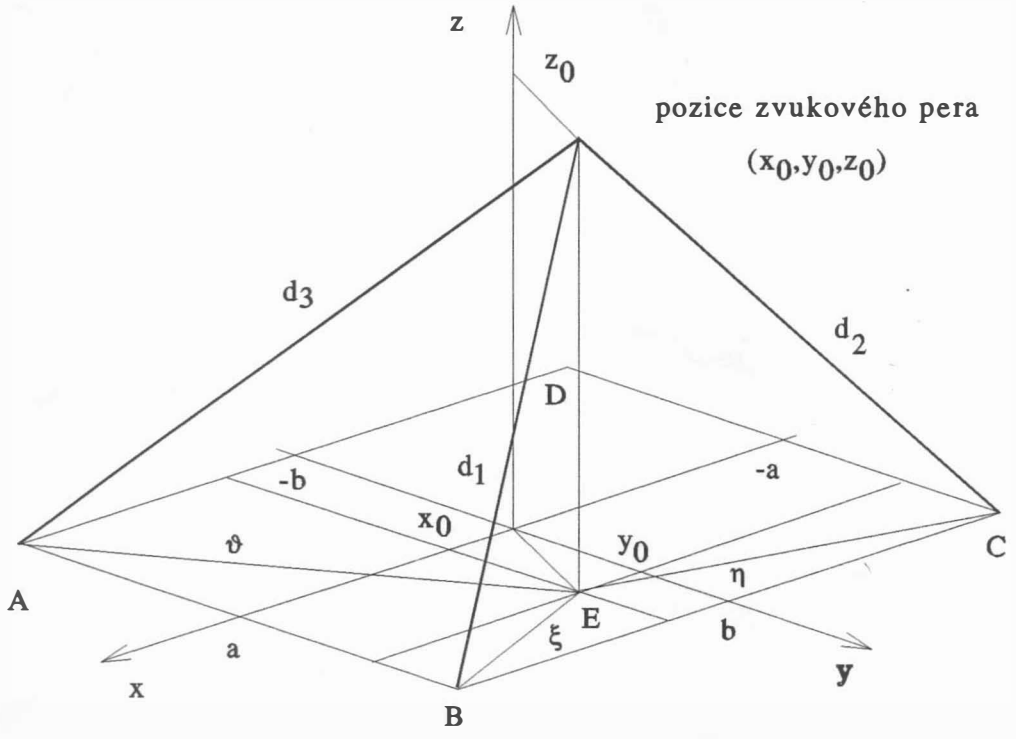
$$y_0 = (d_4^2 + d_3^2 - d_1^2 - d_2^2) / (8\,b)$$

Nyní je nezbytné určit souřadnici z_0. Lze ukázat, že platí:

$$z_0^2 = (d_1^2 + d_2^2 + d_3^2 + d_4^2) / 4 - a^2 - b^2 - x_0^2 - y_0^2$$

Příklad 2

Odvoďte základní vztahy pro odměřování v třírozměrném prostoru při použití zvukového pera. Předpokládejte, že jsou použity tři bodové mikrofony, které jsou umístěny ve třech rozích měřící oblasti.



Odměřování s použitím tří mikrofonů
Obr. 2.4.2

Z obr. 2.4.2 vyplývá, že

$$d_1^2 = z_0^2 + \xi^2 \qquad d_2^2 = z_0^2 + \eta^2 \qquad d_3^2 = z_0^2 + \vartheta^2$$

Pro ξ , η , ϑ platí

$$\xi^2 = (a - x_0)^2 + (b - y_0)^2$$

$$\eta^2 = (a + x_0)^2 + (b - y_0)^2$$

$$\vartheta^2 = (a - x_0)^2 + (b + y_0)^2$$

Na základě uvedených vztahů pak úpravami dostáváme

$$x_0 = (d_2^2 - d_1^2) / (4\ a)$$

$$y_0 = (d_3^2 - d_1^2) / (4\ b)$$

Označíme-li

$$\delta_1 = (d_2^2 - d_1^2) / 2 \qquad \delta_2 = (d_3^2 - d_1^2) / 2$$

a

$$\alpha = 2a^2 + x_0^2 + y_0^2$$

pak z rovnic pro d_1 , d_2 , d_3 dostáváme

$$z_{0_1}^2 = d_1^2 - \alpha + \delta_1 + \delta_2$$

$$z_{0_2}^2 = d_2^2 - \alpha - \delta_1 + \delta_2$$

$$z_{0_3}^2 = d_3^2 - \alpha + \delta_1 - \delta_2$$

Pak

$$z_0^2 = (z_{0_1}^2 + z_{0_2}^2 + z_{0_3}^2) / 3$$

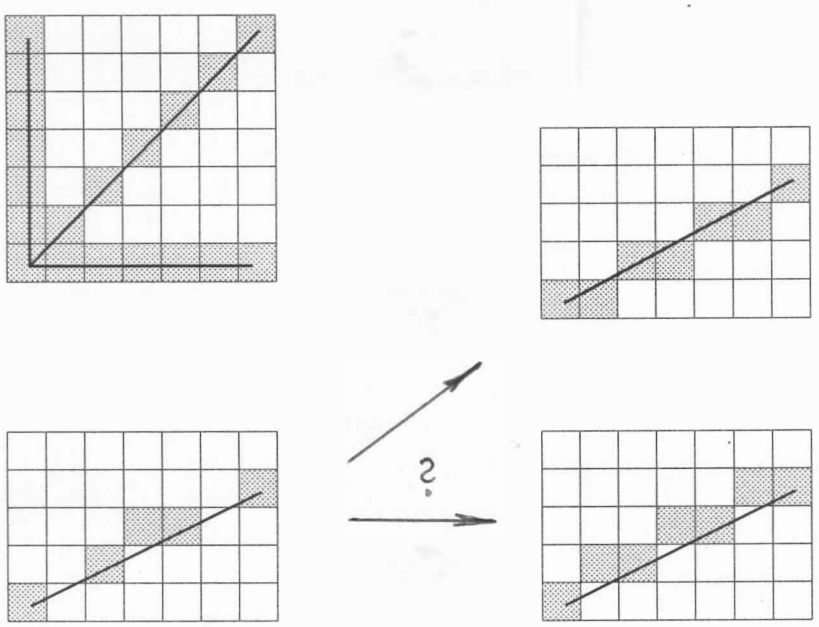
Uvedený postup výpočtu souřadnic poskytuje i možnosti zjištění některých chyb zařízení, např. pokud některý mikrofon nepracuje spolehlivě.

3. Základní algoritmy rastrové grafiky

Rastrová zařízení vyžadují speciální algoritmy k vytváření obrazů, ke kreslení přímek a křivek, případně k vyplňování n-úhelníků tak, abychom měli dojem plné oblasti. V předchozí kapitole a v [49], [109] jsou uvedeny základní principy rastrových zařízení. V této kapitole se budeme zabývat jednotlivými základními algoritmy, které jsou vhodné pro rastrová zařízení obou typů, tj. jak se sekvenčním přístupem, tak i s přímým přístupem k bodu.

3.1 Algoritmus pro kreslení čáry

V případě rastrového zařízení je zobrazovací plocha vlastně rozdělena do matice jednotlivých políček, z nichž každé je možné jednotlivě adresovat, i když ne vždy přímo modifikovat, a definovat též jejich jas či barvu. Jednotlivé políčko je označováno jako pixel. Na rozdíl od vektorových zařízení se v případě rastrových zařízení musíme zabývat otázkou, jakým způsobem se vlastně zobrazí úsečky, resp. křivky. V případě čar vodorovných, svislých anebo čar s 45° sklonem je generace obrazem jednotlivých pixelů v rastru vcelku jednoduchá, viz obr. 3.1.1.



aktivace pixelů v rastru

Obr. 3.1.1

V ostatních případech se však dostáváme do potíží. Co je vlastně základním požadavkem pro hledaný algoritmus? Budeme zajisté požadovat, aby přímka zůstala přímkou při zachování maximální přesnosti. Kreslení čáry by mělo probíhat s maximální rychlostí, a tedy vlastní algoritmus musí být poměrně jednoduchý.

V běžném režimu budeme též požadovat, aby čára měla konstantní jas. Je zřejmé, vzhledem k rastrovému principu, že ne všechna obvyklá kritéria lze splnit beze zbytku, neboť přesnost je dána jemností rastru. Pro zlepšení vizuálního vjemu je možné provádět modulaci jasu zejména "postranních" pixelů, což však vede ke složitějším a tedy i pomalejším algoritmům. Obecně se požaduje, aby algoritmy pokud možno používaly pouze celočíselnou reprezentaci čísel a byly realizovány hardwarově.

Většina metod je založena na přírůstkovém principu - tzn., že z dané pozice chceme nakreslit čáru do pozice vzdálené o $(\Delta x, \Delta y)$.

3.2 Digitální diferenciální analyzátor

Jednou z možností, jak generovat úsečku v rastrovém prostředí, je použití digitálního diferenciálního analyzátoru (DDA) k řešení diferenciální rovnice:

$$\frac{dy}{dx} = \text{konst.} \qquad \text{nebo} \qquad \frac{\Delta y}{\Delta x} = \frac{y_b - y_a}{x_b - x_a}$$

kde (x_a, y_a) je počáteční bod, (x_b, y_b) je koncový bod úsečky. Pak řešením je ($\Delta x = 1$, neboť $x_{i+1} - x_i = 1$):

$$y_{i+1} = y_i + \Delta y$$

$$y_{i+1} = y_i + \frac{y_b - y_a}{x_b - x_a} \qquad x_{i+1} = x_i + 1$$

tj. kde: (x_0, y_0) je bod (x_a, y_b).

Je zřejmé, že tento algoritmus je použitelný pouze pro ty oktanty, kde platí $|\Delta x| \geq |\Delta y|$. Pro ostatní oktanty se musí zaměnit souřadnice x a y. Algoritmus DDA, který je modifikován tak, aby pracoval pro všechny oktanty, lze realizovat podprogramem:

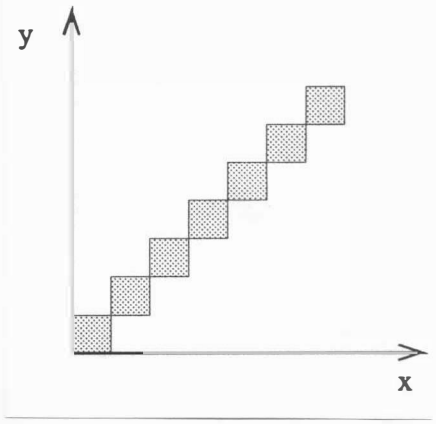
```
procedure DDA ( xa, ya, xb, yb: integer );
var dx, dy, i, ix, iy, x, y: integer;
    er, sm: real;
begin dx := xb - xa; dy := yb - ya;
      x := xa; y := ya; er := 0.5;
      PLOT ( x , y );
      if dx > 0 then ix:=1 else ix:=-1;
      if dy > 0 then iy:=1 else iy:=-1;
      if abs (dx) >= abs(dy) then
      begin sm := abs(dy)/abs(dx);
            for i:=1 to abs(dx) do
            begin er := er + sm;
                  if er >= 1 then
                  begin y := y + iy; er := er - 1 end;
                  x := x + ix;
                  PLOT(x,y)
            end
      end
      else
      begin sm := abs(dx)/(dy);
            for i:=1 to abs(dy) do
            begin er := er + sm;
                  if er >= 1 then
                  begin x := x + ix; er := er - 1 end;
                  y := y + iy;
                  PLOT(x,y)
            end
      end
end;
{ PLOT(x,y) zajistí aktivaci bodu o souřadnicích(x,y) }
```

Algoritmus 3.2.1

Uvážíme-li úsečku z bodu (0,0) do bodu (5,5), pak DDA algoritmus bude mít následující stavy:

xa = 0	ya = 0	x = 0	y = 0
xb = 5	yb = 5	dx = 5	dy = 5
ix = 1	iy = 1	sm = 1	er = 0.5

i	er	x	y
1	0.5	1	1
2	0.5	2	2
3	0.5	3	3
4	0.5	4	4
5	0.5	5	5



Výsledek DDA algoritmu v 1. kvadrantu obr.3.2.1

Z uvedeného algoritmu je zřejmé, že v případě na sebe navazujících úseček bude koncový a počáteční bod zapsán dvakrát. Toto může v některých případech vést k zvýraznění nebo změně barvy těchto bodů, zejména u zařízení provádějících zápis přímo na film.

Uvažme nyní úsečku z bodu (0,0) do bodu (-8,-4) ve třetím kvadrantu. Algoritmus DDA pak bude mít následující stavy:

```
xa = 0     ya = 0
x = 0      y = 0
xb = -8    yb = -4
dx = -8    dy = -4
ix = -1    iy = -1
sm = 0.5   er = 0.5
```

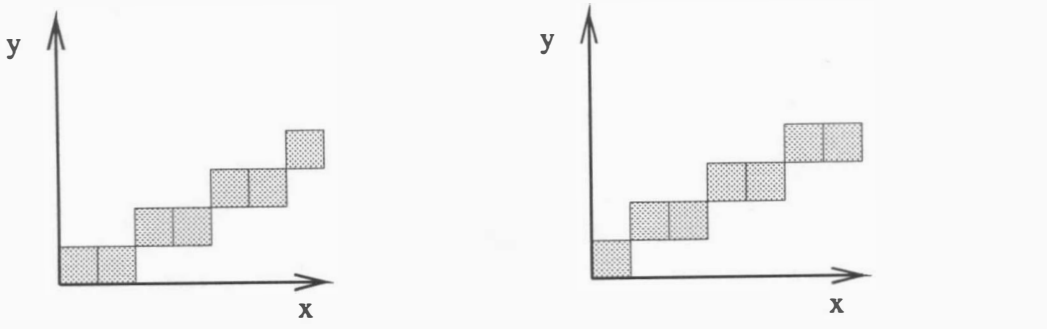
i	er	x	y
1	0	-1	-1
2	0.5	-2	-1
3	0	-3	-2
4	0.5	-4	-2
5	0	-5	-3
6	0.5	-6	-3
7	0	-7	-4
8	0.5	-8	-4

Výsledek je uveden na obr.3.2.2.a.

Budeme-li však mít úsečku z bodu (-8,-4) do bodu (0,0), pak DDA bude mít stavy:

```
xa = -8     ya = -4    x = -8     y = -4
xb = 0      yb = 0     dx = 8     dy = 4
ix = 1      iy = 1     sm = 0.5   er = 0.5
```

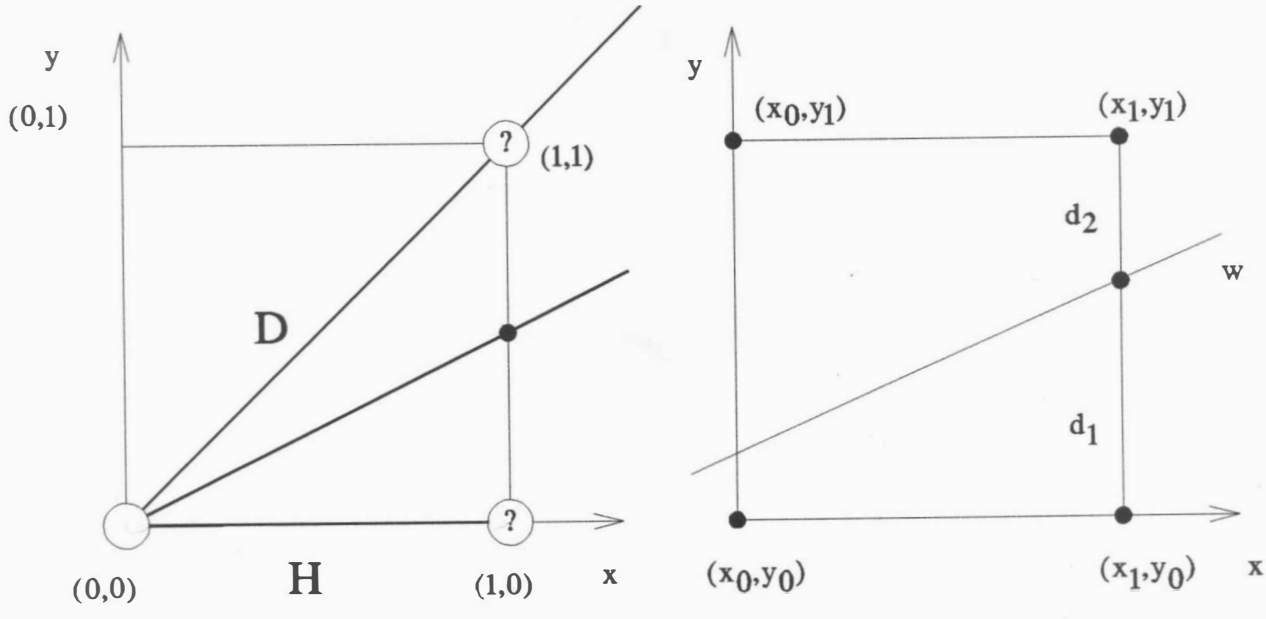
i	er	x	y	i	er	x	y
1	0	-7	-3	5	0	-3	-1
2	0.5	-6	-3	6	0.5	-2	-1
3	0	-5	-2	7	0	-1	0
4	0.5	-4	-2	8	0.5	0	0



Obr. 3.2.2

Výsledek je uveden na obr. 3.2.2.b.

Porovnáním obou výsledků lze snadno nahlédnout, že generovaná čára není symetrická. Navíc DDA algoritmus vyžaduje použití procesoru s pohyblivou řádovou čárkou.



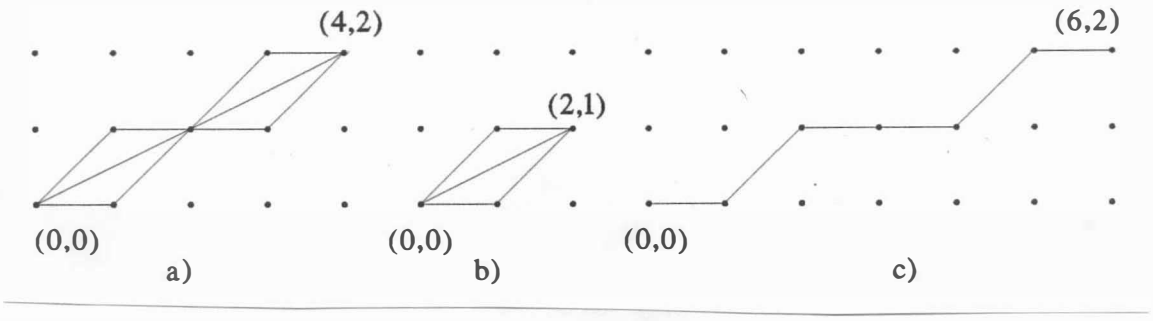
a) b)

Obr. 3.3.1

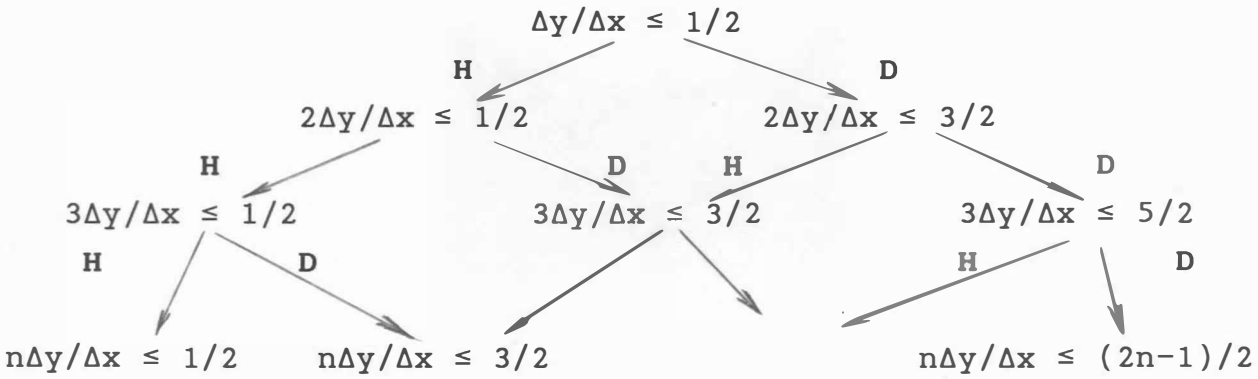
3.3 Bresenhamův algoritmus

Bresenhamův algoritmus, který byl původně navržen pro přírůstkové zapisovače, je též vhodný pro použití v rastrových zařízeních. Výhodou Bresenhamova algoritmu je výlučné použití pouze celočíselné reprezentace dat. Při odvození algoritmu se omezíme pouze na první oktant, tj. $0 \leq |\Delta y| \leq |\Delta x|$. Uvažme případ na obr. 3.3.1.a, kde **D** značí diagonální krok, **H** značí horizontální (vodorovnou) krok.

Je zřejmé, že mezní hodnotou pro rozhodování, zda provést horizontální, či diagonální krok, je hodnota $\Delta y/\Delta x = 0.5$. Jestliže podobnou úvahu provedeme i v následujícím kroku, obdržíme schéma pro postupné rozhodování, zdali zvolit krok diagonální **D**, či krok horizontální **H**, viz obr. 3.3.3. Uvedené schéma lze modifikovat tak, že není nutné používat operací v pohyblivé řádové čárce.

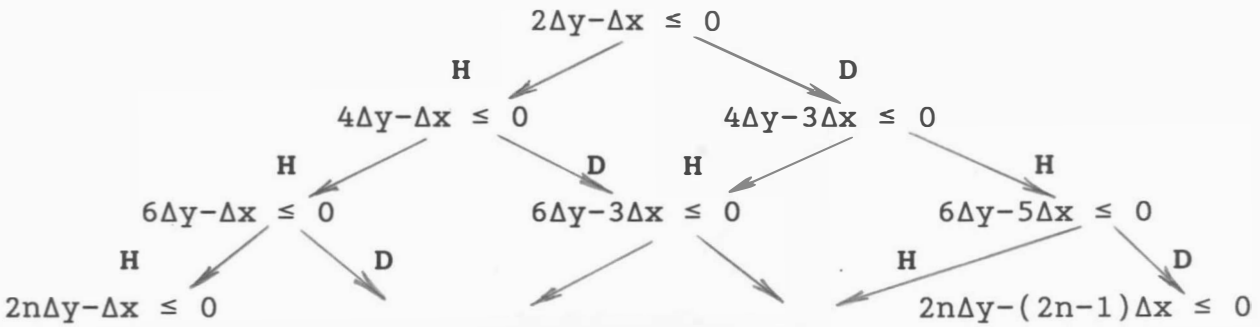


Obr. 3.3.2.



Obr. 3.3.3

Vynásobením výrazů hodnotou $2\Delta x$ dostáváme ($\Delta x > 0$):



Obr. 3.3.4

Algoritmus generace čáry lze pak popsat takto:

```
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;
```

Algoritmus 3.3.1

Uvažme případ z obr.3.3.2.c. Pak pro dané hodnoty x = 6 a y = 2 je $\Delta y/\Delta x = 1/3$ a podle algoritmu na obr.3.3.3 dostáváme posloupnost:

$$\frac{\Delta y}{\Delta x} = \frac{1}{3} \leq \frac{1}{2} \xrightarrow[+]{} \mathbf{H} \qquad \frac{2\Delta y}{\Delta x} = \frac{2}{3} \leq \frac{1}{2} \xrightarrow[-]{} \mathbf{D}$$

$$\frac{3\Delta y}{\Delta x} = \frac{3}{3} \leq \frac{3}{2} \xrightarrow[+]{} \mathbf{H} \qquad \frac{4\Delta y}{\Delta x} = \frac{4}{3} \leq \frac{3}{2} \xrightarrow[+]{} \mathbf{H}$$

$$\frac{5\Delta y}{\Delta x} = \frac{5}{3} \leq \frac{3}{2} \xrightarrow[-]{} \mathbf{D} \qquad \frac{6\Delta y}{\Delta x} = \frac{6}{3} \leq \frac{5}{2} \xrightarrow[+]{} \mathbf{H}$$

Počet kroků je roven hodnotě Δx. Získaná posloupnost elementárních kroků je "**HDHHDH**". Vynásobíme-li všechny zlomky v tomto algoritmu hodnotou $2\Delta x$, dostaneme algoritmus uvedený na obr.3.3.4. Označíme-li výraz $2\Delta y - \Delta x$ identifikátorem d, pak v případě kroku "**H**" musíme připočíst hodnotu a rovnou $2\Delta y$, v případě kroku "**D**" je nutné připočíst hodnotu b rovnou $2\Delta y - 2\Delta x$.

Vzhledem k tomu, že algoritmus používá pouze celočíselnou reprezentaci, je možné jej snadno realizovat integrovaným obvodem.

Odvození Bresenhamova algoritmu pro kreslení úsečky, které bylo uvedeno výše, bylo založeno na intuitivním přístupu. Pro pochopení a případné odvození dalších algoritmů je nezbytné i pochopení matematického přístupu k odvození. Předpokládejme situaci na obr.3.3.1.b, kde d_1, d_2 určují skutečnou vzdálenost přímky w pro $x = x_1$ od nejbližších bodů rastru a kde vzdálenost mezi body rastru je v obou směrech rovna jedné. Zobrazovaná úsečka leží na přímce w a má koncové body $\mathbf{x}_r$ a $\mathbf{x}_s$.

Pro přímku lze psát

$$y = m\,x + b$$

přičemž směrnice m je dána výrazem

$$m = \Delta y \,/\, \Delta x$$

Hodnotu b lze určit dosazením např. bodu $\mathbf{x}_r$ do rovnice, přičemž položíme $\mathbf{x}_0 = \mathbf{x}_r$, pak

$$y_0 = \frac{\Delta y}{\Delta x}\, x_0 + b$$

a

$$b = (\Delta x\, y_0 - \Delta y\, \mathbf{x}_0) \,/\, \Delta x$$

Pro krok i, a tedy i pro prvý, lze psát

$$y = m\,x_i + b = \frac{\Delta y}{\Delta x}\,x_i + b$$

Vzhledem k tomu, že bod (x_i,y) není obecně bodem daného rastru, je nutné pro prvý krok vybrat jeden bod z bodů (x_1,y_0) nebo (x_1,y_0+1). Pro volbu bodu je nutné zvolit kritérium, pomocí kterého rozhodneme, který bod vybrat. Přirozeným kritériem je výběr toho bodu rastru, který je nejblíže bodu (x_1,y), a to vzhledem k ose y, např. $d_1 - d_2$. Pak pro i-tý (i = 0) krok dostáváme

$$d_1 = y - y_i = m\,x_{i+1} + b - y_i = m\,(x_i + 1) + b - y_i$$

$$d_2 = y_{i+1} - y = y_i + 1 - m\,x_{i+1} - b$$

$$= y_i + 1 - m\,(x_i + 1) - b$$

Odečtením dostáváme

$$d_1 - d_2 = 2\,m\,(x_i + 1) - 2\,y_i + 2\,b - 1$$

dosazením za $m = \Delta y / \Delta x$ pak

$$\Delta x\,(d_1 - d_2) = 2\,\Delta y\,x_i - 2\,\Delta x\,y_i + 2\,\Delta y + 2\,b\,\Delta x - \Delta x$$

Označíme-li

$$p_i = \Delta x\,(d_1 - d_2)$$

pak lze psát

$$p_i = 2\,\Delta y\,x_i - 2\,\Delta x\,y_i + c$$

přičemž

$$c = 2\,\Delta y + \Delta x\,(2\,b - 1)$$

Je zřejmé, že konstantu c by bylo možné určit např. dosazením počátečního bodu úsečky. Dosazením za b do výrazu pro konstantu c dostáváme

$$c = 2\,\Delta y + \Delta x\,[\,(2\,\Delta x\,y_0 - 2\,\Delta y\,x_0) / \Delta x - 1\,]$$

$$= 2\,\Delta y + 2\,\Delta x\,y_0 - 2\,\Delta y\,x_0 - \Delta x$$

Dosadíme-li nyní za c do rovnice určující hodnotu p_i, pak dostáváme:

$$p_i = 2\,\Delta y\,x_i - 2\,\Delta x\,y_i + 2\,\Delta y + 2\,\Delta x\,y_0 - 2\,\Delta y\,x_0 - \Delta x$$

Úpravou pro i = 0 dostáváme

$$p_0 = 2\,\Delta y - \Delta x$$

Nyní je nezbytné určit hodnotu p_{i+1} v závislosti na hodnotě p_i. Obecně je možné psát, že:

$$p_{i+1} - p_i = 2\,\Delta y\, x_{i+1} - 2\,\Delta x\, y_{i+1} + c$$

$$- 2\,\Delta y\, x_i + 2\,\Delta x\, y_i - c$$

Úpravou a dosazením za x_{i+1} , kde $x_{i+1} = x_i + 1$, dostáváme

$$p_{i+1} - p_i = 2\,\Delta y - 2\,\Delta x\,(\, y_{i+1} - y_i\,)$$

a tedy lze psát, že:

$$p_{i+1} = p_i + 2\,\Delta y - 2\,\Delta x\,(\, y_{i+1} - y_i\,)$$

Ze zvoleného kritéria vyplývá, že pokud bude $d_1 - d_2 \leq 0$, pak bude vybrán krok horizontální, tj. bude zvolen bod $(x_i + 1, y_i)$. Pak

$$p_1 = p_0 + 2\,\Delta y$$

Pokud bude $d_1 - d_2 > 0$, pak bude vybrán krok diagonální, tj. bude zvolen bod $(x_i + 1, y_i + 1)$. Pak

$$p_1 = p_0 + 2\,\Delta y - 2\,\Delta x$$

Poznámka

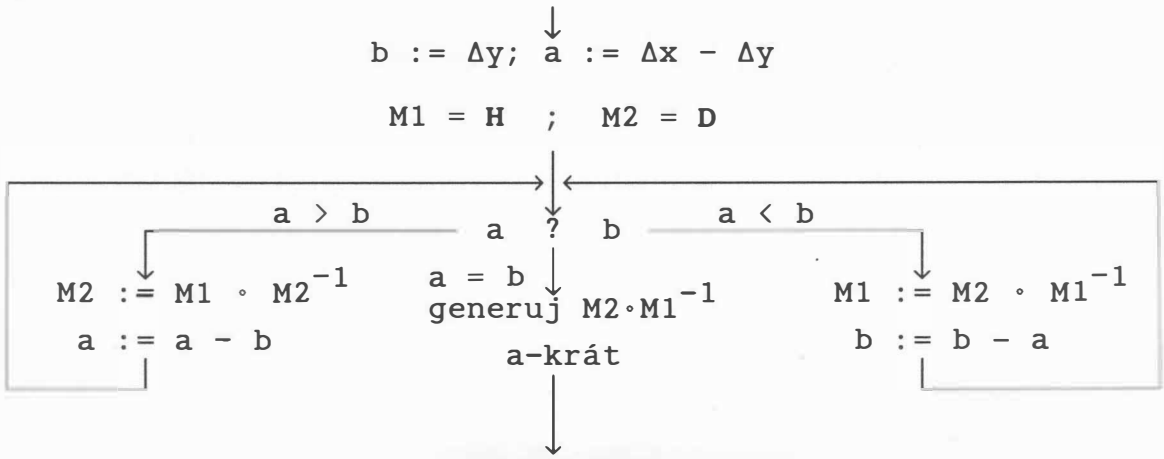
Z uvedeného je zřejmé, že hodnotě p_i odpovídá identifikátor d v algoritmech 3.3.1 a 3.3.2.

Bresenhamův algoritmus pro generaci čáry musí být nyní modifikován pro všechny oktanty, aby mohl být použit pro případ obecné polohy úsečky. Všechny dosud známé algoritmy mají nevýhodu, že nejsou obecně symetrické. Uvážíme-li případ z obr. 3.3.2.a, pak snadno zjistíme, že posloupnost bude jiná v případě kreslení z bodu (0,0) do bodu (4,2) než posloupnost při kreslení z bodu (4,2) do bodu (0,0) - naznačeno čárkovaně. Tuto nevýhodu nelze v zásadě eliminovat, neboť jak je ukázáno na obr. 3.3.2.b, jsou možné pouze dva způsoby propojení bodů (0,0) a (2,1), a to "horem" nebo "dolem". Který způsob použijeme, je závislé na operátoru v podmínce, tj. zdali použijeme operátor < nebo <= při porovnávání chyby d.

```
procedure BRESENHAM ( x1, y1, x2, y2: integer);
var ix, iy, j, d, a, b, dx, dy, xp, yp: integer;
begin xp := x1; yp := y1;
      ix := SIGN ( x2 - x1 ); iy := SIGN ( y2 - y1 );
      dx := abs ( x2 - x1 ); dy := abs ( y2 - y1 );
      if dx >= dy then {1.,4.,5.,8. oktant }
      begin a := 2 * dy;  d := a - dx;  b := a - 2 * dx;
            DRAW STEP*;
            for j := 1 to dx do
            begin if d <= 0 then d := d + a
                        else begin yp := yp + iy;
                                   d := d + b
                             end;
                  xp:=xp+ix;
                  DRAW STEP
            end
      end
      else    {  2.,3.,6.,7. oktant }
      begin a := 2 * dx;  d := a - dy;  b := a - 2 * dy;
            DRAW STEP*;
            for j := 1 to dy do
            begin if d <= 0 then d := d + a
                  else begin xp := xp + ix; d := d + b end
                  yp := yp + iy;
                  DRAW STEP
            end
      end
end;
{ příkazy označené * jen v případě zařízení s přímým }
{ přístupem - jsou nutné k aktivaci počátečního bodu }
```

Algoritmus 3.3.2

Zajímavý algoritmus pro generaci čáry, viz [22], pro případ $0 < \Delta y < \Delta x$ je uveden na obr. 3.3.5.



kde M^{-1} označuje inverzi řetězce M a ∘ označuje zřetězení

Obr. 3.3.5

Činnost algoritmu lze demonstrovat na příkladě kreslení čáry z bodu (0,0) do bodu (51,11) následující tabulkou hodnot:

a	b	M1	M2
40	11	H	HD
29	11	H	HDH
18	11	H	HHDH
7	11	HHDHH	HHDH
7	4	HHDHH	HHDHHHDHH
3	4	HHDHHHDHHHHDHH	HHDHHHDHH
3	1	HHDHHHDHHHHDHH	HHDHHHDHHHHDHHHHDHHHDHH
2	1	HHDHHHDHHHHDHH	HHDHHHDHHHHDHHHHDHHHDHHHHDHHHDHHHDHH
1	1	generuje výslednou posloupnost kroků	

HHDHHHDHHHHDHHHHDHHHDHHHHDHHHHDHHHDHHHHDHHHHDHHHDHH

kterou lze zapsat

$$H^2DH^3DH^4DH^4DH^3DH^4DH^4DH^3DH^4DH^4DH^3DH^2$$

Je vhodné zde poznamenat, že celá posloupnost byla vygenerována pouze s testováním devíti podmínek, avšak operace zřetězení a inverze je poměrně časově náročná.

3.4 Generátor kružnice

V některých aplikacích, zejména pak z oblasti kreslení mechanických částí, je zapotřebí vykreslovat kružnici, resp. její část. V praxi bylo použito mnoho algoritmů, které aproximovaly kružnice pomocí přímkových úseků. Jedním z nejprimitivnějších algoritmů je přímá aplikace parametrických rovnic kružnice:

$$x = R * \cos(t) \qquad y = R * \sin(t)$$

$t \in \langle 0 , 2\pi)$, kde souřadnice bodu (x,y) jsou počítány s malým krokem parametru t pro t=0, t=Δt, ... a jednotlivé body jsou spojovány přímkovými úseky. Tento způsob generování kružnice je neobyčejně náročný pro opětovné výpočty hodnot goniometrických funkcí cos(t) a sin(t).

Kruhový oblouk lze též snadno aproximovat lomenou čárou. Je-li R poloměr kružnice a d povolená odchylka, pak lze spočítat souřadnice lomené čáry aproximující kružnici

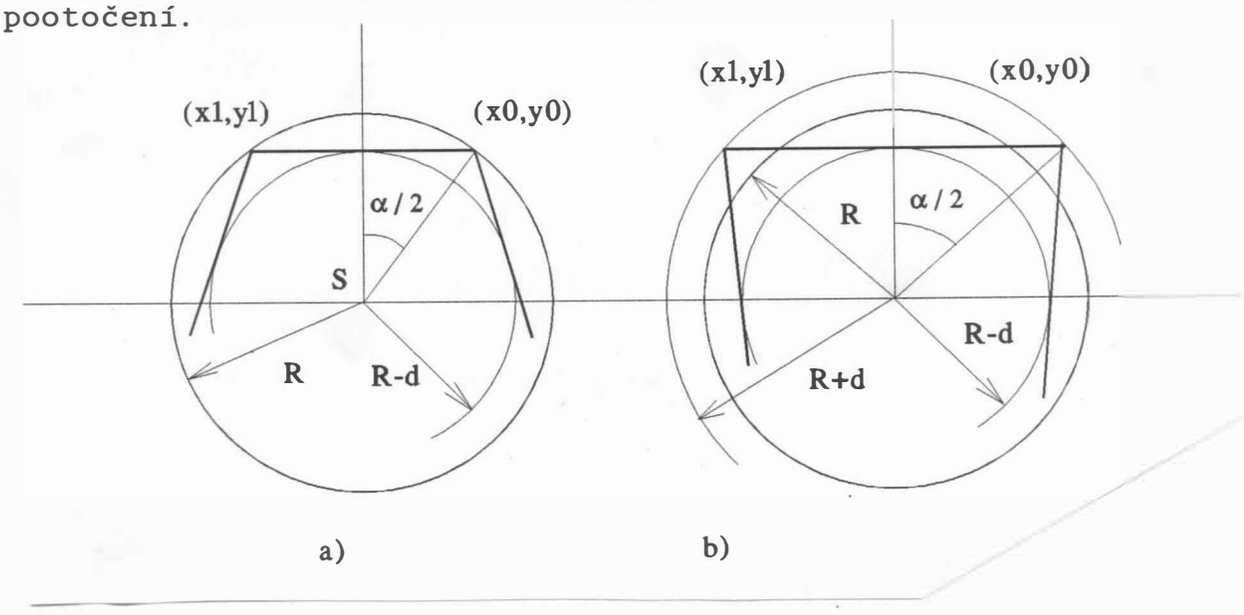
$$x_{i+1} = x_i \cos\alpha - y_i \sin\alpha$$

$$y_{i+1} = x_i \sin\alpha + y_i \cos\alpha \qquad i=0,1,...$$

nebo v maticovém tvaru

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad i=0,1,....$$

kde x_i, y_i jsou počáteční souřadnice lomené čáry a α úhel pootočení.



Obr. 3.4.1

Pro vnitřní nahrazení kružnice lomenou čárou, obr.3.4.1.a, platí:

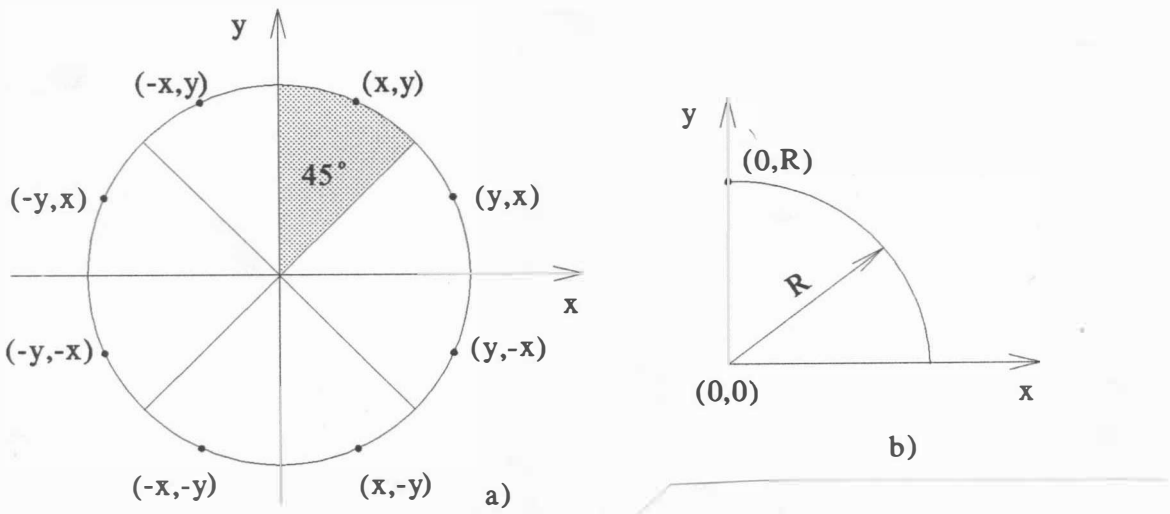
$$\cos \frac{\alpha}{2} = \frac{R - d}{R}$$

V případě nahrazení kružnice lomenou čárou s odchylkou ± d, obr.3.4.1.b, platí

$$\cos \frac{\alpha}{2} = \frac{R - d}{R + d}$$

přičemž bod (x_0, y_0) není bodem kružnice.

Výhodou uvedeného přístupu je, že hodnota goniometrických funkcí se počítá pouze jednou. Nicméně výše uvedený algoritmus je stále příliš složitý. Pro rastrové prostředí lze kružnice generovat pomocí Bresenhamova algoritmu. Poznamenejme, že bude generován pouze 1. oktant a ostatní oktanty se získají např. pomocí symetrie, viz obr.3.4.2.



Obr.3.4.2

K odvození činnosti Bresenhamova generátoru kružnice uvážíme 1. kvadrant a budeme předpokládat, že střed kružnice je v počátku souřadnic. Poznamenejme, že začne-li algoritmus v bodě x = 0, y = R, pak souřadnice y je monotónně klesající funkcí proměnné x v 1. kvadrantu, a jsou tedy možné pouze tři směry: vertikální m_V , diagonální m_D a horizontální m_H, viz obr.3.4.3.

Algoritmus by měl vybrat takový následující pixel, který je nejblíže ke skutečnému průběhu kreslené kružnice. Označíme-li jednotlivé směry výrazy

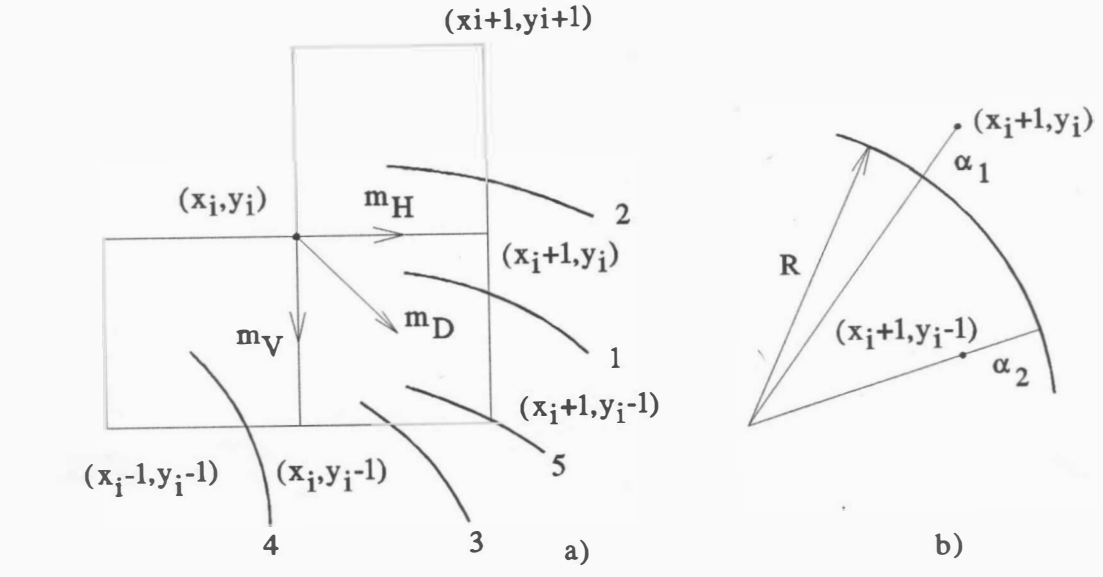
$$m_H = | (x_i + 1)^2 + y_i^2 - R^2 |$$

$$m_D = | (x_i + 1)^2 + (y_i - 1)^2 - R^2 |$$

$$m_V = | x_i^2 + (y_i - 1)^2 - R^2 |$$

pak bude vybrán ten směr, pro který výraz z uvedených tří výrazů nabývá minimální hodnoty.

Vzhledem k rastrovému prostředí existuje pouze pět možných situací průběhu kružnice v rastru.



Obr. 3.4.3

Označme Δ_i rozdíl kvadrátu vzdálenosti bodu $(x_i + 1, y_i - 1)$ od počátku kružnice a kvadrátu poloměru kružnice R , pak:

$$\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

Podobně jako u Bresenhamova algoritmu pro kreslení přímé čáry i v případě generátoru kružnice je podstatné znaménko chyby kreslení. Jestliže $\Delta_i < 0$, pak bod $(x_i + 1, y_i - 1)$ je uvnitř kružnice, tj. jde o případ 1. nebo 2. Je zřejmé, že musí být vybrán buď bod $(x_i + 1, y_i - 1)$,tj. směr m_D , nebo bod $(x_i + 1, y_i)$, tj. směr m_H. K rozhodnutí, jaká možnost se má vybrat, je nutné zjistit, který z výrazů m_H a m_D je v absolutní hodnotě menší, resp. určit znaménko výrazu:

$$\delta = |(x_i + 1)^2 + y_i^2 - R^2| - |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$

Je-li $\delta < 0$, tj. $| \alpha_1 | < | \alpha_2 |$, pak musí být vybrán bod $(x_i + 1, y_i)$,tj. krok m_H. Opačně, je-li $\delta > 0$, tj. $| \alpha_1 | > | \alpha_2 |$, pak musí být vybrán bod $(x_i + 1, y_i - 1)$, tj. krok m_D. Když vzdálenosti α_1 a α_2 jsou si rovné, vybereme horizontální krok. Tedy je-li:

$\delta \leq 0$, vyber krok m_H , tj. bod $(x_i + 1, y_i)$

$\delta > 0$, vyber krok m_D , tj. bod $(x_i + 1, y_i - 1)$

Uvedené výrazy jsou poměrně komplikované. Je tedy vhodné převést výrazy pro δ na takový tvar, kdy není zapotřebí operací násobení a dělení.

Pro případ 1. platí, že

$$(x_i + 1)^2 + y_i^2 - R^2 \geq 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0$$

neboť platí, že bod $(x_i + 1, y_i)$ je vždy vně kružnice, zatímco bod $(x_i + 1, y_i - 1)$ je uvnitř kružnice. Pak výraz pro δ může být přepsán na

$$\delta = (x_i + 1)^2 + y_i^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

po úpravě dostaneme:

$$\delta = 2 [(x_i + 1)^2 + (y_i - 1)^2 - R^2 + y_i] - 1$$

Použijeme-li nyní výraz pro výpočet Δ_i, dostáváme

$$\delta = 2 (\Delta_i + y_i) - 1$$

což je podstatně jednodušší výraz (násobení dvěma se realizuje jako posuv vlevo).

Uvážíme-li případ 2., pak vzhledem k monotónně klesajícímu průběhu funkce, musí být vybrán bod $(x_i + 1, y_i)$. Pak tedy platí

$$(x_i + 1)^2 + y_i^2 - R^2 < 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0$$

neboť body $(x_i+ 1, y_i)$ a $(x_i+ 1, y_i- 1)$ leží uvnitř kružnice. Tudíž $\delta < 0$ a bod $(x_i+ 1, y_i)$ je vybrán podle stejného kritéria jako pro případ 1.

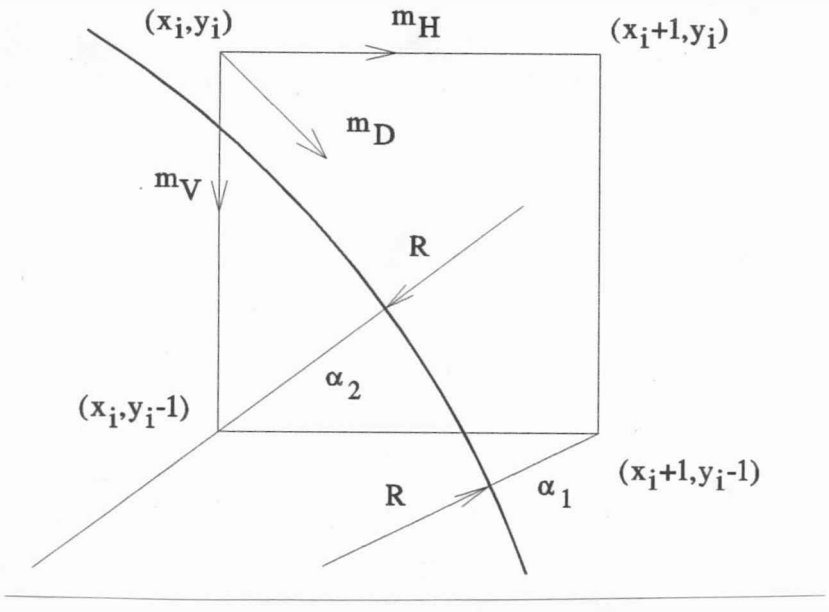
Jestliže $\Delta_i > 0$, pak bod $(x_i+ 1, y_i- 1)$ je vně skutečné kružnice, tj. mohou nastat pouze případy 3., resp. 4. Je zřejmé, že musí být vybrán buď bod o souřadnicích $(x_i, y_i- 1)$, tj. směr m_V , nebo bod $(x_i+ 1, y_i- 1)$, tj. směr m_D . Analogicky jako v předešlém textu se budeme snažit vybrat bod, který způsobí nejmenší chybu při zobrazení.
Pak podle obr. 3.4.4

$$\delta' = | \alpha_1 | - | \alpha_2 |$$

a tedy:

$$\delta' = | (x_i + 1)^2 + (y_i - 1)^2 - R^2 |$$

$$- | x_i^2 + (y_i - 1)^2 - R^2 |$$



Obr. 3.4.4

Jestliže $\delta' < 0$, pak $|\alpha_2| > |\alpha_1|$, a tedy bude vybrán krok m_D. V případě, že $\delta' > 0$, pak $|\alpha_2| < |\alpha_1|$ a musí být vybrán krok m_V . Pro případ $\delta' = 0$ bude vybrán krok m_D .

Tedy je-li:

$$\delta' \leq 0, \text{ vyber krok } m_D, \text{ tj. bod } (x_i + 1, y_i - 1)$$

$$\delta' > 0, \text{ vyber krok } m_V, \text{ tj. bod } (x_i, y_i - 1)$$

Uvedené podmínky pro δ' je opět nutné zjednodušit.

Pro případ 3 platí, že

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 \geq 0$$

$$x_i^2 + (y_i - 1)^2 - R^2 < 0$$

neboť bod $(x_i + 1, y_i - 1)$ je vždy vně skutečné kružnice. Pak výraz pro δ' může být upraven na tvar

$$\delta' = (x_i + 1)^2 + (y_i - 1)^2 - R^2 + x_i^2 + (y_i - 1)^2 - R^2$$

Po úpravě dostaneme

$$\delta' = 2 [(x_i + 1)^2 + (y_i - 1)^2 - R^2 - x_i] - 1$$

Použijeme-li nyní výraz pro výpočet Δ_i , platí:

$$\delta' = 2 (\Delta_i - x_i) - 1$$

Uvážíme-li nyní případ 4, pak vzhledem k tomu, že funkce je monotónně klesající, musí být vybrán bod $(x_i, y_i - 1)$, tj. směr m_V. Je zřejmé, že:

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 > 0$$

$$x_i^2 + (y_i - 1)^2 - R^2 > 0$$

neboť body $(x_i, y_i - 1)$ a $(x_i + 1, y_i - 1)$ jsou vně skutečné kružnice, tedy $\delta' > 0$, a je vybrán bod $(x_i, y_i - 1)$, tj. směr m_V. Nyní zbývá pouze rozhodnout případ 5, který nastává pouze tehdy, prochází-li skutečná kružnice bodem $(x_i + 1, y_i - 1)$, tj. $\Delta_i = 0$. Pro výpočet jednotlivých komponent δ platí

$$(x_i + 1)^2 + y_i^2 - R^2 > 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0$$

a tedy $\delta > 0$ a byl by vybrán bod $(x_i + 1, y_i - 1)$, tj. směr m_D. Analogicky pro komponenty δ' dostáváme

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0$$

$$x_i^2 + (y_i - 1)^2 - R^2 < 0$$

a $\delta' < 0$, což je podmínka pro výběr bodu $(x_i + 1, y_i - 1)$, tj. směr m_D. Tudíž pro případ $\Delta_i = 0$ platí stejná kritéria jako pro případ $\Delta_i > 0$ nebo $\Delta_i < 0$.

Shrneme-li předchozí výsledky, pak dostáváme:

$\Delta_i < 0$	$\delta \leq 0$	vyber bod $(x_i + 1, y_i)$, tj. směr m_H
$\Delta_i < 0$	$\delta > 0$	vyber bod $(x_i + 1, y_i - 1)$, tj. směr m_D
$\Delta_i = 0$		vyber bod $(x_i + 1, y_i - 1)$, tj. směr m_D
$\Delta_i > 0$	$\delta' \leq 0$	vyber bod $(x_i + 1, y_i - 1)$, tj. směr m_D
$\Delta_i > 0$	$\delta' > 0$	vyber bod $(x_i, y_i - 1)$, tj. směr m_H

Nyní lze poměrně snadno odvodit celý algoritmus. Abychom nemuseli pro každý krok stále počítat výraz Δ_i , δ , resp. δ' , pokusme se vyjádřit výrazy jednodušeji. Předpokládejme nejdříve, že se má pokračovat horizontálním krokem m_H. Pak pro (i + 1)-vý bod platí:

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i$$

$$\begin{aligned}\Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 \\ &= (x_{i+1} + 1)^2 + (y_i - 1)^2 - R^2 \\ &= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_{i+1} + 1 \\ &= \Delta_i + 2x_{i+1} + 1\end{aligned}$$

Podobně pro diagonální krok m_D:

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i - 1$$

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2$$

Pro vertikální krok m_V dostáváme:

$$x_{i+1} = x_i$$

$$y_{i+1} = y_i - 1$$

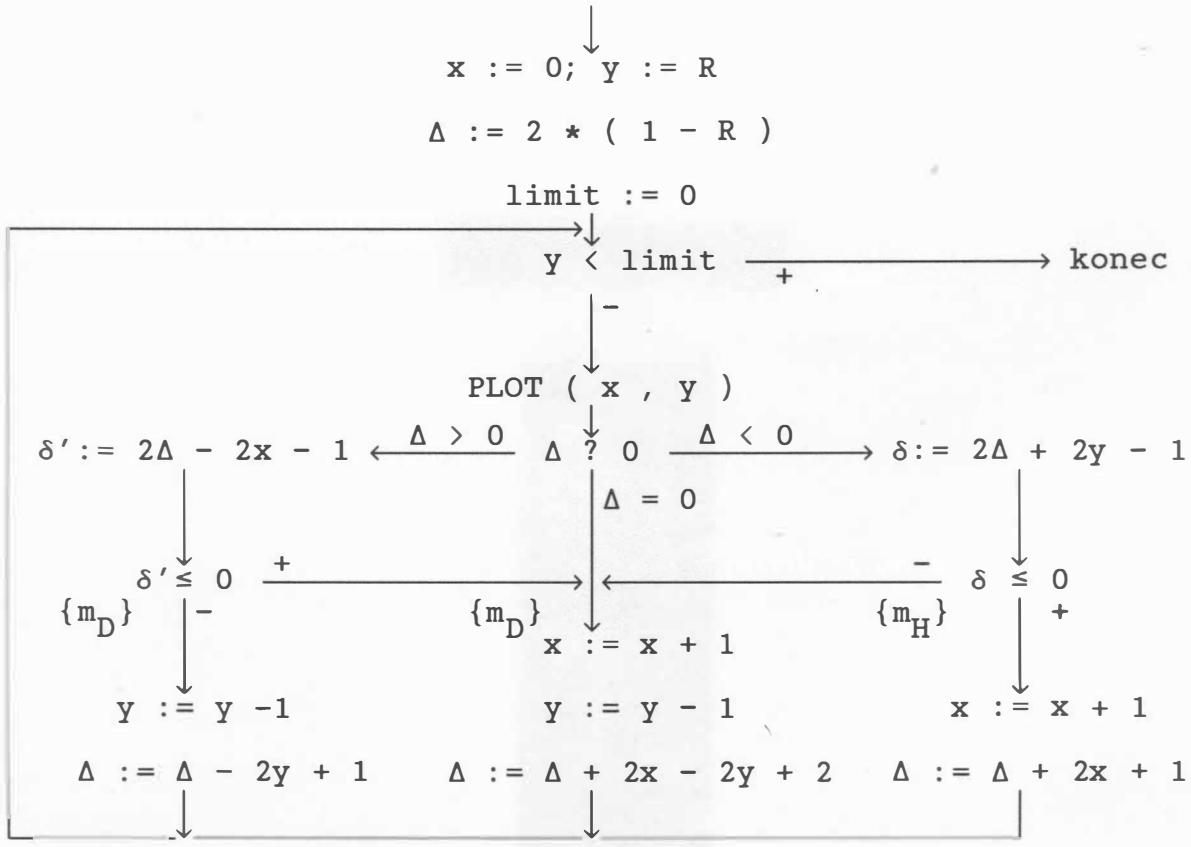
$$\Delta_{i+1} = \Delta_i - 2y_{i+1} + 1$$

Vzhledem k tomu, že Δ_{i+1} je závislé pouze na Δ_i, tj. na hodnotě v předcházejícím kroku, lze indexy eliminovat.

Výsledný algoritmus lze znázornit vývojovým diagramem, kde:

$$\Delta = (1)^2 + (R - 1)^2 - R^2 = 1 + R^2 - 2R + 1 - R^2$$

$$= 2 (1 - R)$$

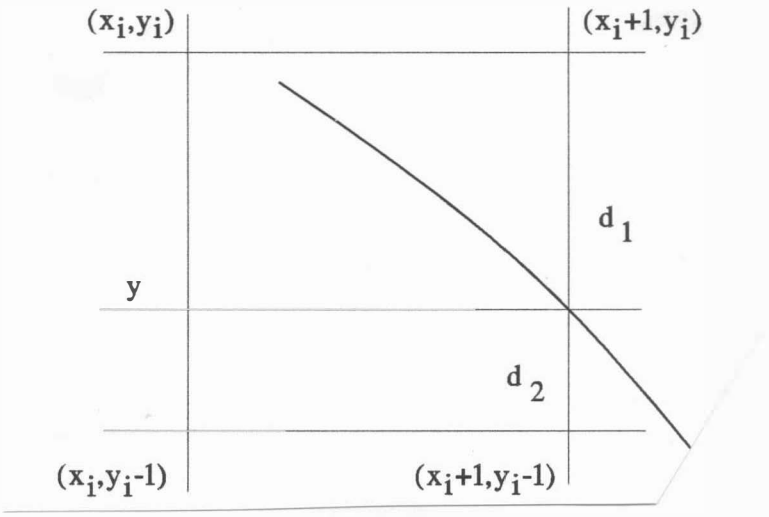


Obr. 3.4.5

V případě, že chceme generovat kružnici jen pro oktant, je nutné nastavit LIMIT na hodnotu trunc($R/\sqrt{2}$).

Další zjednodušení je možné, omezíme-li se pouze na generaci kružnice v jednom oktantu. Na rozdíl od původního Bresenhamova algoritmu, viz obr. 3.4.3, který bral v úvahu možnosti 1 - 5 , mohou nastat pouze případy 1, 2 a 5.

Předpokládejme, že kružnice má střed v počátku souřadného systému a její poloměr je roven r. Chtějme generovat kruhový oblouk z bodu (0,r) do bodu, kdy je x = y. Označme opět vzdálenost skutečného bodu kružnice od odpovídajících bodů rastru d_1 a d_2, viz obr. 3.4.6.



Obr. 3.4.6

Obecně pro kružnici se středem v počátku platí, že

$$x^2 + y^2 - r^2 = 0$$

a tedy je-li x proměnná, jejíž hodnoty volíme, pak pro krok i lze psát

$$y^2 = r^2 - x_i^2$$

Pro d_1 a d_2 pak platí

$$d_1 = y_i^2 - y^2 = y_i^2 - r^2 + (x_i + 1)^2$$

$$d_2 = y^2 - (y_{i-1})^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2$$

Označíme-li

$$p_i = d_1 - d_2$$

pak dostáváme

$$p_i = 2 (x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2 r^2$$

Dosadíme-li do výrazu pro p_i souřadnice počátečního bodu (0,r), pak dostáváme

$$p_0 = 3 - 2 r$$

Pro rekurentní výpočet je nutné určit p_{i+1} jako funkci p_i, resp. souřadnic v předchozím kroku. Vyjádříme-li p_{i+1}, pak

$$p_{i+1} = 2 [x_{i+1} + 1]^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r$$

$$= 2 [(x_i + 1) + 1]^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r$$

Po pronásobení a úpravě dostáváme

$$p_{i+1} = p_i + 4\,x_i + 6 + 2\,(\,y_{i+1}^2 - y_i^2\,) - 2\,(\,y_{i+1} - y_i\,)$$

Z uvedeného vyplývá, že pokud $p_i < 0$, je vybrán krok horizontální, a tedy po dosazení bodu $(x_i + 1, y_i)$ dostáváme

$$p_{i+1} = p_i + 4\,x_i + 6$$

zatímco je-li $p_i \geq 0$, je vybrán krok diagonální, a tedy po dosazení bodu $(x_i + 1, y_i - 1)$ dostáváme

$$p_{i+1} = p_i + 4\,(\,x_i - y_i\,) + 10$$

Zjednodušený algoritmus, který byl navržen Michnerem, může být popsán takto:

```
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;
```

Algoritmus 3.4.1

Poznamenejme, že uvedená procedura generuje pouze jeden oktant a využívá symetrie k zobrazení celého kruhového oblouku.

Poznámka

Z uvedeného je zřejmé, že hodnotě p_i odpovídá proměnná d v algoritmu 3.4.1.

3.5 Generace kuželoseček

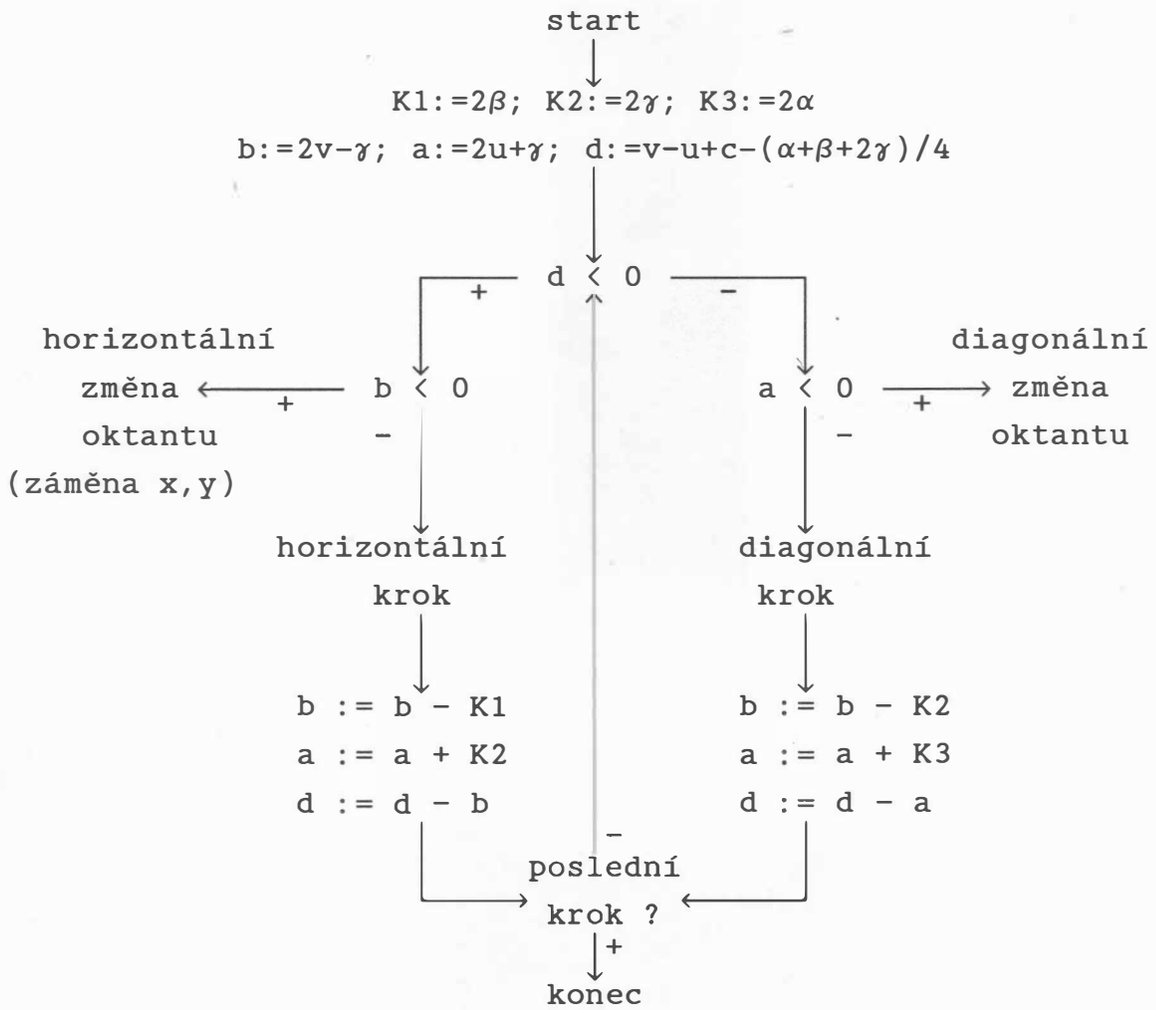
V literatuře lze nalézt mnoho více či méně důmyslných algoritmů, jak generovat kuželosečky v rastrovém prostředí. Kuželosečky mohou být popsány v kartézském souřadném systému implicitní rovnicí $d(x,y)=0$, kde:

$$d(x,y) = 2vx - 2uy + c - \beta x^2 - \alpha y^2 - 2\gamma xy$$

Pak:

$$\frac{dy}{dx} = \frac{v - \beta x - \gamma y}{u + \gamma x + \alpha y}$$

Kuželosečka může být generována např. pomocí Pittewayova algoritmu [39]:



Obr. 3.5.1

Použití algoritmu z obr. 3.5.1 omezuje řešení úlohy, jak určit jednotlivé koeficienty tak, abychom obdrželi nejlepší aproximaci kuželosečky dané s neceločíselnými koeficienty.

3.6 Kódování grafické informace

S rozvojem grafických systémů založených nikoliv na principu postupného vykreslování vektorů, ale na principu zobrazování po sobě jdoucích řádek (podobně jako u televizního přijímače). Vyvinuly se nové techniky, které umožňují efektivní zobrazení grafické informace. V zásadě byly vyvinuty čtyři základní techniky, a to:

- kódování délky běhu (run length encoding)
- kódování buněk
- frame buffer
- řádková konverze (scan line conversion)

Kódování délky běhu

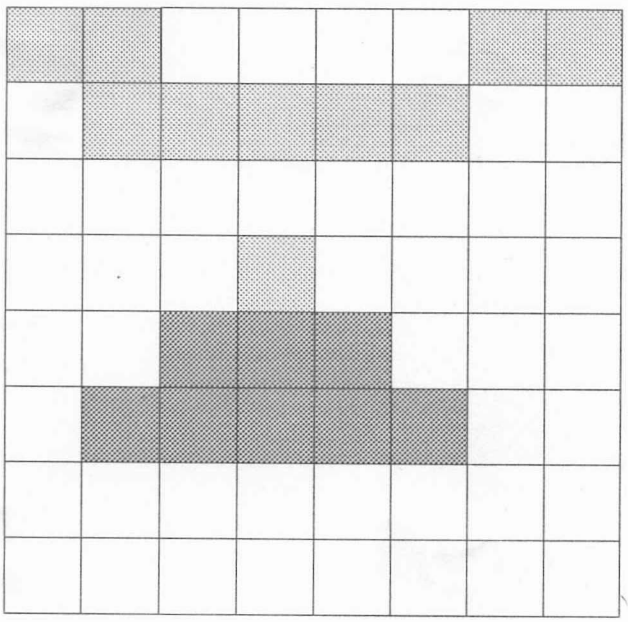
Technika kódování délky běhu (run length encoding) využívá tu skutečnost, že velké oblasti obrázku mají často stejnou intenzitu či barvu. V nejjednodušším tvaru kódování délky běhu určuje jen intenzitu a počet po sobě jdoucích pixelů s touto intenzitou v rámci dané řádky. Pro případ barevných výstupů může být tato technika rozšířena tak, že se místo intenzity udává barva, např. pomocí složek RGB. Je zřejmé, že v případě větších obrázků dochází i ke kompresi dat. Tato technika má nicméně i své nevýhody, které jsou důsledkem sekvenčního uspořádání výsledné sekvence, a to:

- obtížné vkládání a rušení,
- s rostoucí složitostí rostou i nároky na paměť, která může být větší než vlastní frame-buffer.

Kódování obsahu buněk

Technikou kódování buněk (cell encoding) se reprezentuje určitá část obrázku jako celek. Na rozdíl od předchozí techniky, která v podstatě zvažovala informace pouze v jednom směru, technika kódování obsahu buněk bere v úvahu i okolí. Tato technika byla používána zejména u tzv. pseudografických displayů, které umožňovaly zobrazovat znaky (většinou max. 256), které však mohly být definovány uživatelem. Předpokládá se např., že obraz 640x480 může být rozdělen do 4800 políček velikosti 8x8, přičemž obsah políček je možné definovat. Obsah

políčka velikosti n x n je možné reprezentovat pomocí vzorů, kterých je však 2^{n^2}, tj. pro n=8 dostáváme přibližně $1.8*10^{19}$. Tento počet je možné snížit pomocí posuvu, zrcadlení a maskování na 108 vzorů při rozměru políčka 8x8, viz [9]. Tato technika je použitelná i pro případ barevného výstupu, avšak kompresní poměr je již menší. Jistým problémem je pak interakce s takto definovaným obrázkem.



[1,2],[0,4],[1,2]

[0,1],[1,5],[0,2]

[0,8]

[0,3],[1,1],[0,4]

[0,2],[2,3],[0,3]

[0,1],[2,5],[0,2]

[0,8]

[0,8]

kde [x,y] = [intenzita,počet po sobě jdoucích pixelů]

Obr. 3.6.1

Frame buffer

Technika frame-bufferu využívá velký a spojitý paměťový prostor, který slouží k uložení grafické informace pixel po pixelu. Vzhledem k rychle klesající ceně paměťových prvků se používá tato technika nejčastěji ve spojení s osobními počítači a grafickými stanicemi. Samotný frame buffer může být realizován pomocí speciální paměti grafického procesoru, což umožňuje podstatně zvýšit výkonnost systému, anebo jako součást paměti počítače, viz [45],[109].

Frame-buffer je obvykle realizován pomocí spojitého úseku paměti, který si lze představit jako vektor. Je-li r_x, resp. r_y, rozlišovací schopnost ve směru osy x, resp. osy y, a je-li počáteční adresa vektoru B_0, pak adresa zápisu pro pixel s pozicí (x,y) je dána výrazem:

$$B(x,y) = B_0 + r_x \cdot y + x$$

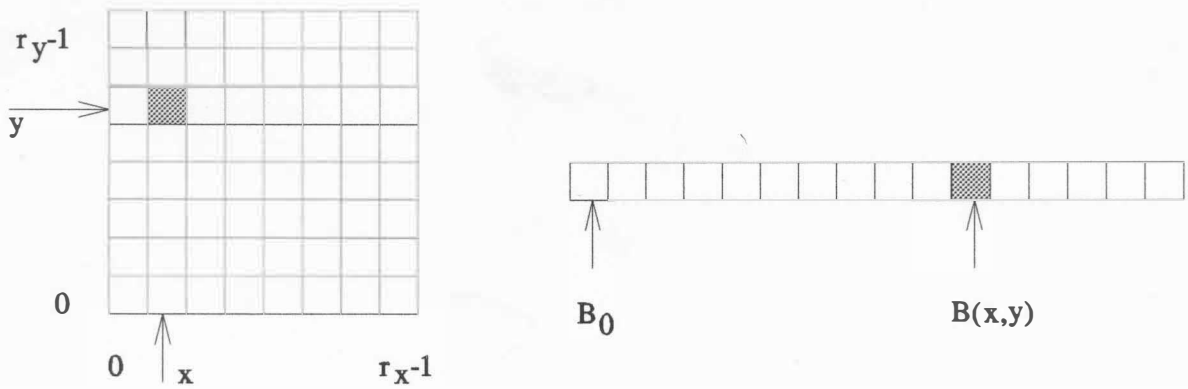
přičemž $x \in \{ 0, \ldots, r_x - 1 \}$ a $x \in \{ 0, \ldots, r_y - 1 \}$

Vzhledem k tomu, že základní algoritmy pro kreslení úseček, oblouků atd. potřebují obvykle aktivovat sousední pixely, je vhodné poznamenat, že:

$$B(x \pm 1, y) = B_0 + r_x . y + x \pm 1 = B(x,y) \pm 1$$

$$B(x, y \pm 1) = B_0 + r_x . (y \pm 1) + x = B(x,y) \pm r_x$$

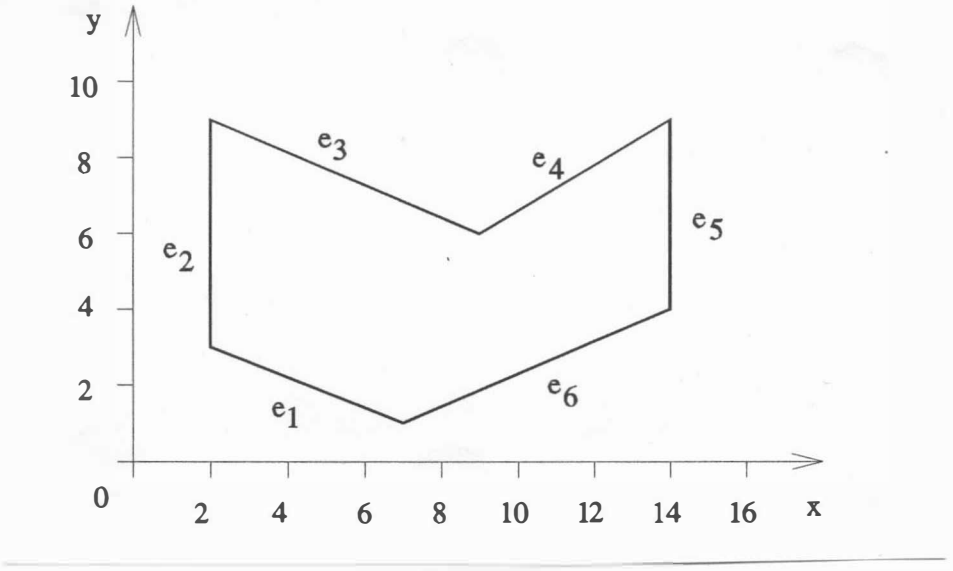
Analogicky lze určit i adresu bodu B(x±1,y±1) reprezentující bod (x±1,y±1) v závislosti na znalosti adresy pixelu (x,y). Pro efektivní implementaci Bresenhamova algoritmu je nutné využít výše uvedené vztahy.



Obr. 3.6.2

Řádková konverze

Řádková konverze (scan line conversion) je technikou, která umožňuje poměrně efektivní reprezentaci obrazu s tím, že se jednoduše určí průsečíky úseček, resp. hran n-úhelníka s každou řádkou, tj. s přímkou y = konst. Navíc je možné realizovat efektivně i další operace nad danou datovou strukturou. Pro názornost uvažme jednoduchý případ n-úhelníka z obr. 3.6.3.



Obr. 3.6.3

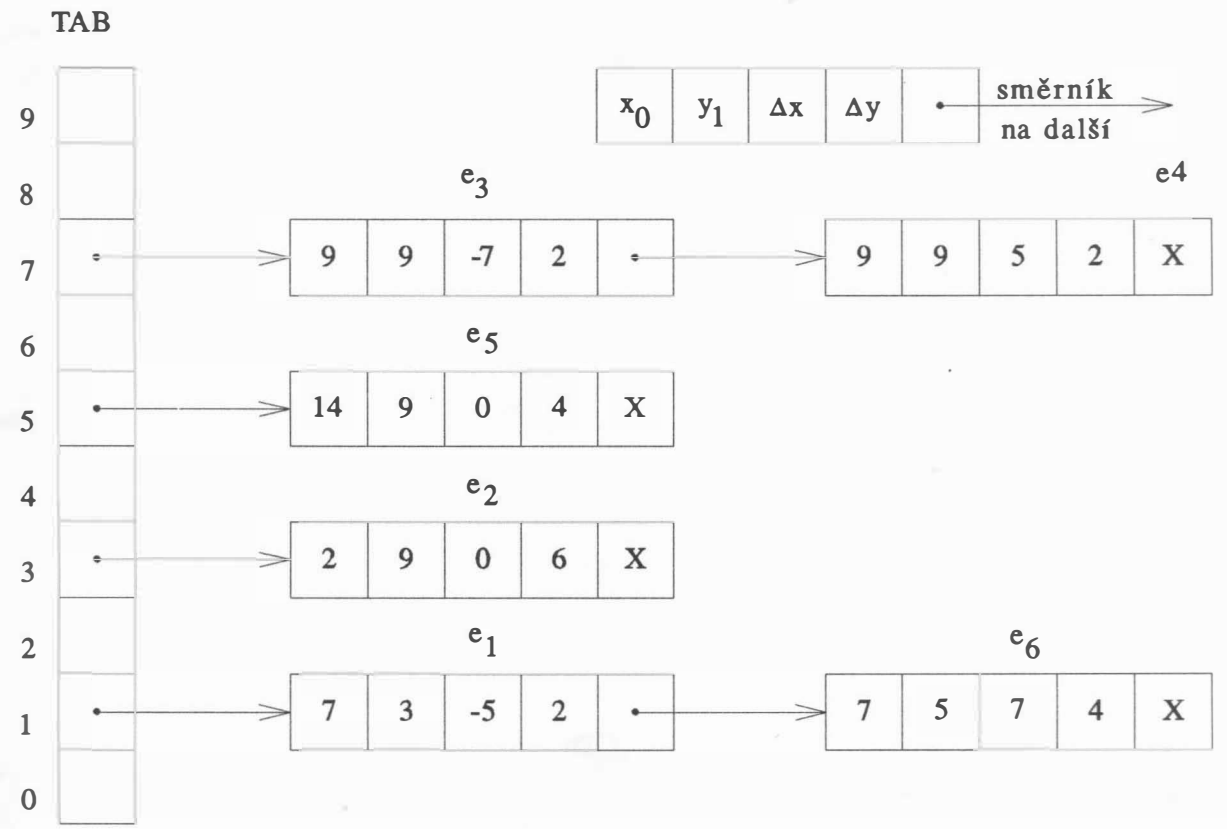
Vyjádříme-li rovnici přímky, na které leží daná úsečka, pak ($\Delta y=1$):

$$x_{i+1} = \frac{1}{m} + x_i = \frac{\Delta x}{\Delta y} + x_i$$

Pro n-úhelník z obr. 3.6.3 dostáváme následující n-tice (za x_0 vezmeme ten koncový bod, který má minimální souřadnici y).

hrana	$[\ y_0\ ,\ x_0\ ,\ y_1\ ,\ \Delta x\ ,\ \Delta y\ ]$
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]

Je zřejmé, že z důvodů rychlosti musí být ještě zvolena vhodná datová struktura, která umožňuje rychlé vyhledávání těch hran, které daná řádka protíná. Jedním z možných řešení je datová struktura, která je zobrazena na obr. 3.6.4.



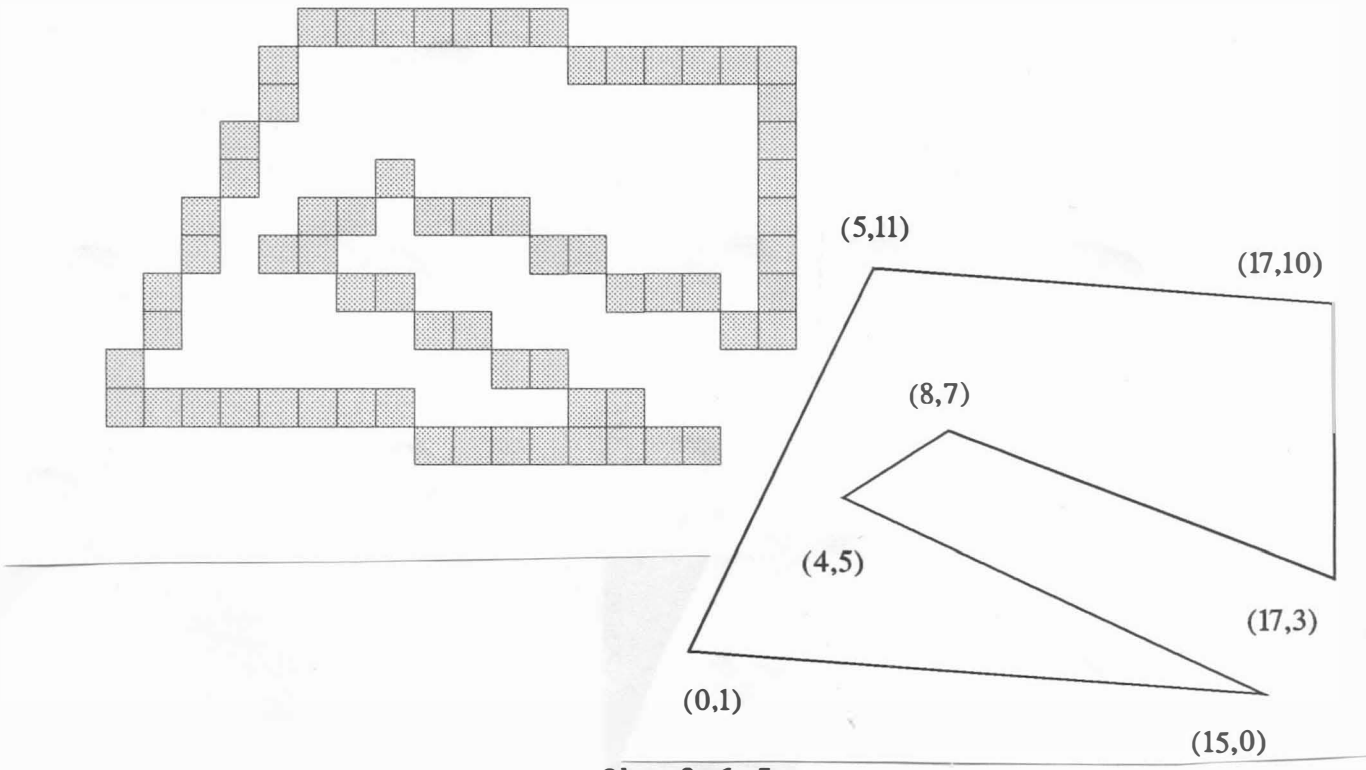
Obr. 3.6.4

Uvedená datová struktura je výhodná z několika důvodů, zejména však proto, že umožňuje velmi rychle nalézt ty hrany, které začínají na daném řádku, resp. vytvořit tzv. seznam aktivních hran, které aktuální řádku protínají. Uvedená datová reprezentace se používá zejména v různých simulátorech, např. pilotáže letadel. Pro operace plnění, šrafování apod. je nutné uvedenou reprezentaci modifikovat tak, že zřetězený seznam je uspořádaný vzhledem k hodnotám x_0 , a to pro každou řádku i v seznamu aktivních hran. Je nutné vhodně reprezentovat vodorovné hrany a případy, kdy řádka vrcholem n-úhelníka prochází nebo se jej dotýká.

Rastrová řádková konverze

Technika řádkové konverze umožňující dobrou reprezentaci objektů využívá operací v pohyblivé řádové čárce. Toto je určitou nevýhodou, neboť ne pro všechny aplikace je používán procesor podporující pohyblivou řádovou čárku. Proto je možné

použít tzv. rastrové řádkové konverze, viz obr.3.6.5. Její výhoda spočívá v tom, že poskytuje nejen rastrovou reprezentaci zobrazovaného objektu, ale je s ní možné navíc i pracovat jako s reprezentací hranovou. Pro objekt na obr.3.6.5 dostáváme datovou strukturu uvedenou na obr.3.6.6.



Obr.3.6.5

y_i	N	(x_l, y_r)	
11	2	(5,5) - (5,10)	
10	2	(5,5) - (11,17)	
9	2	(4,4) - (17,17)	
8	2	(4,4) - (17,17)	
7	4	(3,3) - (8,8)	(8,9) - (17,17)
6	4	(3,3) - (6,7)	(10,11) - (17,17)
5	4	(2,2) - (4,5)	(12,13) - (17,17)
4	4	(2,2) - (6,7)	(14,15) - (17,17)
3	4	(1,1) - (8,9)	(16,17) - (17,17)
2	2	(1,1) - (10,11)	
1	2	(0,7) - (12,13)	
0	2	(8,14) - (14,15)	

kde **N** je počet průsečíků hran s danou řádkou y_i

Obr.3.6.6

Uvedená datová struktura může být vygenerována upraveným Bresenhamovým algoritmem. Pro každý řádek pak dostáváme dvojici hodnot $\langle x_l, x_r \rangle$ určující začátek a konec aktivovaných bodů rastru pro danou souřadnici y_i. V případě, že zobrazovaný útvar je uzavřený, pak pro každou řádku dostáváme dvě dvojice hodnot $\langle x_{l_i}, x_{r_i} \rangle$ a $\langle x_{l_{i+1}}, x_{r_{i+1}} \rangle$. Hodnoty $\langle x_{l_i}, x_{r_{i+1}} \rangle$ pak určují vnější hranice zobrazovaného útvaru. Datovou strukturu je možné modifikovat i pro případ použití kruhových oblouků, viz kap. 3.7.

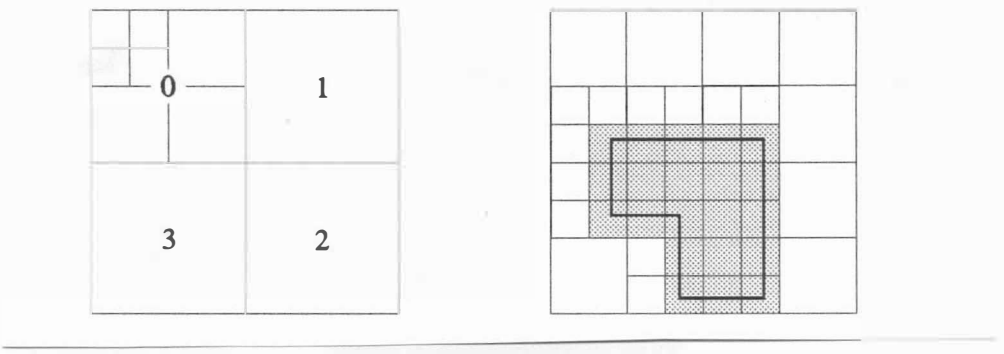
Uvedená datová struktura je vhodná i pro elektrostatické zapisovače, pokud se nedosáhne určité složitosti, od které může být výhodnější použít záznam přímo do paměti reprezentující obraz. Datovou strukturu lze též modifikovat pro případ použití barev či reprezentace ploch v prostoru.

Stromová struktura QUADTREE

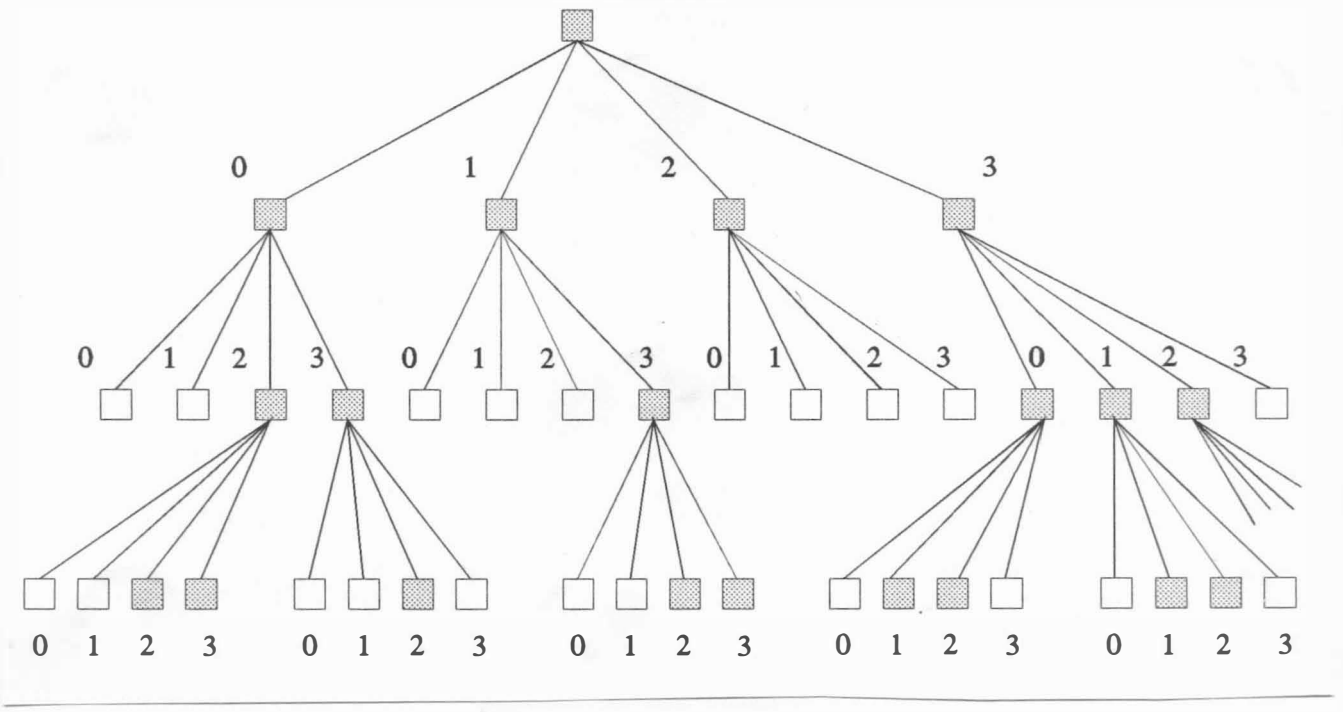
Použití hierarchických či stromových struktur je v počítačové grafice velmi časté. V předchozím případě byla použita hierarchická datová struktura k reprezentaci jednotlivých datových objektů. Nicméně však vývoj elektroniky a prudký pokles ceny pamětí při vzrůstu hustoty integrace polovodičových prvků umožnil použití tzv. **quadtrees a octrees k reprezentaci objektů** na úrovni nikoliv definic, ale na úrovni rastru. Myšlenka je v podstatě velmi jednoduchá a vychází vlastně z Warnockova algoritmu, viz kap. 7.6.

Na obr. 3.6.7.a je naznačen postup dělení oblasti, spolu s odpovídajícím očíslováním. V případě quadtrees se např. daný n-úhelník reprezentuje datovou strukturou tak, že odráží **skutečný tvar** v rastru, viz obr. 3.6.7.b. Datová struktura daného n-úhelníka má pak tvar, který je uveden na obr. 3.6.8.

Výhodou stromové datové struktury quadtrees je, že umožňuje i určení takových veličin, jako je obsah, těžiště apod. Počet úrovní stromu je pak dán velikostí nejmenší zobrazitelné plošky, což bývá většinou pixel.



Obr. 3.6.7



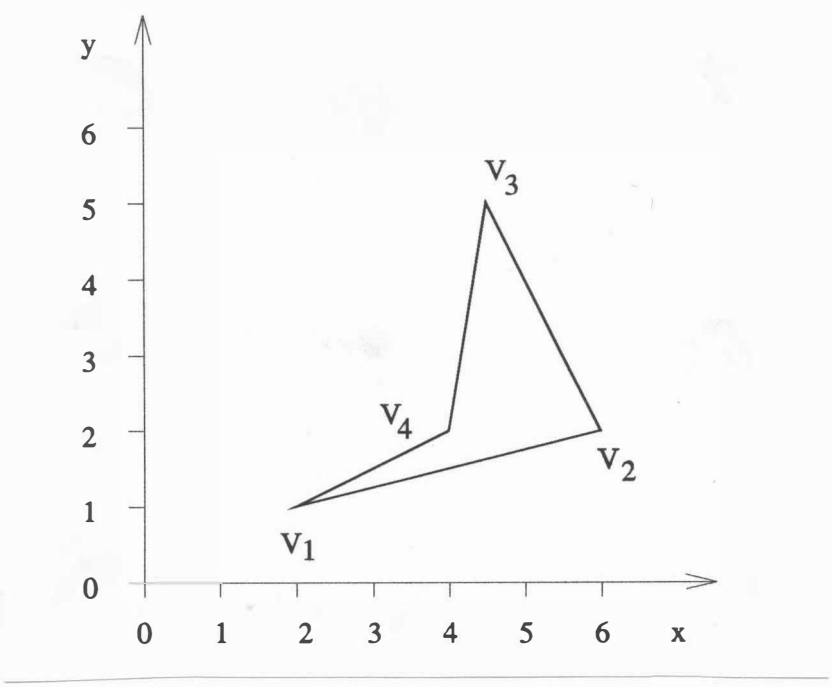
Obr. 3.6.8

3.7 Reprezentace n-úhelníků a oblastí

Až dosud uvedené algoritmy a metody řešily problematiku reprezentace úseček nebo n-úhelníků. V technické praxi však nastávají složitější případy, kdy je nezbytné definovat oblasti, jejichž hranice jsou tvořeny jak úsečkami, tak i oblouky, resp. částmi kuželosečky. Navíc je nezbytné zvládnout i ty situace, kdy oblasti obsahují "díry", jejichž hranice jsou tvořeny analogickým způsobem. Je pochopitelně možné aproximovat oblouky lomenou čárou, avšak to má za následek neúměrné zvětšení velikosti datových struktur i rychlost výpočtů a vykreslování.

Reprezentace n-úhelníků

V mnoha aplikacích je nezbytné použít jednoduchou datovou strukturu pro definici n-úhelníka. Nejjednodušším případem je použití seznamu souřadnic vrcholů, které jsou uspořádány tak, že dva po sobě jdoucí vrcholy definují odpovídající hranu daného n-úhelníka. Vlastní způsob realizace datové struktury se liší případ od případu, např. GKS předpokládá dvě jednorozměrná pole, a to pro souřadnice x a souřadnice y zvlášť, viz obr.3.7.1. Součástí dat pak bývá údaj, který specifikuje počet vrcholů. V některých případech je nutné přidat na konec seznamu souřadnice prvého bodu, aby se zajistilo "uzavření" n-úhelníka.



Obr. 3.7.1

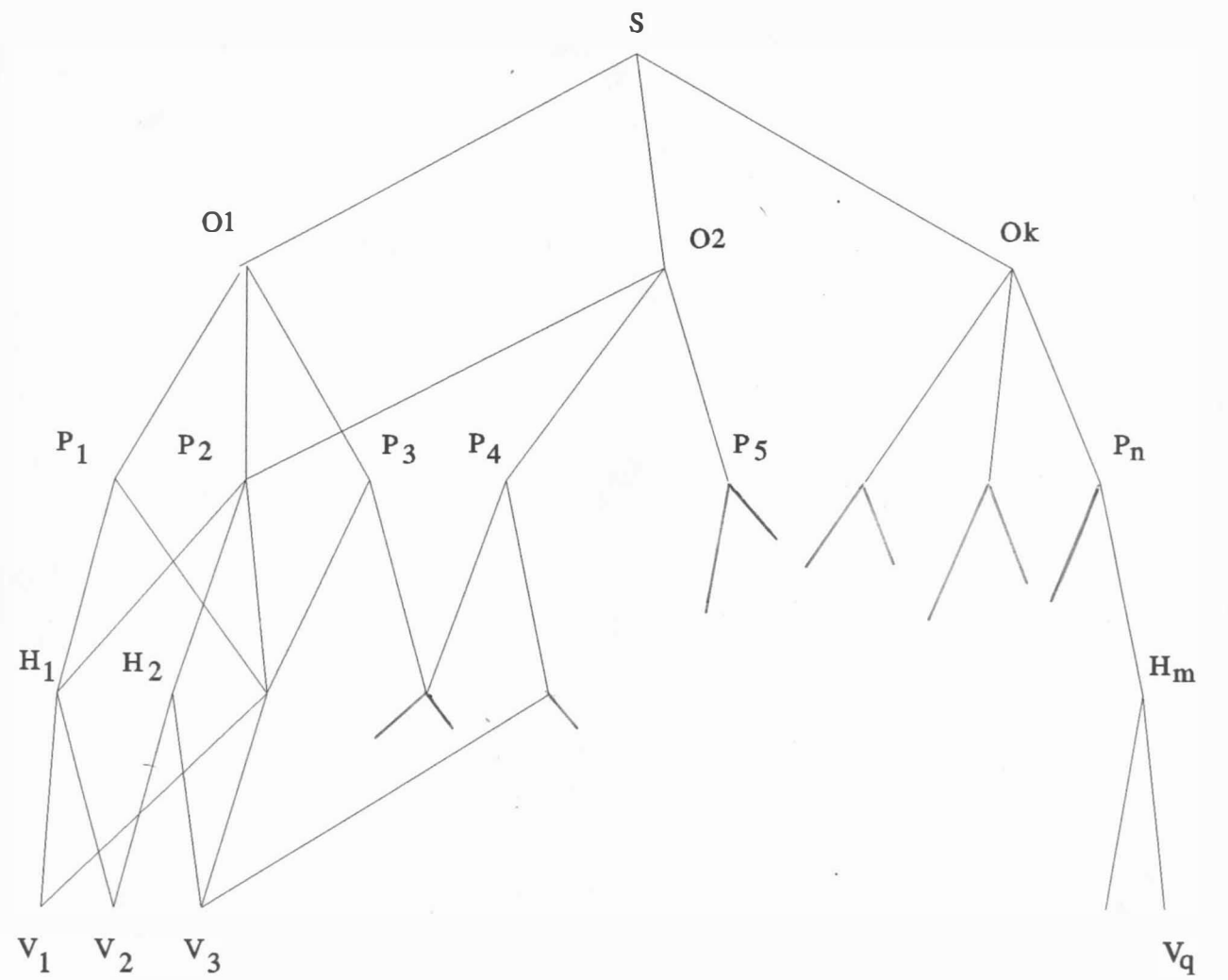
Souřadnice vrcholů Počet vrcholů N = 4

X	2	6	4.5	4
Y	1	2	5	2

Vykreslení daného n-úhelníka je pak možné realizovat např. algoritmem 3.7.1.

```
MOVE TO ( X[N] , Y[N] );
for i:=1 to N do
   begin LINE TO ( X[i] , Y[i] ); end
```

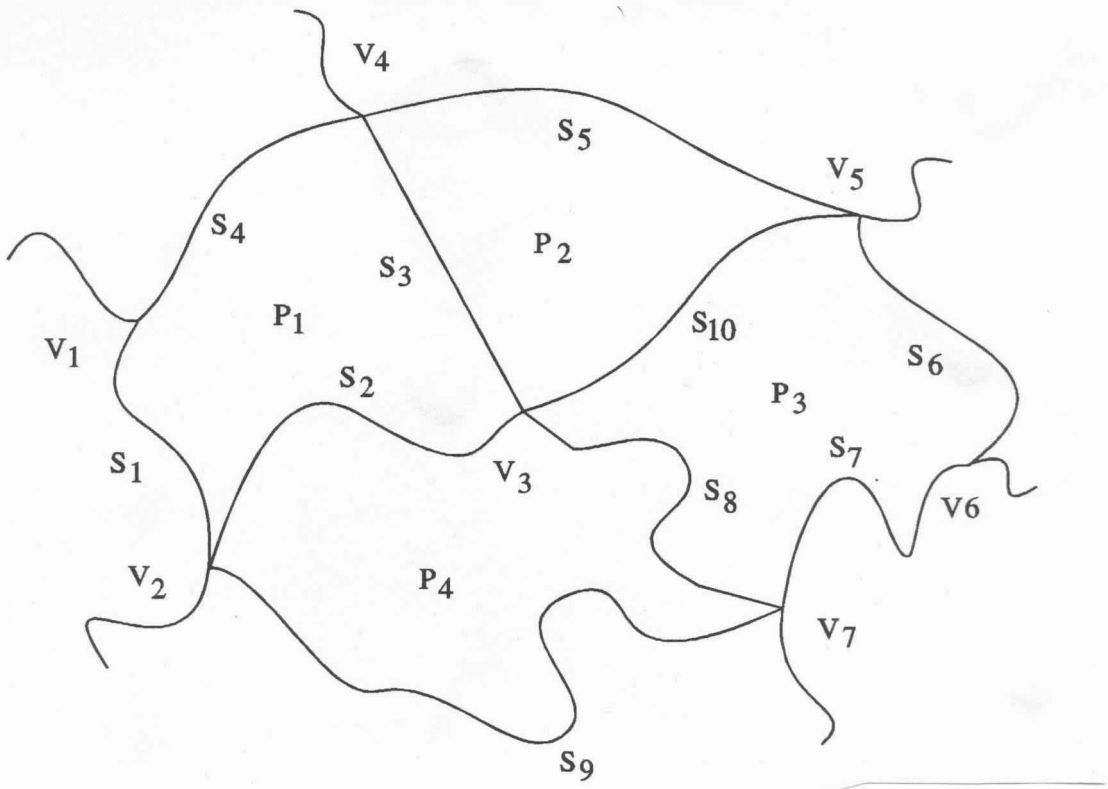
Algoritmus 3.7.1



Obr. 3.7.2

Uvedená datová struktura však není příliš vhodná pro složitější manipulace, neboť neposkytuje informace o tom, jak se daná plocha podílí na definici zobrazovaného objektu. Navíc jsou problémy s reprezentací n-úhelníků, které obsahují díry.

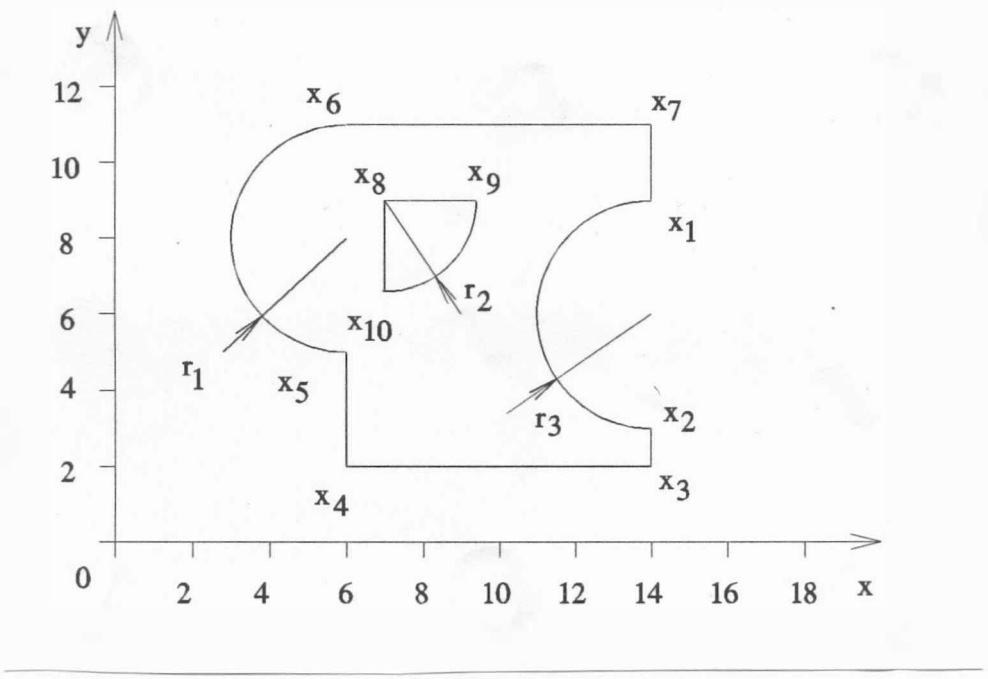
Zobrazovanou scénu je obecně možné popsat pomocí hierarchické struktury, viz obr.3.7.2, pokud jsou objekty tvořeny n-úhelníky, které jsou definovány souřadnicemi vrcholů. Uvedenou datovou strukturu lze modifikovat nebo realizovat mnoha způsoby, např. lze vynechat definici hran (plocha je pak definována vrcholy), nebo realizaci provést pomocí statických polí s omezením max. počtu podřízených uzlů či pomocí dynamických seznamových struktur. Nicméně existují aplikace, pro něž uvedená datová struktura není postačující, např. v kartografii. Hranice kartografických oblastí nejsou přímkové, např. hranice států, ale jde obecně o lomenou čáru, viz obr.3.7.3.



Obr.3.7.3

V tomto případě je např. plocha P_1 definována nejen vrcholy V_1, V_2, V_3, V_4 , ale i vrcholy lomených čar S_1, S_2, S_3, S_4. Navíc může být požadováno i další hierarchické dělení, neboť např. kontinent je rozdělen na republiky, království apod. Ty pak jsou rozděleny na státy či hrabství, která mohou být dále členěna. Z uvedeného je zřejmé, že **návrh datových struktur podstatně ovlivní realizaci algoritmů**.

Uvažme oblast, jejíž hranice je tvořena jak úsečkami, tak i oblouky a která obsahuje též "díry", viz obr. 3.7.4.



Obr. 3.7.4

Pokud nemají být kruhové oblouky aproximovány lineárními úseky, je nezbytné odvodit jednotlivé geometrické transformace i pro kuželosečky, viz kap. 4. Navíc je též nutné kromě koncových bodů oblouku definovat orientaci oblouku z bodu $\mathbf{x}_1$ do bodu $\mathbf{x}_2$. Je vhodné podotknout, že při změně měřítka obecně dochází ke změně kružnice v elipsu.

Kuželosečky je možné reprezentovat pomocí kvadratické formy při použití homogenních souřadnic:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 0$$

Lze pak pro ně definovat jednotlivé geometrické transformace, viz kap.4. Datová struktura reprezentující oblast bez děr musí obsahovat nejen koncové body, ale též i orientaci a koeficienty reprezentující kuželosečku.

Kuželosečka je dána rovnicí

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

a tedy při použití kvadratické formy k zápisu dostáváme:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = [\, x \,,\, y \,,\, 1 \,] \circ \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \circ \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Pro rovnici přímky

$$a\,x + b\,y + c = 0$$

jako speciální případ kuželosečky pak dostáváme:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = [\, x \,,\, y \,,\, 1 \,] \circ \begin{bmatrix} 0 & 0 & a/2 \\ 0 & 0 & b/2 \\ a/2 & b/2 & c \end{bmatrix} \circ \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Je vhodné poznamenat, že matice **A** je symetrická, a stačí tedy obecně 6 koeficientů k reprezentaci jak přímky, tak i libovolné kuželosečky.

Nicméně je většinou zapotřebí úsečka, kruhový oblouk, resp. eliptický oblouk apod. To znamená, že je nutné uchovávat též i informace o koncových bodech úsečky, resp. oblouku. V případě oblouku též informaci určující orientaci.

3.8 Plnění a šrafování

V mnoha aplikacích se používá operací plnění a šrafování. I když je možné realizovat šrafování, resp. plnění n-úhelníka aplikováním operace ořezávání, viz kap.6., je tento způsob neefektivní zejména pro rastrová zařízení s přímým přístupem. Pro tato zařízení je možné realizovat algoritmy založené na jiných principech, které jsou vhodnější z hlediska přímé realizace pomocí hardwaru.

Semínkové plnění

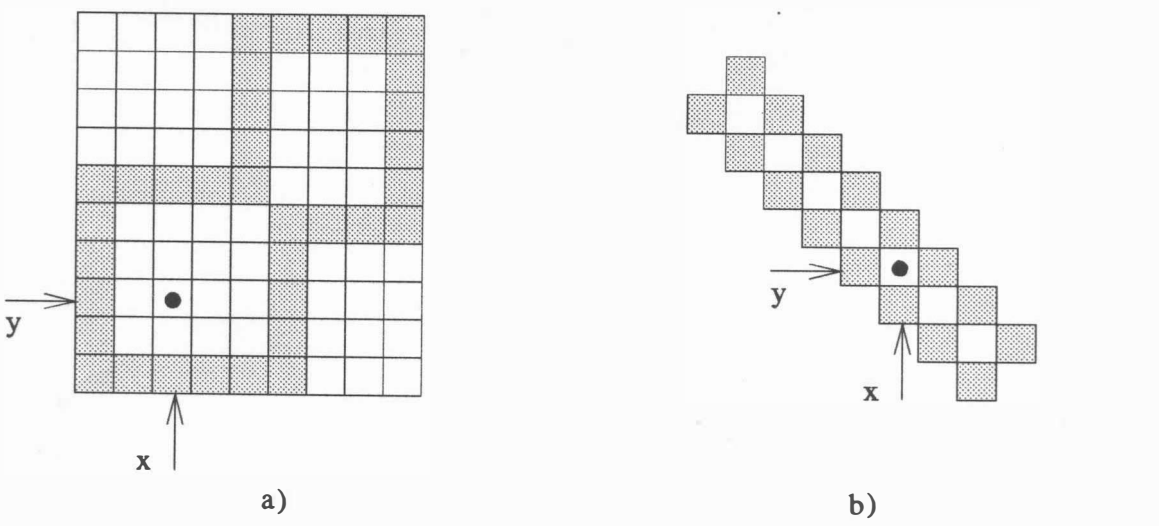
Algoritmus semínkového plnění (seed fill) je pravděpodobně jedním z nejjednodušších algoritmů, který je založený na aktivaci čtyř sousedních bodů, které pak slouží jako startovací body v dalším kroku. Semínkové plnění lze popsat algoritmem 3.8.1.

```
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 };
```

Algoritmus 3.8.1

Uvedený algoritmus je založený na rekurzi a je velmi neefektivní, neboť poskytuje pro další úroveň rekurze i ty body, které byly již v daném nebo předchozím kroku inicializovány. Tyto nevýhody se řeší nerekurzivní modifikací, která je vhodná zejména pro implementaci ve VLSI obvodech. Jednou z možných alternativ je např. postupné plnění svisle či vodorovně od

semínka, přičemž se využívá zásobník k ukládání tzv. pokračovacích bodů. Uvedený algoritmus však selhává v některých případech, viz obr.3.8.1. V případě, který je uveden na obr.3.8.1.a, se vyplní jen spodní část n-úhelníka, je-li zadán startovací bod (x,y), neboť semínko se nemůže "přenést" do horní části n-úhelníka. Analogický případ charakteristický pro úzké obdélníky je uveden na obr.3.8.1.b.



Obr.3.8.1

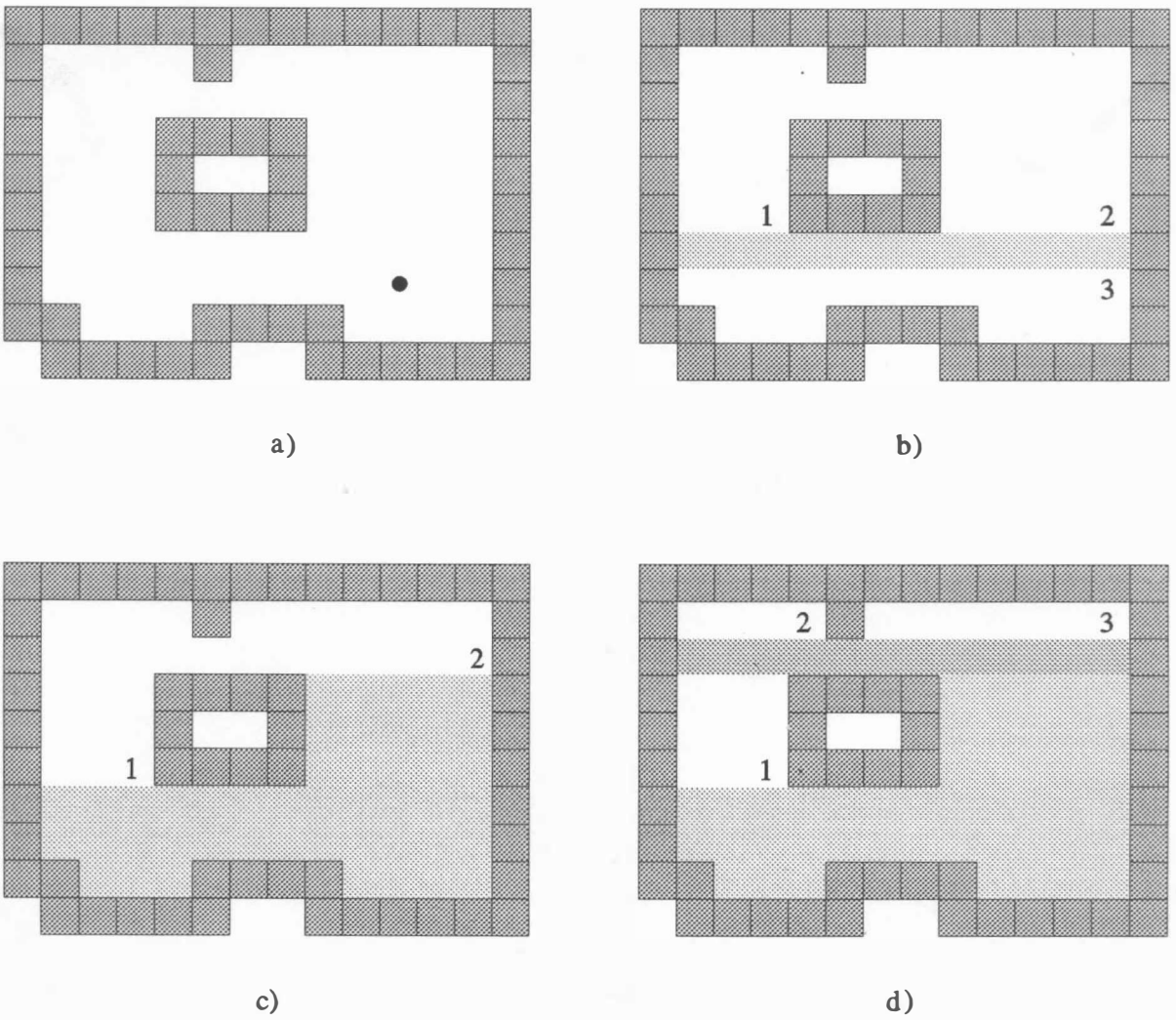
Nerekurzivní verzi algoritmu semínkového plnění lze popsat algoritmem 3.8.2.

```
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 3.8.2

Pro demonstraci funkce algoritmu uvažme jednoduchou oblast, která je na obr.3.8.2.a. Po vyplnění řádku jsou do zásobníku uloženy souřadnice bodů, viz obr.3.8.2.b, které jsou označeny čísly vyjadřujícími pořadí ukládání souřadnic bodů do zásobníku. Tyto body jsou pak postupně vzaty jako startovací.



a) b)

c) d)

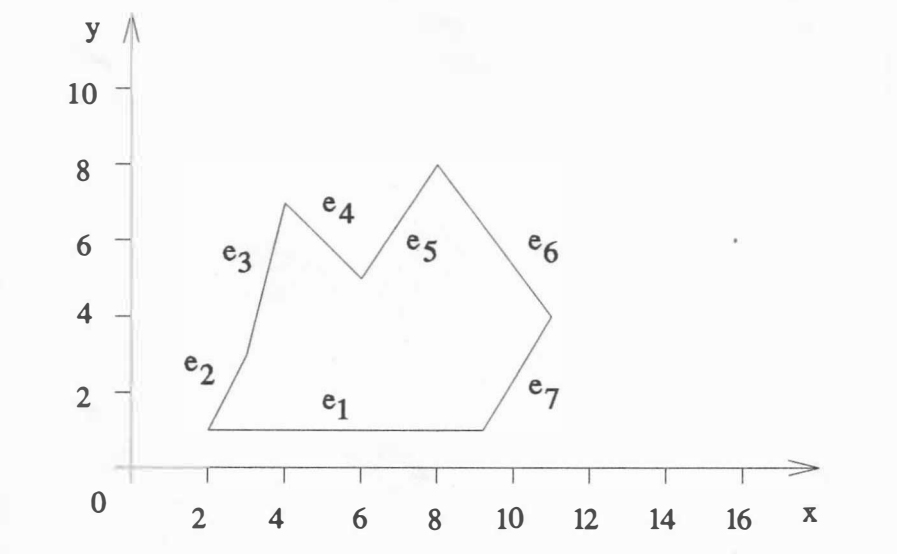
Obr.3.8.2

Kromě uvedených algoritmů existuje celá řada jejich modifikací či algoritmů, které jsou založeny na jiných principech, viz např. [2],[34],[66], využívajících vlastností frame-bufferu. Je zřejmé, že uvedený algoritmus lze modifikovat i pro plnění vzorem, přičemž je pouze nutné pozměnit proceduru Set Pixel Color. Pro zařízení se sekvenčním přístupem k bodu není metoda semínka vhodná a většinou se používá metoda následující.

Plnění řádkovou konverzí

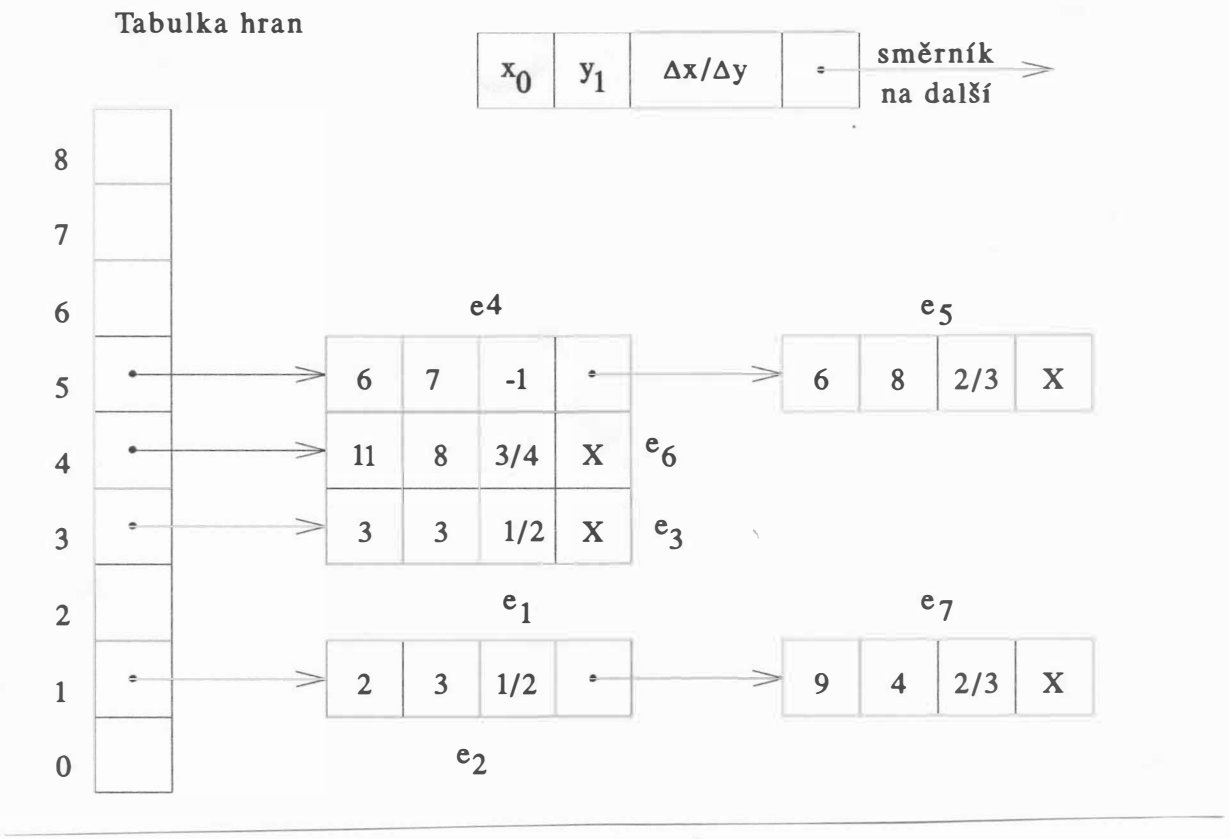
Metoda plnění n-úhelníků řádkovou konverzí (scan line filling) je vhodná pro zařízení se sekvenčním přístupem k bodu, neboť požadavky na výstup z hlediska objemu dat mohou být minimalizovány. Navíc u zařízení umožňujících měnit jas, resp. barvu, umožňuje tato metoda měnit postupně jas, resp. barvu, v rámci kresleného řádku. Metoda je založena na řádkové konverzi, přičemž pro účely plnění je možné datovou strukturu a ukládaná data modifikovat. Pro jednoduchost budeme předpokládat, že plnění se bude provádět ve směru vodorovném. Algoritmus plnění řádkovou konverzí je založen na principu, kdy přímka může mít pouze sudý počet průsečíků s n-úhelníkem. Je nutné řešit speciální případy, kdy hrana leží na přímce, či přímka vrcholem prochází, nebo se jej dotýká. Většina implementací tyto speciální případy v zásadě řeší takto (viz obr.3.8.3):

- leží-li hrana na přímce reprezentující zpracovávaný řádek, pak se tato hrana **vynechá**, tj. nepoužije se v dalším zpracování
- dotýká-li se přímka zpracovávaného řádku vrcholu, pak se obě hrany zpracují normálním způsobem
- prochází-li přímka zpracovávaného řádku vrcholem, pak se buď zkrátí jedna hrana tak, aby začínala až na dalším řádku, anebo je nutné s ní takto zacházet.



Obr.3.8.3

Pro reprezentaci prvků seznamu v datové struktuře řádkové konverze použijeme modifikovanou datovou strukturu, která obsahuje místo Δx a Δy přímo hodnotu $\Delta x/\Delta y$. Pak n-úhelník z obr. 3.8.3 lze reprezentovat pomocí datové struktury, viz obr. 3.8.4.



Obr. 3.8.4

Plnění založené na řádkové konverzi lze popsat algoritmem 3.8.3.

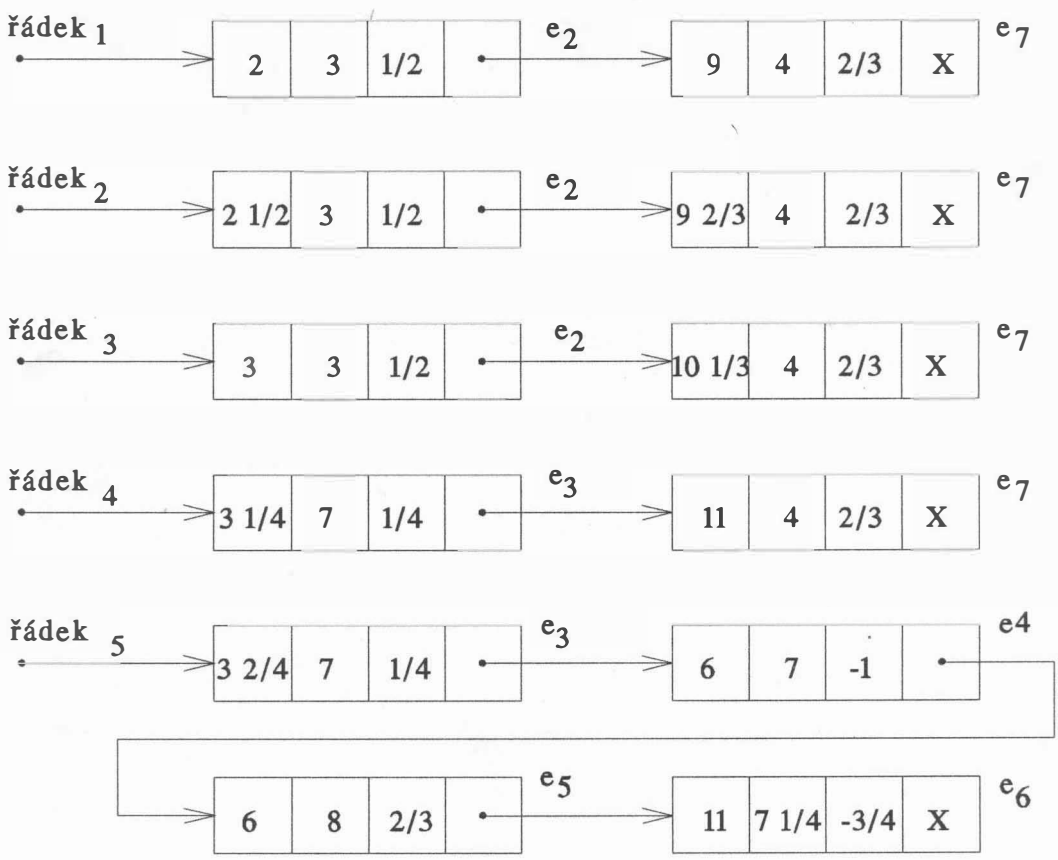
1. Vytvoř prázdný seznam aktivních hran AE tab , který bude charakterizovat ty hrany, které jsou protínány právě zpracovávanou řádkou.
2. Nastav index zpracovávané řádky na nejnižší hodnotu, tj. y = 0
3. Pokud $y \leq y_{max}$, dělej:

a) zatřiď prvky ze seznamu E tab[y] do seznamu aktivních hran AE tab a setřiď je podle hodnot x

b) plň n-úhelník v rámci zpracovávané řádky tak, že se aktivují ty pixely, jejichž souřadnice x je určena sousedními prvky v seznamu v AE tab
c) odstraň z AE tab seznamu ty prvky, jejichž horní souřadnice y je rovna zpracovávanému řádku
d) pro každý zbylý prvek v seznamu aktivních hran AE tab nahraď hodnotu průsečíku x pro následující řádku hodnotou x + 1 / m, tj. připočti hodnotu inverzní směrnice
e) zvyš index zpracovávané řádky o 1, tj y := y + 1

Algoritmus 3.8.3

Na obr. 3.8.5 je uveden obsah seznamu aktivních hran AE tab pro různé řádky v okamžiku jejich zpracování.



Obr. 3.8.5

Uvedeným algoritmem je možné též plnit n-úhelník definovaným vzorem. V tomto případě je nutné aktivovat odpovídající pixely ve shodě s definicí vzoru. Je-li vzor definován tabulkou o rozměru n x m, jejíž prvek definuje jas či barvu, se kterými se daný pixel má aktivovat, pak lze pixel aktivovat příkazem:

```
Set Pixel Color ( x , y , TAB [ x mod n + 1 , y mod m + 1 ] )
```

Vhodně zvoleným vzorem lze pak docílit i vyšrafování daného objektu. Uvedený postup lze použít i k odstraňování neviditelných hran či povrchů, viz kap.7.

Plnění a šrafování rastrovou řádkovou konverzí

V kap.3.6 byl uveden princip rastrové řádkové konverze. Vzhledem k tomu, že pro každou řádku je v datové struktuře obsažena informace o tom, který pixel dané oblasti přináleží, je zřejmé, že algoritmy plnění a šrafování jsou velmi jednoduché, a tedy i rychlé. Algoritmus plnění může být popsán např. alg.3.8.4.

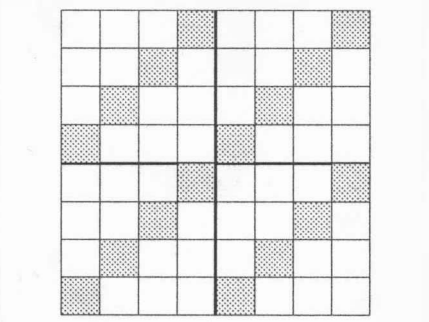
```
procedure FILL AREA;    { plnění }
var i,j,k: integer;
begin
   for y := y_min to y_max do
   begin j:=1;
         while j < počet průsečíků pro dané y do
         begin for k := x_r[y,j]+1 to x_l[y,j+1]-1 do
                     Set Pixel(k,y);
               j:=j+2;
         end
   end
end
```

Algoritmus 3.8.4

```
procedure HATCH AREA;
{ šrafování }
const N = 4;
type pattern = array [1..N,1..N] of byte;
const pat: pattern = ((1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1));
{ odpovídající vzor viz obr. 3.8.6 }
var i,j,k: integer;
begin
   for y := y_min to y_max do
   begin j := 1;
         while j < počet průsečíků pro dané y do
         begin for k := x_r[y,j]+1 to x_l[y,j+1]-1 do
                   if pat[y mod n + 1,k mod N + 1] = 1
                      then Set Pixel(k,y);
               j:=j+2;
         end
   end
end
```

Algoritmus 3.8.5

Šrafování je operace velmi podobná plnění s tím rozdílem, že se plní vzorem, který je v našem případě uložen v poli PAT. Algoritmus 3.8.5. pak ukazuje postup při šrafování.



Obr. 3.8.6

Uvedený vzor na obr. 3.8.6 vyplní daný n-úhelník tak, že šrafy budou mít sklon 45°. Obecně je možné nadefinovat téměř libovolný vzor vhodnou volbou N a nastavením pole PAT.

Vzhledem k jednoduchosti mohou být uvedené algoritmy realizovány hardwarově.

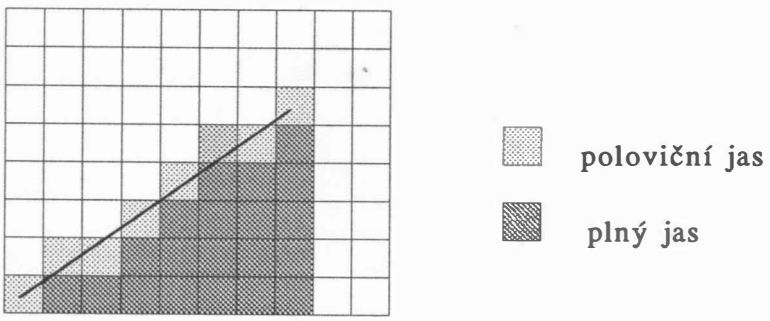
3.9 Antialiasing

Při zobrazování přímek, úseček či jiných elementů v rastrovém prostředí se projevuje vzhledem k omezené rozlišovací schopnosti problém vzorkování, neboť prostor, jas či barva musí být vyjádřeny pomocí diskrétních veličin. Problém vzorkování se pak projevuje v zásadě tím, že:

- úsečky se zobrazují s typickým schodovitým průběhem
- malé objekty, které jsou menší než pixel, se buď zobrazují podstatně větší (vlivem jasu, se kterým se daný pixel aktivuje), nebo se nezobrazují vůbec
- dochází ke ztrátě informace u detailů, které jsou rozměrově blízké rozměru pixelu
- při animaci pohybu "objevováním" a "mizením" malých objektů

Techniky, které se snaží řešit tuto problematiku, se nazývají **antialiasing**. Lze je nalézt např. v [39]-[41], [44], [64], [109]. Problém vzorkování lze řešit několika přístupy, přičemž všechny v zásadě používají modulaci jasu pixelu k vytvoření dojmu "hladkosti".

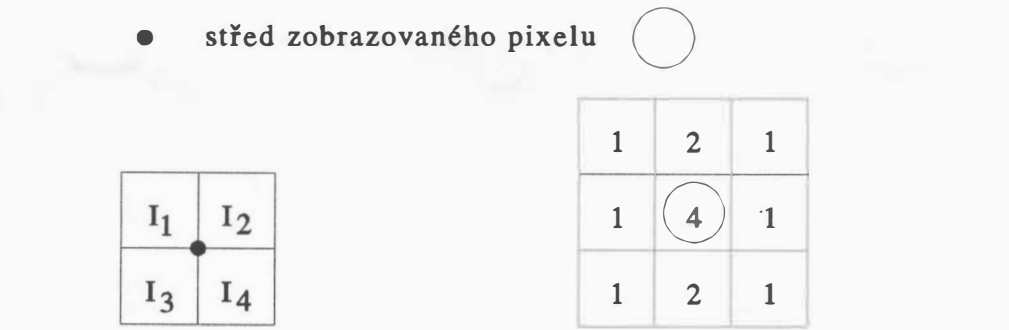
Jednou ze základních technik je rozšířený Bresenhamův algoritmus pro kreslení úsečky, viz algoritmus 3.9.1. Lze ukázat, např. viz [103], že hodnota proměnné d, přičemž $d \in \langle 0 , 1 \rangle$, je úměrná velikosti obsazené plochy pixelu, viz obr.3.9.1.



Obr.3.9.1

```
procedure DRAW ANTIALIAS ( x1,y1, { počáteční bod }
                           x2,y2: integer { koncový bod } );
var x,y,d,m: real;
begin
  x:=x1; y:=y1;
  m:=(y2-y1)/(x2-x1);
  a:=1-m; d:=0.5;
  Set Pixel ( x , y , d );
  repeat
    if d < a
      then { horizontální krok }
        d := d + m
      else { diagonální krok }
        begin d := d - a; y := y + 1 end;
    x := x + 1;
    Set Pixel ( x , y , d )
  until x = x2 ;
```

Algoritmus 3.9.1



Obr. 3.9.2

Druhou možností realizace antialiasingu je vzorkování s vyšší frekvencí, tj. vytvoření obrazu s větším rozlišením, než je možné zobrazit. Pixel se zobrazuje s intenzitou danou aritmetickým nebo váhovým průměrem z vypočtených "okolních" obrazových elementů, viz obr. 3.9.2.

Tato technika se vzrůstajícím redukčním faktorem vyžaduje i zvětšení velikosti potřebné paměti. Z experimentů vyplývá, že výsledky získané při rozlišovací schopnosti 512 x 512 bodů při 16 úrovních jasu jsou lepší než při rozlišení 1024 x 1024 při dvou úrovních jasu. To znamená, že v jistých mezích je lépe "věnovat" kapacitu paměti na reprezentaci více úrovní jasu či barvy místo zvýšení rozlišovací schopnosti při stejných paměťových nárocích.

Další možnou technikou je filtrování. Vychází se z toho, že intenzita pixelu je ovlivňována svým okolím. Obecně lze říci, že platí:

$$c(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} H(\xi,\eta).I(x+\xi,y+\eta)\, d\xi\, d\eta$$

kde $H(\xi,\eta)$ je filtrační funkce, $I(x,y)$ je intenzita získaná v bodě (x,y) a $c(\xi,\eta)$ je intenzita aktivovaného filtru po provedení filtrace. Pro diskrétní hodnoty lze psát:

$$c(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} H(i,j).I(x+i,y+j)$$

přičemž filtrační funkce je dána maticí **H**, např.

$$H(i,j) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{pro } i \in \{-1,0,1\} \;\&\; j \in \{-1,0,1\}$$
$$\text{jinak } H(i,j) = 0 \text{ pro ostatní } i,j$$

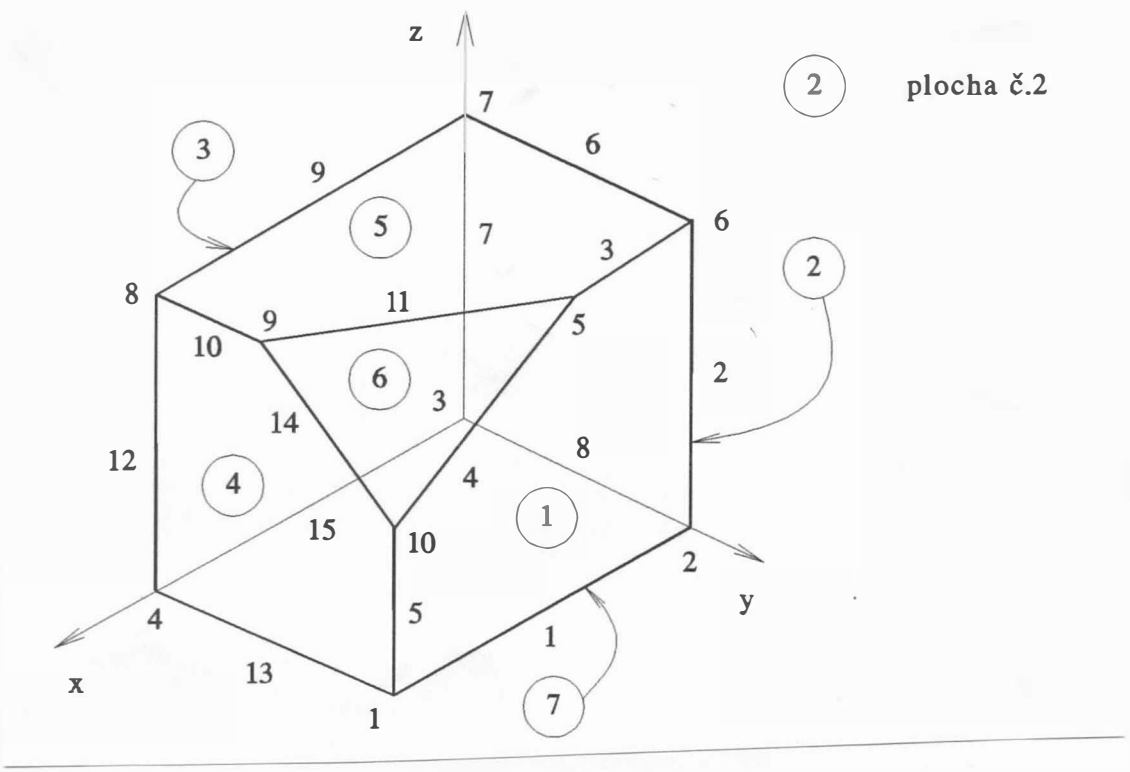
Techniky založené na filtrování již vytvořeného obrazu se používají také při zpracování obrazu, viz [52]. Tyto techniky nabývají na důležitosti zejména s použitím technik typu "ray tracing", viz [41], [64].

3.10 Reprezentace třírozměrných objektů

Až dosud jsme se zabývali možností reprezentace dvourozměrných objektů, tj. objektů, které jsou rovinné. S malými modifikacemi jsou některé datové struktury použitelné i pro reprezentaci **rovinných** objektů v prostoru, např. n-úhelník může ležet obecně na libovolné rovině v prostoru. Z hlediska způsobu reprezentace lze jednotlivé způsoby rozdělit do skupin hraničních reprezentací, nebo objemových reprezentací.

Hraniční reprezentace

Hraniční reprezentace lze dělit na reprezentace pomocí hran, ploch, nebo tzv. okřídlených hran.



Obr.3.10.1

Hranová reprezentace

Tato reprezentace je vhodná pouze pro jednoduché aplikace. Uvážíme-li jednoduché těleso z obr.3.10.1, pak toto těleso lze hranově reprezentovat pomocí tabulek:

- tabulky souřadnic vrcholů **V**, tj. souřadnic x,y,z,
- tabulky hran **H**, která určuje pro příslušnou hranu, jaké vrcholy jsou koncové.

Tabulka **V**, obsahující souřadnice jednotlivých vrcholů tělesa z obr.3.10.1, má pak tvar:

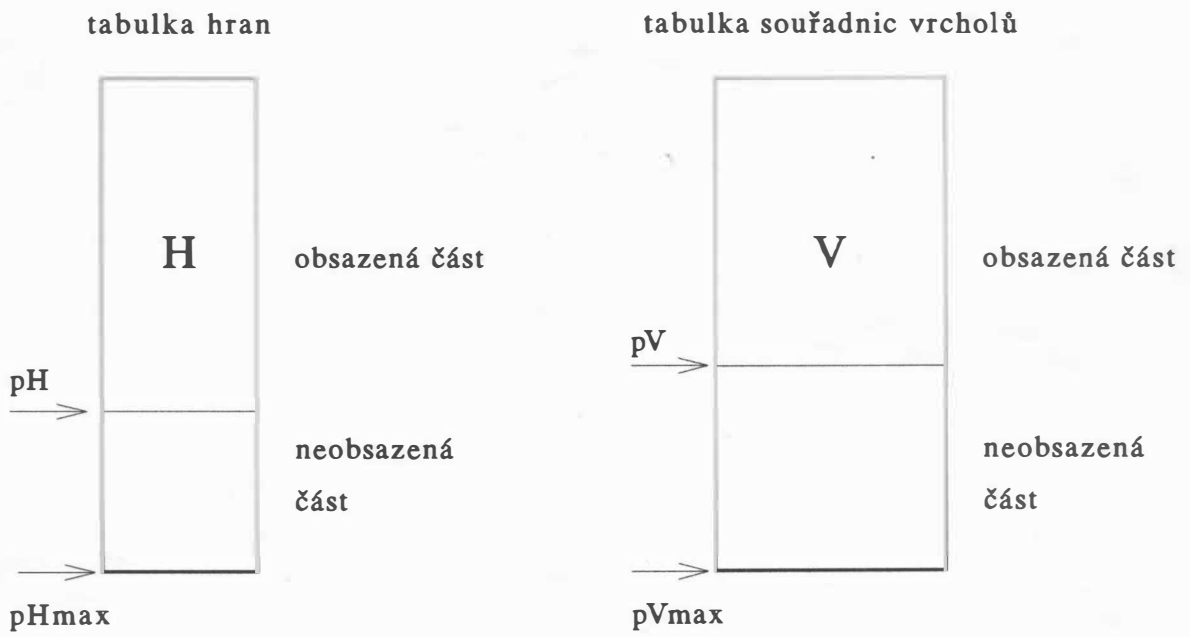
vrchol 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.

$$\mathbf{V} = \begin{bmatrix} 4 & 0 & 0 & 4 & 1 & 0 & 0 & 4 & 4 & 4 \\ 3 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 & 4 & 2 \end{bmatrix}$$

Tabulka **H**, definující hrany, obsahuje čísla vrcholů, které jsou koncovými body příslušné hrany. Pro těleso z obr.3.10.1 má tvar:

hrana 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.

$$\mathbf{H} = \begin{bmatrix} 1 & 2 & 6 & 5 & 10 & 6 & 7 & 3 & 7 & 8 & 9 & 8 & 4 & 9 & 3 \\ 2 & 6 & 5 & 10 & 1 & 7 & 3 & 2 & 8 & 9 & 5 & 4 & 1 & 10 & 4 \end{bmatrix}$$



Obr.3.10.2

Je zřejmé, že uvedená datová struktura není vhodná, pokud chceme např. odstranit neviditelné hrany. Z popisu obecně nelze jednoznačně určit, které hrany ohraničují danou plochu, viz např. obr.3.7.1. Proto je někdy také tento model nazýván **drátěným modelem**. Nicméně uvedená datová struktura je postačující pro mnoho technických aplikací. Pokud by scénu

tvořilo více těles, nepoužívá se v tomto případě žádná strukturovaná organizace dat. Tabulky **V** a **H** pak mohou být fyzicky realizovány buď jako pole (matice) pro jednodušší případy, nebo pomocí seznamových struktur. Při běžných potřebách pravděpodobně použijeme realizaci datových struktur, která je na obr.3.10.2.

Pro složitější aplikace, kdy dochází k rušení a vkládání hran a jiným grafickým editacím, je **nutné zvážit** možnost použití seznamových struktur, které jsou pružnější, avšak implementačně náročnější, neboť některé prostředky mají různá omezení, např. co do počtu prvků seznamů, neposkytují možnost **Garbage Collection**, tj. znovu využití uvolněné paměti. Tento způsob realizace je pak výhodný např. v kartografických aplikacích, viz kap.3.6.

Hranová reprezentace poskytuje jen základní informace o zobrazovaném tělese. Určení, ke kterým hranám přísluší daný bod, je vlastně problémem sekvenčního prohledávání tabulky **V**, což je při rozsáhlejších úlohách neúnosné.

Reprezentace pomocí ploch

Při řešení problému viditelnosti, tj. eliminace neviditelných ploch, je nezbytné mít k dispozici informace o jednotlivých plochách, které dané těleso tvoří. Je nutné tedy hranový model o tuto informaci rozšířit. Doplněním tabulky ploch **P** a **V** k předcházejícím tabulkám **H** a **V** dostáváme již datovou strukturu, která obsahuje informace o plochách a jejich vazbě na hrany a vrcholy. Pro těleso z obr.3.10.1 má tabulka **P** tvar

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 P

Tabulka 3.10.1

Někdy se ještě tabulka H rozšiřuje o informaci, které dva n-úhelníky obsahují danou hranu. Tabulka H' má pak pro těleso z obr.3.10.1 tvar:

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	9
	2	6	5	10	1	7	3	2	8	9	5	4	1	10	5
	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

Prvé dva řádky tabulky H' obsahují indexy vrcholů hran, druhé dva řádky obsahují indexy ploch, jimiž je hrana definována.

Je obvyklé, že seznam hran v tabulce **P** je uspořádán obvykle v jednom směru, a pak je možné i tabulku hran H vynechat a do modifikované tabulky **P'** uvést přímo vrcholy dané plochy. Pak pro těleso z obr.3.10.1 má tabulka **P'** tvar

číslo plochy	počet vrcholů	čísla vrcholů 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

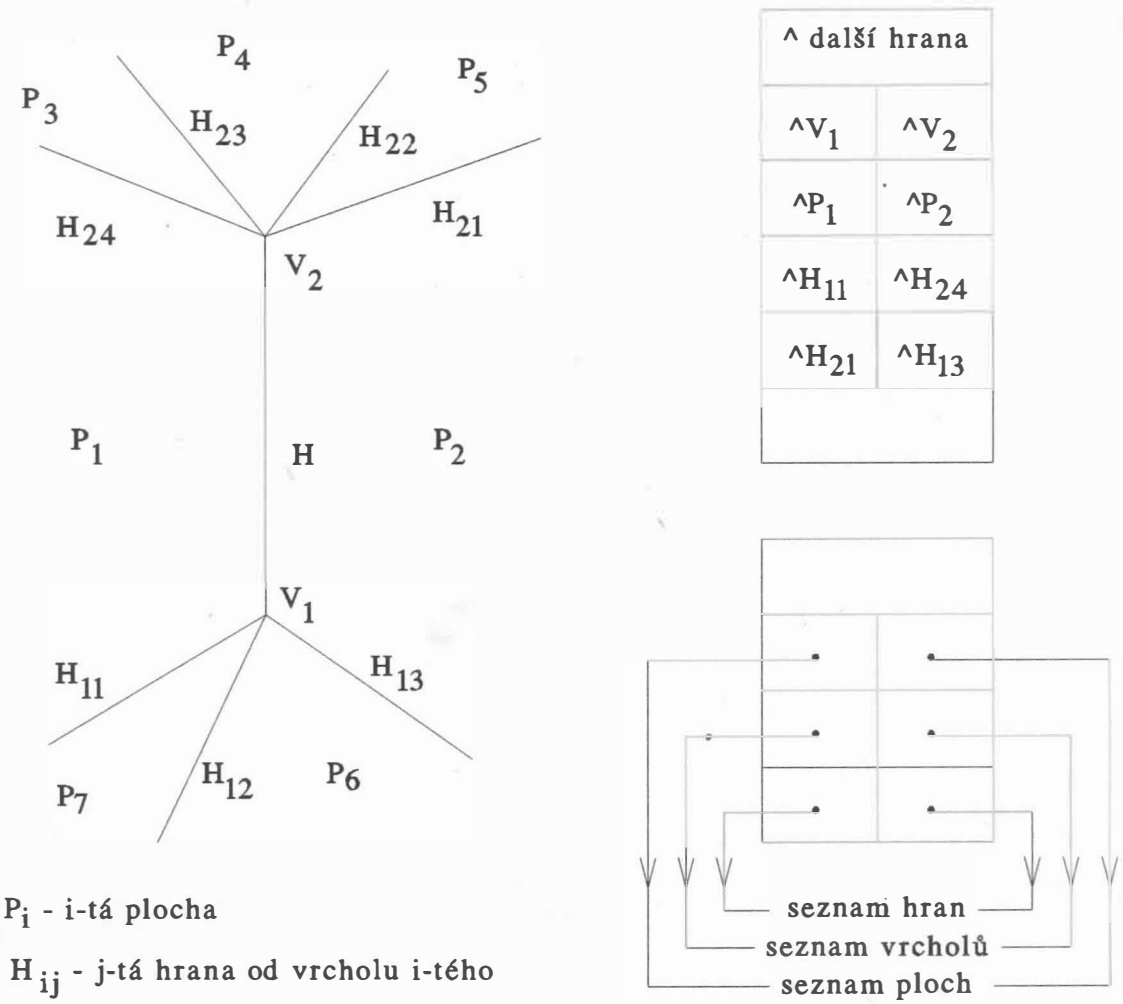
tabulka P'

Tabulka 3.10.2

Pro vnitřní reprezentaci je však vhodné i z tabulky **P'** odvodit tabulky **P** a **H** s tím, že u odkazu na hranu v tabulce **P** je uložena i informace o orientaci hrany. Tato informace se s výhodou využije např. k urychlení procesu eliminace neviditelných hran. Uvedené datové struktury lze opět realizovat pomocí polí nebo seznamů.

Reprezentace "okřídlenými hranami"

Dříve uvedené datové struktury neposkytují možnost uložení dalších potřebných informací. Pro mnoho aplikací je vhodné uchovávat jisté informace o sousedních plochách, hranách či vrcholech. Z tohoto důvodu byla navržena datová struktura známá jako **winged edge** (okřídlená hrana), viz [168], [44], která je znázorněna na obr.3.10.3.



Obr.3.10.3

Její výhodou je, že umožňuje uložení informací o hranách, které tvoří určitou plochu, ale též informace o sousedních plochách či hranách. Navíc lze uchovat i "orientaci" ploch, což může urychlit řešení různých úloh. Při fyzické realizaci se používá téměř výhradně seznamových struktur. V tomto případě je nutné doplnit ještě dané struktury o následující odkazy:

- odkaz na další hranu v seznamu všech hran,
- odkaz na další plochu v seznamu všech ploch,
- odkaz na další vrchol v seznamu všech vrcholů

Uvedené seznamy pak bývá vhodné realizovat jako obousměrné.

Pro hranu h je tedy obecně možné ukládat tyto informace:

- odkazy na vrcholy v_1 a v_2
- odkazy na plochy p_1 a p_2
- odkazy na hrany h_{11} a h_{24}, které jsou částí hranice plochy p_1 a incidují s vrcholem v_1 nebo v_2
- odkazy na hrany h_{21} a h_{13}, které jsou částí hranice plochy p_2 a incidují s vrcholem v_1 nebo v_2

a případně o další dodatečné informace, zdali

- hrana je viditelná, či neviditelná
- hrana je pomocná

Tyto informace se používají např. při aproximaci válce vícebokým hranolem nebo v případě ploch s "děrami". Informace vztahující se k plochám $p_1,\ldots,p_7$ se při zpracovávání hrany h **neuvažují**.

Informace o ploše jsou uloženy v datové struktuře, která kromě jiného obsahuje:

- ukazatel na libovolnou hranu, která tvoří její hranici
- koeficienty A, B, C, D rovnice roviny, na níž plocha leží, tj.

$$Ax + By + Cz + D = 0$$

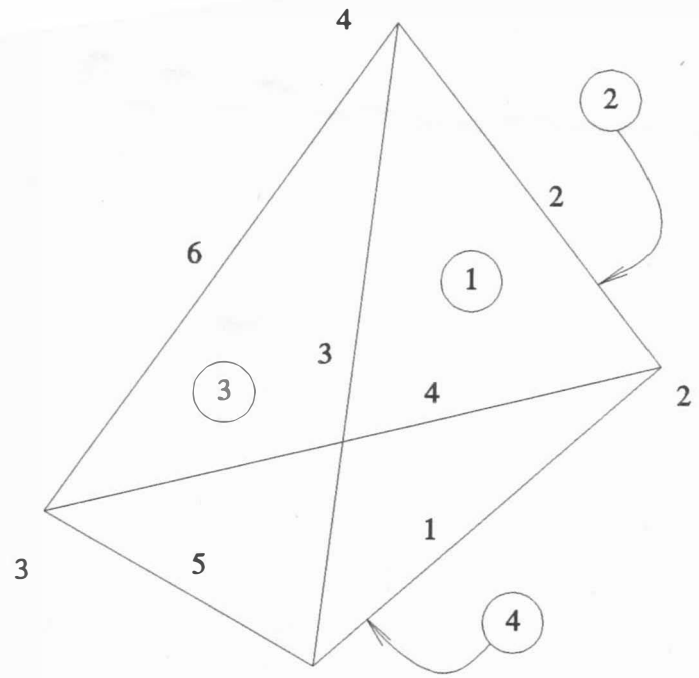
Reprezentace samotného tělesa je pak kromě jiných údajů tvořena třemi cyklickými seznamy, a to:

- seznamem ploch
- seznamem hran
- seznamem vrcholů

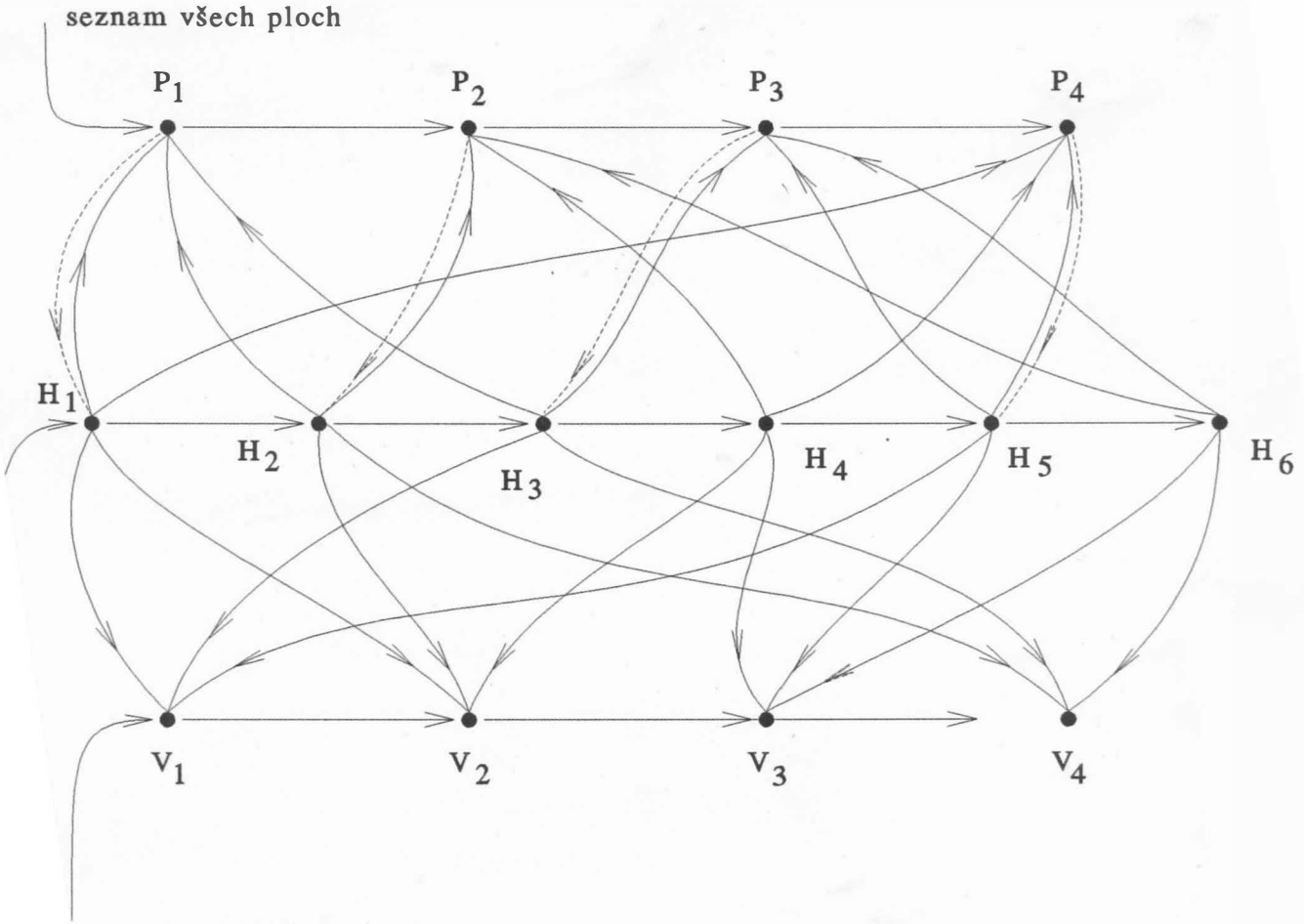
viz obr. 3.10.3.b.

Původní implementace byla provedena v jazyce symbolických adres na počítači PDP-10 a obsahovala 16 podprogramů pro zakládání a manipulaci s uzly, 9 podprogramů pro vlastní práci nad daty, přičemž datové struktury obsahovaly 22 různých typů odkazů, viz [168], [169].

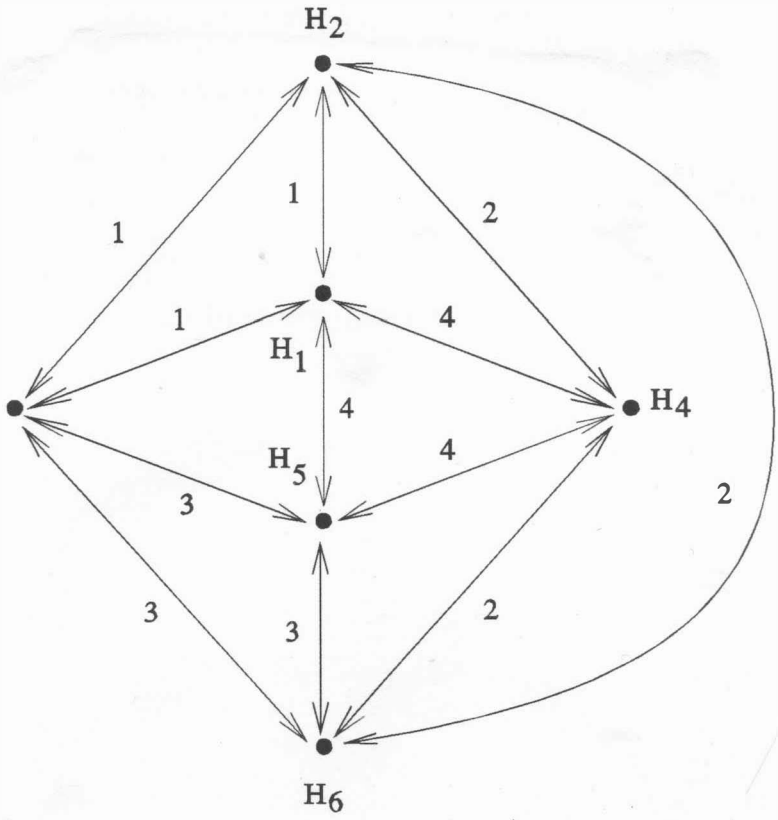
Pro případ tělesa z obr.3.10.4 má pak datová struktura tvar uvedený na obr.3.10.5 a 3.10.6.



Obr. 3.10.4



Obr. 3.10.5

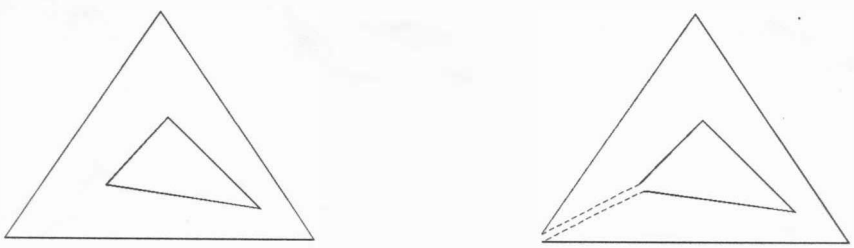


hrana H_1 je společná plochám 1 a 4

a inciduje s hranami H_2, H_3, H_4, H_5

Obr. 3.10.6

V případě, že daná plocha obsahuje "díru", lze situaci vyřešit poměrně jednoduše tak, že se vyberou dva vhodné vrcholy (každý z jiného obrysu), které se spojí dvěma hranami, jež se označí jako pomocné a neviditelné, viz obr. 3.10.7.



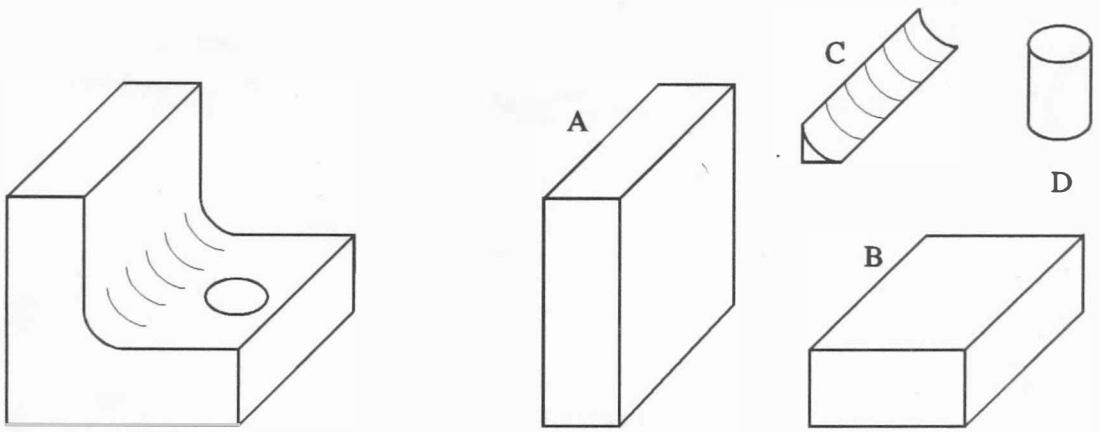
Obr. 3.10.7

Kromě rovinných ploch lze použít i plochy analytické, ať už explicitní či parametrické. Při použití analytických ploch typu koule, anuloid apod. se ve velké většině systémů tyto plochy aproximují pomocí lineárních plošek, neboť v opačném případě by bylo nutné řešit obecně nelineární soustavy rovnic. V některých aplikacích pak nemusí být popis plochy znám a pak je možné použít obecných parametrických ploch. Jejich popis a použití lze nalézt např. v [163].

Je zřejmé, že datová struktura okřídlených hran není vhodná pro zadávání dat, ale pouze pro vnitřní reprezentaci těles.

Objemová reprezentace těles

Objemové reprezentace těles lze rozdělit na reprezentace pomocí octree a tzv. CSG stromů.



Obr. 3.10.8

Reprezentace pomocí CSG stromů

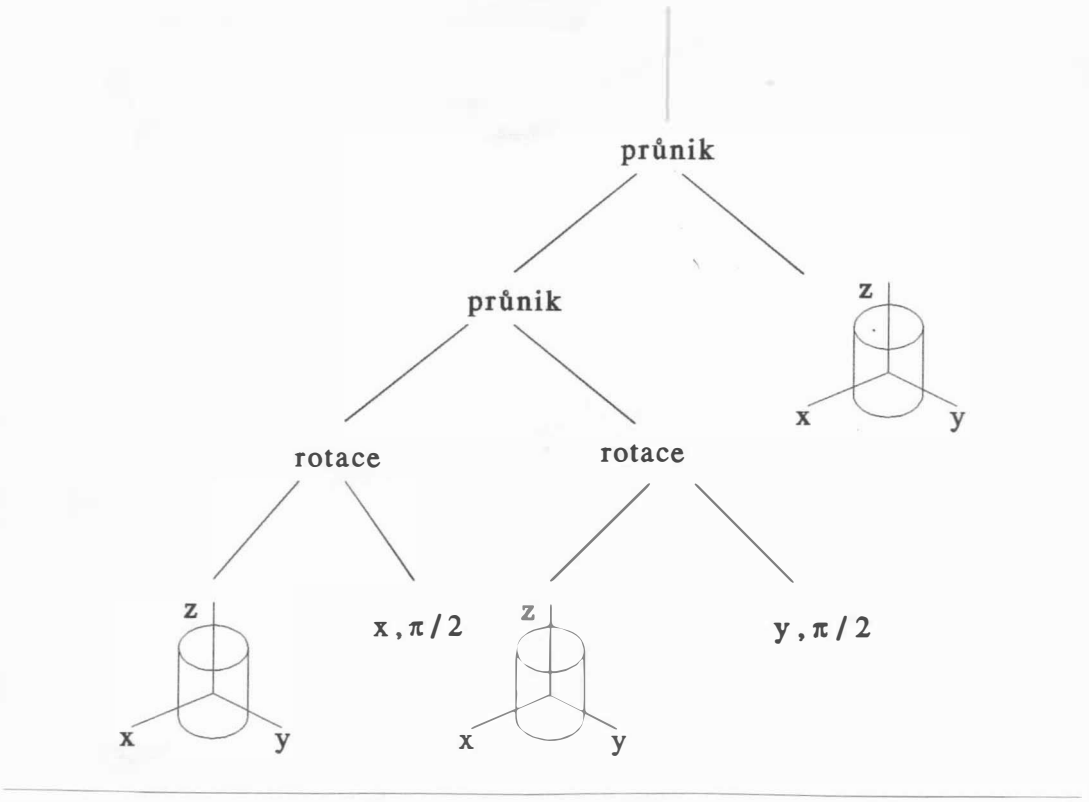
Až dosud uvedené techniky reprezentace těles umožňovaly jen velmi těžko výpočet mechanických veličin, jako je těžiště, objem tělesa apod. Tyto možnosti poskytuje tzv. objemová reprezentace těles, jejíž jedna reprezentace je též známa pod názvem "konstruktivní geometrie těles" (Constructive Solid Geometry - CSG). Základem reprezentace jsou tzv. základní tělesa, kterými mohou být koule, válce, kužele, anuloidy, kvádry, ale také poloprostory. S těmito tělesy lze pak provádět

základní geometrické transformace, jako je: posuv, rotace, zmenšení či zvětšení. Navíc je pak možné provádět i množinové operace sjednocení, průniku, rozdílu apod. K popisu základních těles se používá analytického vyjádření. Složená tělesa jsou pak reprezentována pomocí binárních hierarchických struktur nebo tzv. CSG stromů, jejichž uzly obsahují informace o typu množinové operace, nebo o použité geometrické transformaci a jejích parametrech.

Z obr.3.10.8 je zřejmé, že výsledné těleso lze složit jako

$$((A \cup B) \cup C) - D$$

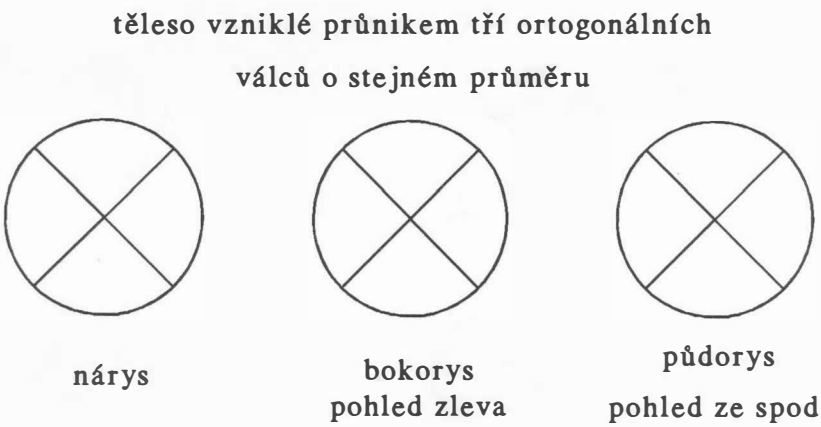
neuvažujeme-li geometrické transformace, tj. operace posuvu apod.



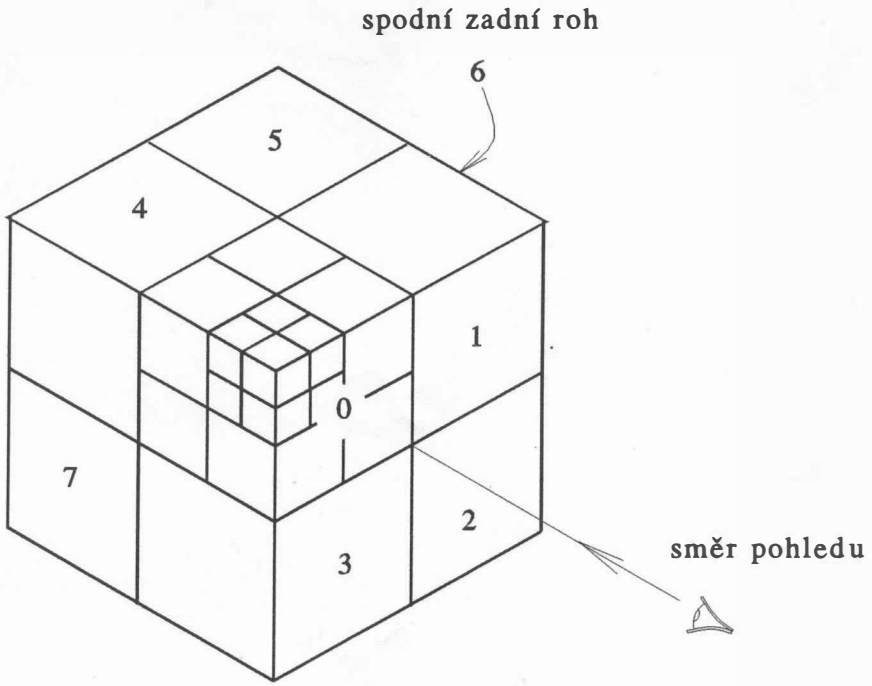
Obr.3.10.9

Výhodou CSG stromů je úspornost popisu a snadnost modelování, avšak při zobrazování vznikají problémy, které lze řešit buď vytvořením hraniční reprezentace, anebo použitím technik, které hraniční reprezentaci nepotřebují, např. ray tracing, viz kap.7.7, a které se stále častěji používají. Jako

jednoduchý příklad na CSG stromy uvažme průnik tří ortogonálních válců. Pak CSG strom má tvar uvedený na obr.3.10.9. Výsledné těleso má pak všechny ortogonální průměty stejné a jeho tvar je ukázán na obr.3.10.10.



Obr.3.10.10



Obr.3.10.11

Stromová struktura OCTREE

Stromová datová struktura, nazývaná octree, se používá k reprezentaci prostorových objektů. Vznikla prostým rozšířením datové struktury quadtree o další rozměr a vlastně jde o výčet prostorových elementárních prvků, které jsou tělesem obsazeny. Je zřejmé, že jde v zásadě o způsob definice daného objektu. V datové struktuře odpovídá každému uzlu prostorová oblast, která je reprezentována krychlí.

Číslování oblastí je ukázáno na obr.3.10.11 spolu s?naznačením postupu dalšího možného dělení. Toto dělení je možné až do takové úrovně, kdy prostorový element již nelze dále dělit. Tento element se nazývá **voxel**, což je vlastně rozšířením představy pixelu o další třetí rozměr. Výhodou této datové reprezentace je to, že umožňuje snadné určení objemu, těžiště apod.

Pro zobrazení však již nelze přímo použít techniky uvedené v kap.5. Nejjednodušší způsob zobrazení datové struktury octree je její převod do datové struktury quadtree, kterou lze již zobrazit elementárním způsobem. Vlastní převod lze popsat např. rekurzivním postupem [8], viz alg.3.10.1.

```
type oct_node_ptr = ^oct_node;
     oct_entry = record
       case homogeneous boolean of
                    { představuje uzel stromu celou oblast ? }
          true: (color: integer); { barva prostorové oblasti }
          false: (child: oct_node_ptr)
       end;
      end; { record }
     oct_node = array [0..7] of oct_entry;
     quad_node_ptr = ^quad_node;
     quad_entry = record
       case homogeneous: boolean of
          true: (color: integer);
          false: (child: quad_node_ptr)
       end;
      end; { record }
     quad_node = array [0..3] of quad_entry;
var newquadtree: quad_node_ptr;      backcolor: integer;
```

```
procedure convert_oct_to_quad ( octree: oct_node;
                                var quadtree: quad_node);
var k: integer;
begin
  for k := 0 to 3 do
  begin  quadtree[k].homogeneous := true;
     if octree[k].homogeneous
     then
       if (octree[k].color > -1)
       then {přední oktant je plný }
          quadtree[k].color:=octree[k].color
       else    { přední oktant je prázdný }
          if octree[k+4].homogeneous
          then
             if (octree[k+4].color > -1)
             then {přední oktant je prázdný, zadní plný }
                   quadtree[k].color:=octree[k+4].color
             else { přední a zadní oktant je prázdný }
                   quadtree[k].color:=backcolor
          else
          begin { přední prázdný, zadní smíšený }
             quadtree[k].homogeneous :=false;
             new(newquadtree);
             quadtree[k].child:= newquadtree;
             convert_oct_to_quad(octree[k+4].child^,newquadtree^);
          end
     else
     begin { přední smíšený, zadní není neznám }
        quadtree[k].homogeneous :=false;  new(newquadtree);
        quadtree[k].child:= newquadtree;
        convert_oct_to_quad ( octree[k+4].child^, newquadtree^);
        convert_oct_to_quad ( octree[k].child^, newquadtree^);
     end
  end  { cyklus for }
end;
```

Algoritmus 3.10.1

Podrobnější informace o použití stromových datových struktur lze nalézt v dostupné literatuře, viz např. [165]-[167].

Literatura

[1] Akimoto,T., Mase,K., Hashimoto,A., Suenaga,Y.: Pixel Selected Ray Tracing, in [64], 1989, pp.29-50.

[2] Acland,B.,West,N.: Real Time Animation on a Frame Store Display System, Computer Graphics (SIGGRAPH'80), Vol.14, 1980, pp.182-188.

[3] Ackland,B.,West,N.: The Edge Flag Algorithm - A Method for Raster Scan Displays, IEEE Trans. on Computers, Vol. C30, 1981, pp.41-48.

[4] ACM Computer Science Conference, Proceedings of the 1984, Philadelphia, ACM 1984.

[5] Angell,I.O.: A Practical Introduction to Computer Graphics, MacMillan Press, 1985.

[6] Alvisi,L., Casciola,G.: Two and Four Array Mask Algorithms in Practice, TR Dept. of Mathematics, Univ. of Bologna, 1988.

[7] Alvisi,L., Casciola,G.: TAM rivisitato: un metodo rapido ed astto per la rappresentazione prospettica di superfice, PIXEL No.10, 1988, pp.15-24.

[8] Baker,M.P.,Hearn,D.: Computer Graphics, Prentice Hall, International Edition, 1986.

[9] Barrett,R.C.,Jordan,B.W.: A Cell Organized Raster Display for Line Drawings, CACM, Vol.17, 1974, pp.70-77.

[10] Bartsch,H.J.: Matematické vzorce, SNTL, 1971.

[11] Berger,M.: Computer Graphics with Pascal, Benjamin Cummings Publishing Comp., Menlo Park, 1986.

[12] Bergeron,R.D.(Ed.): SIGGRAPH'82 Conference Proceedings, ACM SIGGRAPH, Vol.16, No.3, July 1982.

[14] Birren,F.: Creative Color, Van Nostrand Reinhold Co., New York, 1961.

[15] Blinn,J.F.: Models of Light Reflection for Computer Synthesized Pictures, Computer Graphics (SIGGRAPH'77), Vol.11, 1977, pp.191-198.

[16] Blinn,J.F.: Computer Display of Curved Surfaces, PhD Thesis, Univ. of Utah, 1978.

[17] Blinn,J.F.,Newell,M.E.: Clipping Using Homogeneous Coordinates, Computer Graphics (SIGGRAPH'78), Vol.12, 1978, pp.245-251.

[18] Bono,P.R.,Herman,I.(Ed.): GKS Theory and Practice, EUROGRAPHICS, 1987.

[19] Bouknight,J.: A Procedure for Generation of Three Dimensional Half-toned Computer Graphics Presentation, CACM, Vol.13, 1970, pp.527-563.

[20] Butland,J.: Surface Drawing Made Simple, CAD Journal, Vol.11, 1979, pp.19-22.

[21] Casciola,G.: Basic Concepts to Accelerate Line Algorithms, Computer & Graphics, Vol.12, 1988, pp.489-502.

[22] Castle,C.M.A.,Pitteway,M.L.V.: An Application of Euklid's Algorithm to Drawing Straight Lines, in[39],, 1985, pp.135-139.

[23] Catmull,E.E.: A Subdivision Algorithm for Computer Display of Curved Surfaces, PhD Thesis, Univ. of Utah, 1974.

[24] Catmull,E.: A Tutorial on Compensation Tables, SIGGRAPH'79, Computer Graphics, Vol.14, No.3, July 1980, pp.279-285.

[25] Cheng,F.,Yen,Y.: A Parallel Line Clipping Algorithm and Its Implementation, in [41], 1989.

[26] Clark,J.H.: The Geometry Engine: A VLSI Geometry System for Graphics, Computer Graphics (SIGGRAPH'82), Vol.16, 1982, pp.127-133.

[27] Claussen,U.: On Reducing the Phong Shading Method, in [64], 1989, pp.333-380.

[28] Computational Geometry, Proceedings of the Fifth Annual Conference, ACM, 1989.

[29] Cook,R.L.: A Reflection Model for Realistic Image Synthesis, PhD. Thesis, Cornell Univ., 1982.

[30] Cyrus,M.,Beck,J.: Generalized Two and Three Dimensional Clipping, Computers & Graphics, Vol.3, No.1, 1979, pp.23-28.

[31] Devillers,O.: The Macro Regions: An Efficient Space Subdivision Structure for Ray Tracing in [64], 1989, pp.27-38.

[32] Drs,L.,Všetečka,J.: Objektivem počítače, SNTL, 1981.

[33] Duce,D.A.,Jancene,P.(Ed.): EUROGRAPHICS'89, Conference Proceedings, North Holland Publ. Comp., 1989.

[34] Dunlavey,M.R.: Efficient Polygon Filling Algorithms for Raster Displays, Trans. on Graphics, Vol.2., 1983, pp.264-273.

[35] Ellis,T.M.R.,Semenkov,O.I.(Ed.): Advances in CAD/CAM, North Holland Publ. Comp., IFIP, 1983.

[36] Encarnacao,J.,Schlechtendahl,E.G.: Computer Aided Design - Fundamentals and System Architectures, Springer Verlag, 1983.

[37] Enderle,G.,Kansy,K.,Pfaff,G.: Computer Graphics Programming, Springer Verlag, 1984.

[38] Enderle,G.,Grave,M.,Lillenhagen,F.(Ed.): Advances in Computer Graphics I, Springer Verlag, 1986.

[39] Earnshaw,R.A.(Ed.): Fundamental Algorithms for Computer Graphics, NATO ASI Series, Series F, Vol.17., Springer Verlag, 1985.

[40] Earnshaw,R.A.(Ed.): Theoretical Foundations of Computer Graphics and CAD, NATO ASI Series, Series F, Vol.40, Springer Verlag, 1987.

[41] Earnshaw,R.A.,Wyvill,B.(Ed.): New Advances in Computer Graphics, Proceedings of Computer Graphics International 89, Springer Verlag, 1989.

[42] Fitzgerald,W.,Gracer,F.,Wolfe,R.: GRIN: Interactive Graphics for Modeling Solids, IBM Res.& Devel., Vol.25, No.4., July 1981, pp.281-294.

[43] Floyd,R.,Steinberg,L.: An Adaptive Algorithm for Spatial Gray Scale, SID 1975, Int. Symp. Dig. Techn., 1975, pp.36-37.

[44] Foley,J.D.,van Dam,A.: Fundamentals of Interactive Computer Graphics, Addison-Wesley, 1984.

[45] Foley,J.D.: Next Generation User Interface Development Tools, in [64], 1989, pp.537-538.

[46] Franklin,W.R.,Lewis,H.R.: 3-D Graphic Display of Discrete Spatial Data by Prism Maps, Computer Graphics (ACM SIGGRAPH'78), Vol.12, No.3, August 1978.

[47] Franklin,W.R.: An Exact Hidden Sphere Algorithm That Operates in Linear Time, Computer Graphics and Image Processing, Vol.15, 1981, pp.364-379.

[48] Fuchs,H.(Ed.): SIGGRAPH'81 Conference Proceedings, ACM, SIGGRAPH, Vol.15, No.3, August 1981.

[49] Gervantz,M.,Purgathofer,W.: A Simple Method for Color Quantization: Octree Quantization, Proceedings Computer Graphics International'88, Springer Verlag, 1988, pp.219-231.

[50] Getto,P.: Fast Ray Tracing of Unevaluated Constructive Solid Geometry Models, in [41], 1989, pp.563-578.

[51] Ghazanfarpour,D.,Peroche,B.: Anti - aliasing by successive Steps with a Z-Buffer, in [64], 1989, pp.235-244.

[52] Gonzales,R.G.,Wintz,P.: Digital Image Processing, Addison Wesley, 1977.

[53] Gottlieb,M: Hidden Line Subroutines for Three Dimensional Plotting, Byte, Vol.3, No.5, 1978, pp.49-58.

[54] Gorelik,A.G.: Logical Functions as a Means of Modelling Geometrical Objects, in [35], pp.135-151.

[55] Granát,L.,Sechovský,H.: Počítačová grafika, SNTL, 1980.

[56] Graphical Kernel System for Three Dimensions (GKS-3D) - Functional Description, Norma ISO/TC97/SC21 IS 7942.

[57] Greenaway,D.S.,Warman,E.A.(Ed.): EUROGRAPHICS'82 Conference Proceedings, North Holland Publ.Comp., 1982.

[58] Greenberg,D.P.,Meyer,G.W.: Perceptual Color Spaces for Computer Graphics, Computer Graphics, Vol.14, 1980, pp.254-261.

[59] Greenberg,D.P.,Marcus,A.,Schmidt,A.,Gorter,V.: The Computer Image-Applications of Computer Graphics, Addison Wesley, 1982.

[60] Greenberg,D.P.,Meyer,G.W.: Color Education and Color Synthesis in Computer Graphics, Color Research and Application, Vol.11, John Wiley & Sons, Supplement 1986, pp.S39-44.

[61] Greenberg,D.P.: Advances in Global Illumination Algorithms, in [64], 1989, pp.401-402.

[62] ten Hagen,P.J.W.,Tomiyama,T.(Ed.): Intelligent CAD Systems I, Springer Verlag, 1987.

[63] Hamlin,G., Gear,C.: Raster Scan Hidden Surface Algorithm Techniques, Computer Graphics (SIGGRAPH'77), Vol.11, 1977, pp.206-213.

[64] Hansmann,W.,Hopgood,F.R.A.,Strasser,W.(Ed.): EUROGRAPHICS'89 , Conference Proceedings, North Holland Publ. Comp., 1989.

[65] Haralick,R.M.: Pictorial Data Analysis, NATO ASI, Series F, Vol.4., Springer Verlag, 1983.

[66] Harrington,S.: Computer Graphics - A Programming Approach, McGraw Hill, 1987.

[67] Heckbert,P.: Color Image Quantization for Frame Buffer Display, Computer Graphics, Vol.16, No.3, July 1982, pp.297-305.

[68] Hilbert,R.:Construction and Display of Three Dimensional Polygon Histograms, Computer Graphics, Vol.15, No.2, July 1981.

[69] Hopgood,F.R.A.,Duce,D.A.,Gallop,J.R.,Sutcliffe,D.C.: Introduction to the Graphical Kernel System (GKS), Academic Press, 1983.

[70] Hopgood,F.R.A.,Hubbolt,R.J.,Duce,D.A.(Ed.): Advances in Computer Graphics II, Springer Verlag, 1986.

[71] Hubbolt,R.J.(Ed.): EUROGRAPHICS'82, Tutorial Notes, EUROGRAPHICS Assos., Geneva, 1982.

[72] Hubbolt,R.J.,Arnold,A.C.,Hewitt,W.T.: Interactive Computer Graphics - Course Notes, Univ. of Manchester, Computer Graphics Unit, 1984.

[73] Inselberg,A.: The Plane with Parallel Coordinates, The Visual Computer, Vol.1, 1985, pp.69-91.

[74] Inselberg,A.,Comut,T.,Reif,M.: Convexity Algorithms in Parallel Coordinates, JACM, Vol.34, No.4, October 1987, pp.765-801.

[75] Inselberg,A.,Dimsdale,B.: Parallel Coordinates for Visualizing Multi-Dimensional Geometry, in [84], 1987, pp.25-44.

[76] Jarvis,J.F.,Judice,C.N.,Ninke,W.H.: A Survey of Techniques for the Display of Continues Tone Pictures on Bilevel Displays, Computer Graphics and Image Processing, Vol.5, 1976, pp.13-40.

[77] Jevans,D.A.J.: Optimistic Multiprocessor Ray Tracing, in [41], 1989, pp.507-522.

[78] Joseph,H.: Computer Graphics Hardware - Introduction and State of the Art, EUROGRAPHICS'87 Tutorial, EUROGRAPHICS, 1987.

[79] Kay,D.S.: Transparency, Refraction and Ray Tracing for Computer Synthesized Images, PhD Thesis, Cornell Univ., 1979.

[80] Kilgour,A.C.: Unifying Vector and Polygon Algorithm for Scan Conversion and Clipping, TR CSC/87/R7, Univ. of Glasgow, May 1987.

[81] Knuth,D.E.: Digital Halftones by Dot Diffusion, ACM Trans. on Graphics, Vol.6, No.4, October 1987, pp.245-273.

[82] Kubo,S.: Continuous Color Presentation Using a Low Cost Ink Jet Printers, in [84],1987, pp.348.

[83] Kunii,T.L.(Ed.): Frontiers in Computer Graphics, Springer Verlag, 1985.

[84] Kunii,T.L.(Ed.): Computer Graphics 1987, Proceedings of the 5th International Conference on Computer Graphics, Springer Verlag, 1987.

[85] Liang,Y.D.,Barsky,B.A.: An Analysis and Algorithms for Polygon Clipping, CACM, Vol.26, No.11, 1984, pp.868-876.

[86] Liang,Y.D.,Barsky,B.A.: A New Concept and Method for Line Clipping, ACM Transaction on Graphics, Vol.3, No.1, 1984, pp.1-22.

[87] Lewell,J.: Computer Graphics - `A Survey of Current Techniques and Applications, Orbis Publ. Ltd., London, 1985.

[88] Mach,K.D.,Petty,J.S.: Contouring and Hidden Line Algorithms for Vector Graphic Display, Rep. AFAPL-TR-77-3, 1977.

[89] Měření barev, ČSN 01 1718.

[90] Meyer,G.W.: Wavelength Selection for Synthetic Image Generation, Computer Vision, Graphics and Image Processing, Vol.41, 1988, pp.57-79.

[91] Meyer,G.W.: Tutorial on Color Science, The Visual Computer, Vol.2, Springer Verlag, pp.278-290.

[92] Murch,G.M.: Human Factors of Color Displays, TR, Tektronix, Oregon, 1989.

[93] Murch,G.: Color Matching of Display and Printer, in [64], 1989, pp.313-314.

[94] Newmann,W.M.,Sproull,R.F.: Principles of Interactive Computer Graphics, 2nd ed., McGraw Hill, 1981.

[95] Nicholl,T.M.,Lee,D.T.,Nicholl,R.A.: An Efficient New Algorithm for 2D Line Clipping: Its Development and Analysis, ACM Computer Graphics, Vol.21, No.4, July 1987, pp.253-262.

[96] O'Bara,R.M.,Abi-Ezzi,S.: An Analysis of Modeling Clip, in [64], 1989, pp.367-380.

[97] Pavlidis,T.: Graphics and Image Processing, Springer Verlag, 1982.

[98] Peitgen,H.O.: The Impact of Fractal Geometry for Computer Graphics, in [64], 1989, pp.315-316.

[99] Perdue,L.: Supercharging Your PC, McGraw Hill, 1987.

[100] Phillips,R.L.(Ed.): SIGGRAPH'78, Conference Proceedings, ACM SIGGRAPH, Vol.12, No.3, August 1978.

[101] Pins,M.,Hild,H.: Variation on Dither Algorithm, in [64], 1989, pp.381-392.

[102] Pitteway,M.L.V.: The Algebra of Algorithms - A New Toy for the Theoretician?, IUCC Bulletin, Vol.1, 1979, pp.139-144.

[103] Pitteway,L.M.V.,Watkinson,D.J.: Bresenham's Algorithm with Gray Scale, CACM, Vol.23, 1980, pp.625-626.

[104] Plastock,R.A.,Kaley,G.: Theory and Problems of Computer Graphics, McGraw Hill, New York, 1986.

[105] Pollack,B.W.(Ed.): SIGGRAPH'79 Conference Proceedings, ACM, SIGGRAPH, Vol.13., No.2., August 1979.

[106] Popsel,J.,Hornung,C.: Highlighting Shading - Lighting and Shading in a PHIGS+/PEX Environment, in [64], 1989, pp.317-332.

[107] Preparata,F.P.,Shamos,M.I.: Computational geometry - An Introduction, Springer Verlag, 1985.

[108] Rogers,D.F.,Adams,J.A.: Mathematical Elements for Computer Graphics, McGraw Hill, New York, 1976, 2. vydání 1990.

[109] Rogers,D.F.: Procedural Elements for Computer Graphics, McGraw Hill, 1985.

[110] Rogers,D.F.,Earnshaw,R.A.(Ed.): Techniques for Computer Graphics, Springer Verlag, 1987.

[111] de Ruiter,M.M.(Ed.): Advances in Computer Graphics III, Springer Verlag, 1988.

[112] Santo,H.P.: Métodos Gráficos e Geometria Computacionais, Dinalivro, Lisboa, 1985.

[113] Sheppard,J.: Human Color Perception - A Critical Study of the Experimental Foundation, Elsevier, New York, 1968.

[114] Shirai,Y.: Three Dimensional Computer Vision, Springer Verlag, 1987.

[115] Skala,V.: An Interesting Modification to the Bresenham Algorithm for Hidden-Line Problem Solution, in [39], 1985, pp.593-602.

[116] Skala,V.: An Intersecting Modification to the Bresenham Algorithm, Computer Graphics Forum, Vol.6, No.4, 1987, pp.343-347.

[117] Skala,V.: Algorithms for 2D Line Clipping, in [41], 1989, pp.121-128.

[118] Skala,V.: Algorithms for 2D Line Clipping, in [64], 1989, pp.355-367.

[119] Slavkovský,P.: Problém viditelnosti v počítačové grafice, kandidátská disertační práce, MFF UK, Bratislava, 1987.

[120] Smith,A.R.: Color Gamut Transform Pairs, SIGGRAPH'78 Conference Proceedings, ACM, SIGGRAPH, 1978, pp.12.

[121] Smith,A.R.: Tint Fill, Computer Graphics (SIGGRAPH'79), Vol.13, 1979, pp.276-283.

[122] Světelně-technické názvosloví, ČSN`36 0000.

[123] Staudhammer,J.,Livadas,P.E.: Computer Graphics - A Tutorial, The Second International Conf. on Computers and Applications, Beijing, China, 1987.

[124] Strasser,W.(Ed.): Advances in Computer Graphics Hardware I, Springer Verlag, 1987.

[125] Sutherland,I.E.,Hodgman,G.W.: Reentrant Polygon Clipping, CACM, Vol. 17., No.1, January 1974, pp.32-42.

[126] Sutherland,I.E.,Sproul,R.F.,Schumacker,R.A.: A Characterization of Ten Hidden-Surface Algorithms, Computing Surveys, Vol.6, 1974, pp.1-55.

[127] Tanner,P.(Ed.): SIGGRAPH'83, Conference Proceedings, ACM, SIGGRAPH, Vol.17, No.3, July 1983.

[128] Teunissen,W.J.M.: HIRASP - A Hierarchical Modelling System for Raster Graphics, PhD Thesis, 1988.

[129] Thalmann,N.M.,Thalmann,D.(Ed.): Computer Generated Images, Proceedings of Graphics Interface'85, Springer Verlag, 1985.

[130] Thomas,J.J.(Ed.): SIGGRAPH'80, Conference Proceedings, ACM, SIGGRAPH'80, Vol.14,, No.3, July 1980.

[131] Toifl,J.: Grafické vstupní zařízení počítače, Výběr informací, č.2, SNTL, 1973.

[132] Toifl,J.: Grafické výstupní zařízení počítače, Výběr informací, č.4, SNTL, 1973.

[133] UNIRAS - Firemní materiály firmy European Software Contractors, 1985.

[134] UNIRAS - Universal Raster Report, Firemní materiály, 1985.

[135] Vandoni,C.E.(Ed.): EUROGRAPHICS'85, Conference Proceedings, North Holland Publ. Comp., 1985.

[136] Vít a kol.: Televizní technika, SNTL, Praha, 1979.

[137] Warnock,J.E.: A Hidden Line Algorithm for Halftone Picture Representation, Univ. of Utah, Comp.Sci.Dept., Report TR 4-5, May 1968.

[138] Warnock,J.E.: A Hidden Surface Algorithm for Computer Generated Halftone Pictures, Univ. of Utah, Comp.Sci.Dept., TR 4-15, June 1969.

[139] Watkins,G.S.: A Real Time Visible Surface Program, Univ. of Utah, Comp.Sci.Dept., Report UTEC-CSC-70-101, June 1970.

[140] Watkins,S.L.: Masked Three Dimensional Plot Program with Rotation, Algorithm 483, CACM, Vol.17, 1974, pp.520-523.

[141] Watters,G.,Willis,P.: Scan Converting Extruded Lines at Ultra High Definition, Computer Graphics Forum, Vol.6, No.2, May 1987, pp.133-140.

[142] Weiler,K.,Atherton,P.: Hidden Surface Removal Using Polygon Area Sorting, Computer Graphics (SIGGRAPH'77), Vol.11, 1977, pp.214-222.

[143] Whitted,T.: An Improved Illumination Model for Shaded Display, CACM, Vol. 23,1980, pp.343-349.

[144] Williams,H.: Hidden-Line Plotting Program, Algorithm 420, CACM, Vol.15, 1972, pp.100-103.

[145] Wright,T.J.: A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables, IEEE Trans. on Computers, Vol.C-22, 1973, pp.28-33.

[146] Wyvill,G.,Sharp,P.: Fast Antialiasing of Ray Traced Images, in [41], 1989, pp.579-590.

[147] Xu,H.,Peng,Q.S.,Liang,Y.D.: Accelerated Radiosity Method for Computer Environment, in [64], 1989, pp.51-62.

[148] Zhang,J.: A Fast Hidden-Line Removal Algorithm, in [41], 1898, pp.591-602.

[149] Blinn,J.F.,Carpenter,L.C.,Lane,J.M.,Whitted,T.: Scan Line Methods for Displaying Parametrically Defined Surfaces, CACM, Vol.23, pp.23-34, 1980.

[150] Phong,B.T.: Illumination for Computer Generated Images, PhD Thesis, Univ. of Utah, 1973.

[151] Carpenter,L.C.,Lane,J.M.: A Generalized Scan Line Algorithm for the Computer Display of Parametrically Defined Surfaces, Computer Graphics and Image Processing, Vol.11, pp.290-297,1979.

[152] Gouraud,H.: Computer Display of Curved Surfaces, PhD Thesis, Univ. of Utah, 1971.

[153] Kay,Douglas Scott: Transparency, Refraction and Ray Tracing for Computer Synthesized Images, MSc Thesis, Cornell Univ., 1979.

[154] Kay, Douglas Scott, Greenberg,D.: Transparency for Computer Synthesized Images, Computer Graphics (SIGGRAPH'79 Proceedings), Vol.13, pp. 158-164, 1979.

[155] Beckmann,P.,Spizzichiono,A.: Scattering of Electromagnetic Waves from Rough Surfaces, MacMillan Press, New York, 1963, pp.1-33,70-98.

[156] Cook,R.L.,Torrance,K.E.: A Reflectance Model for Computer Graphics, ACM on Graphics, Vol.1., pp.7-24, 1982.

[157] Torrance,K.E., Sparrow,E.M.: Polarization, directional distribution, and off-specular peak phenomena in light reflected from roughened surfaces, Journal of the Optical Society of America, Vol.57,7 (July 1966),916-925.

[158] Torrance,K.E., Sparrow,E.M.: Theory for off-specular reflectiom from roughened surfaces, Journal of the Optical Society of America, Vol.57,9 (Sept. 1967),1105-1114.

[159] Trowbridge,T.S., Reitz,K.P.: Average irregularity representation of roughened surface for ray reflection, Journal of the Optical Society of America, Vol.65, 5 (May 1975),531-536.

[160] Agoston,G.A.: Color Theory and Its Applicaton in Art and Design, Springer Verlag 1987

[161] Baldwin,L.: Color Consideration, BYTE September 1984, pp.227-246

[162] Cahill,B.: Drawing on the 8514/A, BYTE March 1990, pp.279-289

[163] Drs,L.: Plochy ve výpočetní technice, Matematický seminář SNTL, SNTL 1984

[164] Skala,V.: Filling and Hatching Operations for Non-Convex Areas with Conic Edges for the Raster Environment, ACM-SIGGRAPH Workshop Lisboa Local Group, Lisboa, Portugal, 1988

[165] Samet,H.: The Quadtree and Related Hierarchical Data Structures, Computing Surveys, Vol.16, No.2, June 1984, pp.187-260

[166] Samet,H., Webber,R.E.: Storing a Collection of Polygons Using Quadtrees, ACM Trans. on Graphics, Vol. 4, No. 3, July 1985, pp. 182-222

[167] Kessener,L.R.A., Peters,F.J.,van Lierop,M.L.P.(Ed.): Data Structures for Raster Graphics, Proceedings of a Workshop held at Steensel, Springer Verlag, 1986

[168] Baumgart,B.G.: A Polyhedron Representation for Computer Vision, Proc.Nat.Comp.Conf., AFIPS 1975, pp.589-596

[169] Kilgour,A.: Techniques for Modelling and Displaying 3D Scenes, Technical Report, Univ. of Glasgow, 1986

[170] Pratt,M.J.: Types of Modeller, Technical Report, Dept. of Mathematics, Cranfield Inst. of Technology, Cranfield U.K., September 1982

[171] Thalmann,N.M.,Thalmann,D.(Ed.): Computer Animation, Theory and Practice, Springer Verlag, 1985.

[172] Greenberg,D.P.: Ray Tracing and Radiosity, State of Art in Image Synthesis, course notes, SIGGRAPH'86, ACM, 1986

[173] Greenberg,D.P.,Cohen,M.F.,Torrance,K.E.: Radiosity: A Methods for Computing Global Illimination, The Visual Computer, Vol.2., No.5.,September 1986.

[173] Burgoon,D.A.: Global Illumination Modeling Using Radiosity, Hewlett Packard Journal, December 1989, pp.78-88.

[174] Goral,C.M.,Torrance,K.E.,Greenberg,D.P.,Battaile,B: Modelling the Interaction of Light between Diffuse Surfaces, SIGGRAPH'84, ACM, 1984.

[175] Cohen,M.F.,Greenberg,D.P.: The Hemi-Cube: Radiosity Solution for Complex Environments, SIGGRAPH'85, ACM, 1985.
[176] Graphics Databook, firemní materiály INMOS SGS-Thompson, 1990.
[177] Systems Solutions, firemní materiály IChips Europe, 1990.
[178] Jarvis,J.F., Roberts,C.S.: A New Technique for Displaying Continuous Tone Images on a Bilivel Display, IEEE Trans. Communic., Vol.24, pp.891-898, 1976.
[179] Abhyankar,S.S.,Chandrasekar,S.,Chandru,V.: Improper Intersection of Algebraic Curves, ACM Trans. on Graphics, vol.9, No.2, 1990, pp.147-159.
[180] Andreev,R.D.: Algorithm for Clipping Arbitrary Polygons, Computer Graphics Forum, Vol.7, No.3, 1988, pp.183-192.
[181] Brunet,P.,Navazo,I.: Solid Representation and Operation Using Extended Octrees, ACM Trans. on Graphics, vol.9, No.2, 1990, pp.170-197.
[182] Day,A.M.: The Implementation of an Algorithm to find the Convex Hull of a Set of Three Dimensional Points, ACM Trans. on Graphics, vol.9, No.1, 1990, pp.105-132.
[183] Dobkin,D.P.,Levy,S.L.F.,Thurston,W.P., Wilks,A.R.: Contour Tracing by Piecewise Linear Approximations, ACM Trans. on Graphics, vol.9, No.4, 1990, pp.389-423.
[184] Edelsbrunner,H.,Mucke,E.P.: Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms, ACM Trans. on Graphics, vol.9, No.1, 1990, pp.66-104.
[185] Falcidieno,B.,Floriani,L.: A Hierarchical Boundary Model for Solid Object Representation, ACM Trans. on Graphics, vol.7, No.1, 1988, pp.42-60.
[186] Fournier,A.,Fussel,D.: On the Power of the Frame Buffer, ACM Trans. on Graphics, vol.7, No.2, 1988, pp.103-128.
[187] Gaude,S.,Hobson,R.,Chilka,P.,Calvert,T.: Multiprocessor Experiments for High Speed Ray Tracing ACM Trans. on Graphics, vol.7, No.3, 1988, pp.151-179.
[188] Heal,B.: Hidden Octree Removal, Computer Graphics Forum, Vol.7, No.3, 1988, pp.199-206.
[189] Herman,I.,Reviczky,J.: Some Remarks on the Modelling Clip Problem, Computer Graphics Forum, Vol.7, No.4, 1988, pp.265-272.

[190] Herman,I.: On The Projective Invariant of Conics in Computer Graphics, Computer Graphics Forum, Vol.8, No.4, 1990, pp.301-314.

[191] Hobby,J.D.: Rasterization of Nonparametric Curves, ACM Trans. on Graphics, vol.9, No.3, 1990, pp.3262-277.

[192] Kuijk,A.A.M.,Blake,E.H.: Faster Phong Shading via Angular Interpolation, Computer Graphics Forum, Vol.8, No.4, 1990, pp.315-325.

[193] Lamming,L.,Rhodes,W.L.: A Simple Method for Improved Color Printing of Monitor Images, ACM Trans. on Graphics, vol.9, No.4, 1990, pp.345-375.

[194] Levoy,M.: Efficient Ray Tracing of Volume Data,, ACM Trans. on Graphics, vol.9, No.3, 1990, pp.245-261.

[195] Nicholl,R.A., Nicholl,T.M.: Performing Geometric Transformations by Program Transformation, ACM Trans. on Graphics, vol.9, No.1, 1990, pp.28-40.

[196] Preparata,F.P.,Vitter,J.S.,Yvinec,M.: Computation of the Axial View of a Set of Isothetic Parallelopides, ACM Trans. on Graphics, vol.9, No.3, 1990, pp.278-300.

[197] Pun,T.,Blake,E.: Relationships Between Image Synthesis and Analysis: Towards Unification?, Computer Graphics Forum, Vol.9, No.2, 1990, pp.149-164.

[198] Rokne,J,G.,Wyvill,B.,Wu,X.: Fast Line Scan Conversion, ACM Trans. on Graphics, vol.9, No.4, 1990, pp.376-388.

[199] Rossignac,J.,Voelcker,H.B.: Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection, and Shading Algorithms, ACM Trans. on Graphics, vol.8, No.1, 1989, pp.51-87.

[200] Rushmeier,H.E.,Torrance,K.E.: Extended the Radiosity Method to Include Specularly Reflecting and Transluent Materials, ACM Trans. on Graphics, vol.9, No.1, 1990, pp.1-17.

[201] Stone,M.C.,Cowan,W.B.,Beatty,J.C.: Color Gamut Mapping and the Printing of Digital Color Images, ACM Trans. on Graphics, vol.7, No.4, 1988, pp.249-293.

[202] Thomas,D.,Netravali,A.N.,Fox,D.S.: Anti-aliased Ray Tracing with Covers, Computer Graphics Forum, Vol.8, No.4, 1990, pp.315-324.

[203] Veenstra.J.,Ahuja,N.: Line Drawing of Octree Represented Objects, ACM Trans. on Graphics, vol.7, No.1, 1988, pp.61-75.

[204] Ware,C.,Cowan,W.: The RGYB Color Geometry, ACM Trans. on Graphics, vol.9, No.2, 1990, pp.226-232.

[205] Zyda,M.J.: A decomposable Algorithm for Contour Surface Display Generation, ACM Trans. on Graphics, vol.7, No.2, 1988, pp.129-148.

[206] Ježek,F.: AutoCAD - učební text, VŠSE Plzeň, 1991.

[207] Poláček,J.,Ježek,F.,Kopincová,E.: Počítačová grafika, skripta ČVUT-FS Praha, 1991.

[208] Ranklin,J.R.: Computer Graphics Software Construction, Advances in Computer Science Series, Prentice Hall, 1989.

[209] Rushmeier,H.E.,Torrance,K.E.: The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium, Computer Graphics, Vol.21, No.4, July 1987.

[210] Počítačová grafika - Názvosloví, ČSN 36 9001 část 13.

[211] Systémy zpracování informací (GKS), ČSN 36 9180.

[212] Firemní materiály firmy INTERGRAPH, June 1991.

[213] Skala,V.: Počítačová grafika I, Skripta VŠSE (Západočeská univerzita), Plzeň, 2. vydání, 1991.

[214] Skala,V.: Počítačová grafika II, Skripta VŠSE (Západočeská univerzita), Plzeň, 2. vydání, 1991.

[215] Drzaic,P.S.: Nematic Droplet Polymer Films for High Contrast coloured Reflective Displays, Displays Technology and Applications, Butterworth-Heinemann Ltd., Vol.12, No.1, January 1991, pp.2-13.

[216] Schwarz,M.W.,Cowan,W.B.,Beatty.J.C.: An Experimental Comparison of RGB, YIQ, LAB, HSV, and Opponent Color Models, ACM Transaction on Computer Graphics, Vol.6,No.2, April 1987, pp.123-158.

[217] Colour Addendum to ISO 8613 - Working Draft, ISO TC97 SC18 WG5, X3H3/88-47.

[218] Prett,V.: Cifrovaja obrobotka izobraženij, Mir, Moskva, 1982.

[219] Serba,I.: Termodynamický přístup k výpočtu osvětlení prostorové scény v počítačové grafice, seminář MOP 91, 1991.

[220] Sochor,J.: Sledování paprsku ve 3D scéně, seminář MOP 91, 1991.

[221] Hall,R.: Illumination and Color in Computer Generated Imaginary, Springer Verlag, 1989.